18th International configuration workshop : proceedings of the 18th International configuration workshop within CP 2016 conference, Toulouse, France, from September 05-06, 2016

Élise Vareilles, Lars Hvam, Cipriano Forza, Caroline Becker

18th International Configuration Workshop

Proceedings of the 18th International Configuration Workshop
within CP 2016 Conference

Edited by
Élise Vareilles,
Lars Hvam, Cipriano Forza and Caroline Becker

September 5-6, 2016
Toulouse, France

Organized by
Centre Génie Industriel, Mines Albi, France

ISBN: 979-10-91526-04-3

École des Mines d'Albi-Carmaux
Campus Jarlard
Route de Teillet
Albi 81013 Cedex 09
France

18th International Configuration Workshop

Chairs

Élise Vareilles, Mines Albi, France
Lars Hvam, Technical University of Denmark, Denmark
Cipriano Forza, University of Padova, Italy
Caroline Becker, PROS Toulouse, France

Program Committee

Michel Aldanondo, Mines Albi, France
Andres Barco, Mines Albi, France
Caroline Becker, PROS, France
Jean-Guillaume Fages, COSLING S.A.S., France
Andreas Falkner, Siemens AG, Austria
Hélène Fargier, IRIT, Université de Toulouse, France
Alexander Felfernig, Graz University of Technology, Austria
Cipriano Forza, Universita di Padova, Italy
Gerhard Friedrich, Alpen-Adria-Universitaet Klagenfurt, Austria
Paul Gaborit, Mines Albi, France
Luis Garcés, Mines Albi, France
Chiara Grosso, Universita di Padova, Italy
Albert Haag, SAP SE, Germany
Alois Haselboeck, Siemens AG, Austria
Lothar Hotz, HITeC e.V. / University of Hamburg, Germany
Lars Hvam, Technical University of Denmark, Denmark
Dietmar Jannach, TU Dortmund, Germany
Manuel Korell, Finja AB / Achoice, Denmark
Thorsten Krebs, encoway GmbH, Germany
Katrin Kristjansdottir, Technical University of Denmark, Denmark
Matthieu Lauras, Mines Albi, France
Sara Safie, Technical University of Denmark, Denmark
Abdourahim Sylla, Mines Albi, France
Juha Tiihonen, University of Helsinki, Finland
Élise Vareilles, Mines Albi, France
Markus Zanker, Alpen-Adria-Universitaet Klagenfurt, Austria
Linda Zhang, IESEG School of Management, France

Special Thanks

Paul Gaborit, Mines Albi, France

Contents

Foreword

Product configuration is the task of composing product models of complex systems from parameterizable components in the mass-customization business model. This task demands for powerful knowledge representation formalisms and acquisition methods to capture the great variety and complexity of configurable product models. Furthermore, efficient reasoning methods are required to provide intelligent interactive behavior in configurator software, such as solution search, satisfaction of user preferences, personalization, optimization, diagnosis, etc.

The Configuration workshop is of interest for both, researchers working in the various fields of Artificial Intelligence as well as for industry representatives interested in the relationship between configuration technology and the business problem behind configuration and mass customization. It provides a forum for the exchange of ideas, evaluations, and experiences especially related to the use of Artificial Intelligence techniques in the configuration context.

This year's workshop is organized within the International Conference on Principles and Practice of Constraint Programming (CP2016). It is still a two-day event that continues the series of 17 successful Configuration Workshops started at the AAAI'96 Fall Symposium and continued at IJCAI, AAAI, and ECAI conferences since 1999.

A total of 17 papers has been selected for presentation on the Configuration workshop 2016. The 18th International Configuration Workshop continues the concept of Best Paper Award introduced in the last edition. As it was done in 2015, the best paper is selected in a two-phase audience vote during the last session.

Élise Vareilles
July 2016

Recommendation for product configuration: an experimental evaluation

Hélène Fargier[1] and **Pierre-François Gimenez**[2] and **Jérôme Mengin**[3]

Abstract. The present work deals with the the recommendation of values in interactive configuration, with no prior knowledge about the user, but given a list of products previously configured and bought by other users ("sale histories"). The basic idea is to recommend, for a given variable at a given step of the configuration process, a value that has been chosen by other users in a similar context, where the context is defined by the variables that have already been decided, and the values that the current user has chosen for these variables. From this point, two directions have been explored. The first one is to select a set of similar configurations in the sale history (typically, the k closest ones, using a distance measure) and to compute the best recommendation from this set - this is the line proposed by [9]. The second one, that we propose here, is to learn a Bayesian network from the entire sample as model of the users' preferences, and to use it to recommend a pertinent value.

1 Introduction

In on-line sale contexts, one of the main limiting factors is the difficulty for the user to find product(s) that satisfy her preferences, and in an orthogonal way, the difficulty for the supplier to guide potential customers. This difficulty increases with the size of the e-catalog, which is typically large when the considered products are configurable. Such products are indeed defined by a finite set of components, options, or more generally by a set of variables (or "features"), the values of which have to be chosen by the user. The search space is thus highly combinatorial. It is generally explored following a step-by-step configuration session: at each step, the user freely selects a variable that has not been assigned yet, and chooses a value. Our issue is to provide such problems with a recommendation facility, by recommending, among the allowed values for the current variable, one which is most likely to suit the user.

The problem of providing the user with an item that fulfills her preferences has been widely studied, leading to the content-based and the collaborative filtering approaches, and every variation in between [1, 22, 17]. However, these solutions can't deal with configurable products, e.g. cars, computers, kitchens, etc. The first reason is that the number of possible products is huge – exponential in the number of configuration variables. For instance, in the car configuration problem described in [4] the definition of "Traffic" delivery vans involves about 150 variables, and an e-catalog of 10^{27} feasible versions. The second reason is that the recommendation task considered in interactive configuration problem is quite different from the one addressed in classical product recommendation: the system is not asked to recommend a product (a car) but a value for the variable selected by the user[4]. Finally, the third reason is that we cannot assume any prior knowledge about the user, nor about its buying habits - complex configurable products, like cars, are not bought so often by one individual. So we have no information about similarity between users (upon which collaborative filtering approaches are based) nor on the preferences of the current user (upon which content-based filtering approaches are based).

The present work deals with the the recommendation of values in interactive configuration, with no prior knowledge about the user, but given a list of products previously configured and bought by other users ("sale histories"). The basic idea is to recommend, for a given variable at a given step of the configuration process, a value that has been chosen by other users in a similar context, where the context is defined by the variables that have already been decided, and the values that the current user has chosen for these variables. From this point, two directions can be explored. The first one is to select a set of similar configurations in the sale history (typically, the k closest ones, using a distance measure) and to compute the best recommendation from this set - this is the line proposed by [9]. The second one, yet not explored is to learn a from the entire sample a model of the users' preferences, e.g. a Bayesian net, and to use it to propose a pertinent value.

The paper is structured as follows: the basic notations are presented in Section 2. The next two sections present the two families of approaches that we have explored: Bayesian nets in Section 3 and k-closest neighbors in Section 4. They are experimentally compared and discussed in Section 5.

2 Background and notations

A configuration problem is defined by a set $\mathcal{X}$ of n discrete variables, each variable X taking its value in a finite domain $\underline{X}$. A complete configuration is thus a tuple $o \in \prod_{X\in\mathcal{X}} \underline{X}$; we denote by $\underline{\mathcal{X}}$ the set of all of them.

If W is a tuple of variables, $\underline{W}$ denotes the set of partial configurations $\prod_{X\in W} \underline{X}$; we will often denote such a partial configuration by the corresponding lower case letter w. Also, if W and V are two sets of variables, and if $w \in \underline{w}$, then $w[V]$ is the projection of w onto $V \cap W$. Furthermore, if $w \in \underline{W}$, w is said to be compatible with v if $w[V \cap W] = v[V \cap W]$; in this case we write $w \sim v$. Finally, in the case where w and v are compatible, we denote by $w.v$ the tuple

[1] IRIT, CNRS, University of Toulouse, France, email: fargier@irit.fr
[2] IRIT, CNRS, University of Toulouse, France, email: pgimenez@irit.fr
[3] IRIT, CNRS, University of Toulouse, France, email: mengin@irit.fr

[4] Note that we are not concerned here with the choice of the *variable* – this choice is under the control of the user, not under the one of the recommender system. It is worthwhile noticing that the fact that the variables are considered and assigned in a free order forbids the use of techniques based on decision trees.

that extends w with values of v for variables in $V \setminus W$ (equivalently, wv extends v with values of w for variables in $W \setminus V$).

Not all combinations represent feasible products, because of some possible feasibility or marketing constraints; let $\mathcal{P}$ be the subset of $\underline{\mathcal{X}}$ that represent feasible products. In practice, the set $\mathcal{P}$ is still a huge set.

In interactive configuration problems, the user builds the product she is interested in through a variable by variable interaction. At each step, let $\mathrm{Assigned}$ be the set of variables for which she has already chosen values, u be the tuple values assigned to these variables and $\mathrm{UnAssigned}$ the set of free variables ; then the user freely selects the next variable to be assigned (we denoted Next this variable). The system then has to:

- Compute the set of admissible values for Next: it is the set of values $v \in \underline{\mathrm{Next}}$ such that there is at least one feasible product $o \in \mathcal{P}$ with this combination of values, that is $o[\mathrm{Next}] = v$ and $o[\mathrm{Assigned}] = u$. The computation of this set has been studied elsewhere [3, 15, 16, 6].
- Propose a recommended value for Next, chosen among the admissible values.

The computation of a pertinent recommendation is the topic of the present work.

The recommendation of feature values, when any, is often limited to the proposition of a default value, generally the one advised by the seller in a static way or through a set of rules. Other approaches are based on similarity measures and propose to determine the k-nearest neighbor configuration that are similar to the current set of user requirements. These type of approaches support the idea that the user sets her most important requirements and let the system complete the configuration but seldom takes place in a process of interactive configuration (the reader shall consult [13] for a survey about recommendation technologies for configurable product).

In the context considered by this paper, *sales histories* are available, on which the system can rely to base its recommendation. Formally, a sale history is a (multi) set $\mathcal{H} \subseteq \underline{\mathcal{X}}$ of complete configurations that correspond to products that have been bought by past users (thus they are feasible, i.e. belong to $\mathcal{P}$). In the sequel, for a partial configuration u, $\#(u)$ will denote the number of configurations in $\mathcal{H}$ that extend u.

3 Recommendation with Bayesian networks

Users have different preferences, depending on the taste and the environment of the user, which make them prefer different products - hence a large variety of products in the histories. We do not have any information about their taste, nor do we use any information about their environment. Instead, it can be assumed that there is a ground probability distribution p over the set of complete configurations (i.e. the space of all feasible products), indicating how likely it is that each object is the one that the current user prefers. This probability may depend on her personality, and on her current environment, but it can be assumed that the sales history gives a good approximation of that probability distribution: the configured products eventually bought by the past users are the one they prefer.

Therefore, if Next is the next variable to be assigned a value, and if u is the vector of values that have been chosen for the variables already decided, we propose to estimate, for each possible value v for Next, the marginal conditional probability $p(\mathrm{Next} = v \mid \mathrm{Assigned} = u)$: it is the marginal probability that Next has value v in the most preferred product of the current user, given the choices that she made so far; hence we can recommend the most probable value (among the admissible ones):

$$\underset{v \in \underline{\mathrm{Next}}}{\operatorname{argmax}} \; p(\mathrm{Next} = v \mid \mathrm{Assigned} = u).$$

The idea of our work is that the sale history is a sample of $\underline{\mathcal{X}}$ according to the unknown distribution p, that we can use to estimate probabilities. A first, naive method to compute $p(v \mid u)$ would be to count the proportion of v within the sold products that verify u. Even if this idea works for small u's, after a few steps the number of products that verify u would be too low and the computations would not be reliable enough (and even impossible when no product in the history verifies u). Hence the idea of learning, off-line, a Bayesian network from the data set and to use it, on-line, during the step-by-step configuration session: the user defines a partial configuration u by assigning some variables and chooses a variable Next ; the recommendation task consists in computing the marginal $p(\mathrm{Next} \mid \mathrm{Assigned} = u)$ and recommending the user with the value of Next that maximizes this probability.

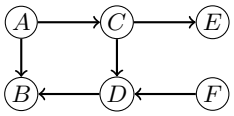
3.1 Bayesian networks

A Bayesian network (BN) [21] over set of variables $\mathcal{X}$ is defined by a directed acyclic graph (DAG) over a $\mathcal{X}$, and a set of local conditional probability tables (CPT), one for each variable of $\mathcal{X}$. If $\mathcal{N}$ denotes a Bayesian network, for $X \in \mathcal{X}$ we denote by $\mathrm{Pa}_{\mathcal{N}}(X)$ the set of parents of X in the graph; the local probability table associated to $\mathcal{X}$ specifies the probability $p_{\mathcal{N}}(X = x \mid u)$ for every $x \in \underline{\mathcal{X}}$ and every $u \in \underline{\mathrm{Pa}_{\mathcal{N}}(x)}$; if U denotes the parents of X, we denote the table associated to X by $\Theta_{\mathcal{N}}(X \mid U)$.

A Bayesian network $\mathcal{N}$ uniquely defines a probability distribution $p_{\mathcal{N}}$ over $\mathcal{X}$: the probability of a complete configuration $o \in \underline{\mathcal{X}}$ is

$$p_{\mathcal{N}}(o) = \prod_{X \in \mathcal{X}} \Theta_{\mathcal{N}}(o[X] \mid o[\mathrm{Pa}_{\mathcal{N}}(X)]) = \prod_{X \in \mathcal{X}} \Theta_{\mathcal{N}}(X, o).$$

Example 1. Consider following Bayesian network:



The probability of a configuration $abcdef$ can be computed as:

$$\Theta(a)\Theta(c \mid a)\Theta(e \mid c)\Theta(f)\Theta(d \mid cf)\Theta(b \mid ad)$$

and $\Theta(D, abdcf)$ is defined to be $\Theta(D \mid cf)$.

In the sequel, we will often omit the subscript $\mathcal{N}$ when there is no ambiguity.

3.2 Learning a Bayesian network

The learning of Bayesian networks from data proceeds in two steps: finding the structure of the network, i.e. of the DAG underlying the Bayesian network and then its parameters, i.e. the conditional probabilities table. Both aim at maximizing likelihood estimates, i.e. the probability of observing the given set.

Since learning the most probable a posteriori Bayesian network from data is an NP-hard problem [7], heuristic strategies had to be found. There are two main families of approaches in structure learning: the score-based ones and the constraint-based ones.

The formers search for a network that maximizes a score pointing out to what extend the network fits the data [8]. The score may be a Bayesian function, such as Bayesian Dirichlet scores (analysed in [12]), or come from information theory, such as the Bayesian Information Criterion [23] or the Akaike Information Criterion [2].

The latter approach looks for conditional independences, through independence tests, assuming the faithfulness of the network to learn. An early example is the Inductive Causation algorithm of Pearl [27] ; a more recent one is PC [25].

Finally, hybrid method exist, such as MMHC [26], that learns the undirected structure of the network with a constraint-based approach (named MMPC) and then orients the edge of the DAG with a score-based method. Another example is Sparse Candidate (SC) [14].

3.3 Computing marginals

The computation of the *posterior* marginal probability $p(\text{Next} \mid \text{Assigned})$ is a classical task of Bayesian inference. In general, it is broken down into computations of two separate *prior* marginals, since, by definition $p(\text{Next} \mid \text{Assigned}) = p(\text{Next} \wedge \text{Assigned})/p(\text{Assigned})$.

Recall that, for a given configuration o, $p(o)$ is defined to be the product of local, conditional probabilities that correspond to o in the CPT's of the network. Then, given a variable $X \subseteq \mathcal{X}$ and a partial configuration $x \in \underline{X}$, the marginal probability $p(x)$ is the sum of the probabilities of the complete configurations that extend x:

$$p(x) = \sum_{\substack{w \in \underline{\mathcal{X}} \\ w[X] = x}} = \sum_{\substack{w \in \underline{\mathcal{X}} \\ w[X] = x}} \prod_{Y \in \mathcal{X}} \Theta(w[Y] \mid w[Pa_{\mathcal{N}}(Y)]).$$

Computing such prior marginals is known to be an NP-hard problem when p is represented by a Bayesian network [10]– the size of the formula can grow exponentially fast with the number of variables. Exact inference algorithms, such as variable elimination [28], value elimination [5], jointree algorithms [19], cutset conditioning [20], recursive conditioning [11], work by breaking down this sum-product formula, into sub-sums and sub-products. These algorithms have a worst-case time complexity exponential with respect to the treewidth of the network. Variable elimination and jointree methods [19] are costly in space while recursive conditioning allows an any-space inference and can be polynomial in space. Even if they target a NP-hard task, the algorithms are efficient enough on real world benches to allow an on-line use.

3.4 Recommendation using Naive Bayesian Networks

In a Naive Bayesian network, one central variable (the one on which inference is to be made) is targeted and the others are assumed independent from each other conditionally to this variable of interest. A naive Bayesian network is therefore a Bayesian network the structure of which is a tree, and where the variable of interest (in our case, Next) is the parent of every other variables (in our case, the variables in Assigned). For any value v of Next and any assignment u of Assigned, we know that $P(v|u)$ is proportional to $P(vu)$; under the strong assumptions of the naive Bayesian network:

$$P(vu) = P(v) \prod_{X \in \text{Assigned}} P(u[X] \mid v)$$

So we will recommend the value v that maximizes

$$P(v|u) \propto P(v) \prod_{X \in \text{Assigned}} P(u[X] \mid v)$$

Since the variable we are recommending a value for depends on the configuration process, we would need a naive Bayesian network for every variable: to recommend a value for Next, we would use the naive Bayesian network for which Next is the variable of interest. The computation of the networks is preprocessed: (all) the prior distributions $P(X)$ and (all) the conditional tables $P(Y|X)$ (i.e., potentially all the naive Bayesian networks) are computed off line, before the configuration process, from the sample:

$$P(X = x) = \frac{\#(x)}{|\mathcal{H}|} \text{ for each } X \in \mathcal{X}$$

$$P(Y = y|X = x) = \frac{\#(x.y) + 1}{\#(x) + |\underline{Y}|} \text{ for each pair } X, Y \in \mathcal{X}$$

The (pre)computation of n prior tables and n^2 conditional probability tables are thus sufficient to make a prediction for any variable at any moment.

The strong assumptions of naive Bayesian networks is generally inconsistent: when we want to recommend a value for Next, we assume that all the variables in Assigned are conditionally independent given Next. In spite of this naive and strong assumption, they are efficient enough for some applications. Among their qualities, they are easy to learn and easily scalable, requiring a number of parameters quadratic in the number of variables.

4 k-nearest neighbor

In [9], three algorithms are proposed that are based on the selection of a neighborhood: rather than computing the preference from the entire sample, the system should focus on sold configurations that are similar to the present one - i.e. use the k nearest neighbors. All the methods proposed in [9] are based on the Hamming distance; namely, given an assignment u of Assigned, and a complete configuration w, $d(u, w)$ counts the number of variables in Assigned on which the two configurations disagree:

$$d(u, w) = |\{x \in \text{Assigned} \mid u(x] \neq w[x]\}|$$

At each step, these methods first selects the set $N(k, u)$ of the k-nearest neighbors of the current partial configuration u, and compute the recommendation on this basis.

4.1 Weighted Majority Voter

The simplest algorithm is the Weighted Majority Voter, which predicts the value of Next on the basis of a weighted majority vote of the k nearest neighbors. The weight of a configuration w in $N(k, u)$ is set equal to the degree of similarity between this configuration and the current one, u, i.e. the number of variables that are given the same value by both:

$$weight(u, w) = |\{X \in \text{Assigned} \mid u[X] = w[X]\}|$$

The recommended value for Next is chosen among the ones that are authorized by the constraints by maximizing:

$$vote(v) = \sum_{\substack{w \in N(k,u) \\ w[\text{Next}] = v}} weight(u, w)$$

4.2 Most Popular Choice

Most Popular Choice predicts the most popular (actually, the most probable) extension of the current configuration, u, from the knowledge of the closed neighbors and recommends the value supported by this configuration. It holds that, for any full configuration uw that extends u, $P(uw) = P(u|w).P(w)$. [9] make the assumption that the variables that have not been assigned are mutually independent, and that the ones that are assigned are independent from one another given w. Hence we have:

$$P(uw) = \Pi_{X \in \mathcal{X} \setminus \text{Assigned}} P(w[X]) \ . \ \Pi_{X \in \text{Assigned}} P(u[X]|w)$$

The probabilities are estimated from the k nearest neighbors of u:

- for $X \in \mathcal{X} \setminus$ Assigned and $x \in \underline{\mathcal{X}}$:

$$P(x) = \frac{1}{k}|\{w' \in N(k,u), w[X] = x\}|;$$

- for $X \in$ Assigned and $x \in \underline{\mathcal{X}}$, let $N(k,u,w)$ be the set of neighbors of u that agree with w on Assigned: $N(k,u,w) = \{w' \in N(k,u), w'[\text{Assigned}] = w[\text{Assigned}]\}$, then $P(x \mid w)$ is the fraction of $N(k,u,w)$ that has value x, with a kind of m-estimate correction since $N(k,u,w)$ may be empty:

$$P(x|w) = \frac{|\{w' \in N(k,u,w)|w'[X] = x\}| + 1}{|N(k,u,w)| + k}$$

The value recommended for variable Next is the one prescribed by the w that maximizes $P(uw)$. The drawback of this method is that nothing guarantees that the value computed is compatible with u according to the constraint.

4.3 Naive Bayes Voter

The Naive Bayes Voter is similar to the Naive Bayes method proposed in Section 3.4, with the difference that it uses the k nearest neighbors to build a naive Bayes network. Since these neighbors depends on the current configuration, is not possible to preprocess the computation of the probability table - this approach may be much slower than the classical naive Bayes.

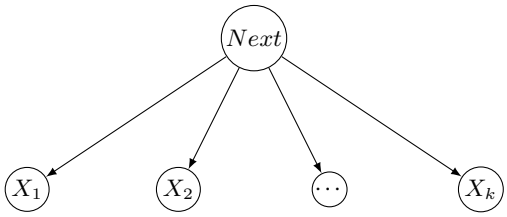


Figure 1. The naive Bayesian network built by Naïve Bayes Voter. Next is the variable of interest.

The recommended value for Next is chosen among the ones that are authorized by the constraints by maximizing $P(v|u) \propto p(v) \prod_{X \in Assigned} p(u[X] \mid v)$, where:

- $p(v) = \frac{1}{k}|\{w \in N(k,u)|w[\text{Next}] = v\}|$
- for every $X \in$ Assigned and every $v \in \underline{\text{Next}}$, let $N(k,u,v)$ be the set of neighbors of u that have value v for Next, then

$$p(u[X] \mid v) = \frac{|\{w \in N(k,u,v)|w[X] = u[X]\}| + 1}{|N(k,u,v)| + k}$$

5 Experiments

The approaches proposed in this paper have been tested on a case study of three sales histories provided by Renault, a French automobile manufacturer [5]. These data sets, named "*small*", "*medium*" and "*big*", are genuine sales histories - each of them corresponds to a configurable car, and each example in the set corresponds to a configuration of this car which has been sold:

– dataset "*small*" has 48 variables and 27088 examples.
– dataset "*medium*" has 44 variables and 14786 examples.
– dataset "*big*" has 87 variables and 17724 examples.

Most of the variables are binary, but not all of them.

We used the R package *bnlearn* to learn the Bayesian networks [24] - more precisely, we used Hill Climbing (HC) to learn the two datasets of about 50 variables (*small* and *medium*) and MMHC to learn the *big* dataset of about 90 variables. The average number of parents of a node in the obtained BN is about 1.17, 1.02 and 0.98 - for *small*, *medium* and *big*, respectively. As to Bayesian inference, we used the jointree algorithm provided by the library Jayes [18]. We implemented the Naive Bayes approach and the algorithms based on the k nearest neighbors (k is set to 20 in the experiments reported here; other values of k do not improve the results).

5.1 Experimental protocol

We used a two-folds cross-validation: each dataset has been cut by half, an algorithm learns with one half (which constitute the sale history) and is tested with the other (which can be view as a set of online configuration sessions).

The protocol is described in Algorithm 1. Each test is a simulation of a configuration session, i.e. a sequence of variable-value assignments. In real life, a genuine variable ordering was used by the user for her configuration session and the different sessions generally obey different variable orderings. Unfortunately, the histories provided by Renault describe sales histories only, i.e. sold products, and not the sequence of configuration in each session. That is why we generate a session $session$ for each product P in the test set by randomly ordering its variable-value assignments. Then, for each variable-value assignment (X, x) in this sequence, the recommender is asked for a recommendation for X, say r: r may be equal to x; or not, if r more probable than x according the inference process ; then X is set to x. We consider a recommendation as correct if the recommended value is the one of X in the product P (i.e. if $r = x$). Any other value is be considered as incorrect.

The recommendation algorithm is evaluated by (i) the time needed for computing the recommendations and (ii) its success rate, obtained by counting the number of correct and incorrect recommendations.

5.2 Oracle

In order to easily interpret the results of the cross-validation, we propose to compute the highest success rate attainable for the test set. If we where using an algorithm that already knows the testing set, it would use the probability distribution estimated from this testing set. Therefore it would recommend for the variable Next, given the assigned values u, the most probable value of X in the subset of products, in the test set, that respect u. More precisely, for any x in the domain of Next, it would estimate $p(x|u)$ as $\#(ux)/\#(u)$. Notice that $\#(u)$ is never equal to zero, since the test set contains

[5] available at `http://www.irit.fr/~Helene.Fargier/BR4CP/benches.html`

Algorithm 1: Protocol for evaluating value recommendation in interactive configuration

Input: The training set Htr and the testing set $Htest$
Output: The success rate
main :
learning of Htr
$success \leftarrow 0$
$error \leftarrow 0$
$Assigned \leftarrow \emptyset$
for each $P \in Htest$ **do**
  $session \leftarrow$ randomly order the variable-value assignments in P
  $Assigned \leftarrow \varnothing$
  for each $(\text{Next}, x) \in session$ **do**
    $r \leftarrow$ recommended value for Next given Assigned
    if $r = x$ **then** increment $success$ by 1
    else increment $error$ by 1
    $Assigned \leftarrow Assigned \cup \{(\text{Next}, x)\}$
return $success/(success + error)$

at least one product consistent with u: the one corresponding to the current session. It is an algorithm overfitted to the testing set.

We call this algorithm "Oracle". Its success rate is higher than the one of any other strategy. Its success rate isn't 100% since there is an intrinsic variability in the users (otherwise only one product would be sold ...). The success rate of the "Oracle" is generally not attainable by the other algorithms, because the "Oracle" has access to the testing set, what is obviously not the case of the algorithms we evaluate.

5.3 Results

The experiment have been made on a computer with a quad-core processor i5-3570 at 3.4Ghz, using a single core. All algorithms are written in Java, and the Java Virtual Machine used was OpenJDK.

Success rate

Figures 2, 3 and 4 give the success rate of the pure BN-based approach (BN and Naive Bayes) on the one hand, and of the methods based on k closest neighbors on the other hand, on our configuration instances. The experiment is completed with the application of the configuration protocol on classical Bayesian networks benchmarks [24] [6]. The oracle is given as an ideal line.

It appears that on the configuration instances, the pure naive based approach, which makes very strong independence assumptions, has a low success rate (this error rate is bad also on classical BN benchmarks). This is not surprising, since the variables are not independent from one another, at least because of the constraints. The independence assumptions at work in the methods based on the k closest neighbors are in a sense less drastic, since the distance used to select the neighborhood implicitly captures some dependencies.

On configuration problems, 3 methods are have very good results: Classical Bayes Net, Naive Bayes Voter and Most Popular Choice. Their success rate is very good (only a few points from the Oracle). The gap with the Oracle gets larger when the number of assigned variables increases: the Oracle's performance becomes less and less attainable. Indeed, the prediction of the Oracle relies on the testing sample, that includes the product of the ongoing configuration. When few variables are instantiated, the Oracle uses a rather big subsample to make its estimation. When a lot of variables are instantiated, the Oracle uses a small subsample, so small that sometimes it contains only the ongoing configuration. In this case, the Oracle can't make a bad recommendation. This can be interpreted as overfitting, since the Oracle is tested on the sample it learned. This phenomena is especially visible with the dataset "*big*", because it has more variables that "*small*" or "*medium*".

Classical Bayes is the more accurate method on BN instances we tested (*hailfinder*, *alarm*, *child*, *insurance*, see e.g. Figure 5 for the *insurance* bench), which is not surprising either, because the network learnt precisely captures the indepedencies (the sample is perfectly faithful to the BN). But the Naive Bayes Voter and Most Popular Choice do not perform so bad on these instances, from which it can be concluded that these approaches capture a great part of the dependencies, even not explicitly.

CPU time

The CPU time (see Figures 2, 3 and 4) clearly breaks the set of algorithms in two groups: the ones that learn, off-line, the dependencies from the entire data set and the ones that compute a new neighborhood at each step.

The former group of method are one order of magnitude quicker than the latters on the *small* and *medium* instances (and some times two: Weighted Majority voter, which has good performances in terms of prediction, is much slower). This is explained by the time needed to extract the k best neighbors before computing the recommendation. On the other hand, this time is not too sensitive to the size of the problem - it remains low on the *big* instance.

One can check that on this data set, which corresponds to a real world application, the CPU times of all the method tested are compatible with an on line use, with less than 10 ms in any case. Unsurprisingly, the approximation by a naive Bayesian net is the one that run the fastest (less than 0.05 ms in any case). The time need by Classical Bayesian Nets is in the same order of magnitude, less than 0.1 ms, for the *small* and *medium* data set. It stays under 0.25 ms for the *big* data set.

Influence of the sample's size

The drawback of the methods based on a neighborhood is that their performances seem to depend on the size of the original sample: the greater, the better the prediction but the higher the time needed to make it. To confirm this, we performed another experiment, varying the size of the sample (from the full sample to a sample containing only $\frac{1}{64}^{th}$ of the original one).

This of course leads to an improvement of the performances in terms of CPU and space, but also to a a strong degradation of the accuracy. As a matter of fact, on the *small* data set, the handling by Most Popular Choice of a sample of $\frac{1}{32}$ of the original one needs twice less time, but the error rate stay over a 9% line (instead of an average of 4%). Naive Bayes Voter and Classical Bayesian Networks are more resistant: for Naive Bayes Voter the time for handling a sample of $\frac{1}{32}$ of the original one is divided by 3, with an error rate staying over a 8% line (instead of an average of 5%).

[6] On these benchmarks, the protocol remains the same but has another interpretation: the assignment of a variable corresponds to the conditioning of the knowledge base by an observation; the "recommendation" then corresponds to the inference of the most probable value for a variable of interest

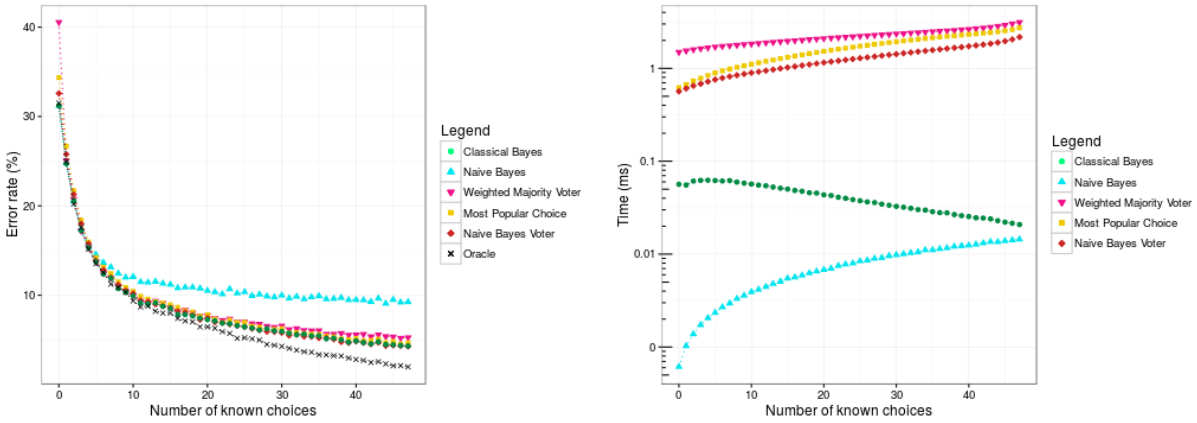


Figure 2. Average error rate and time on dataset "*small*"

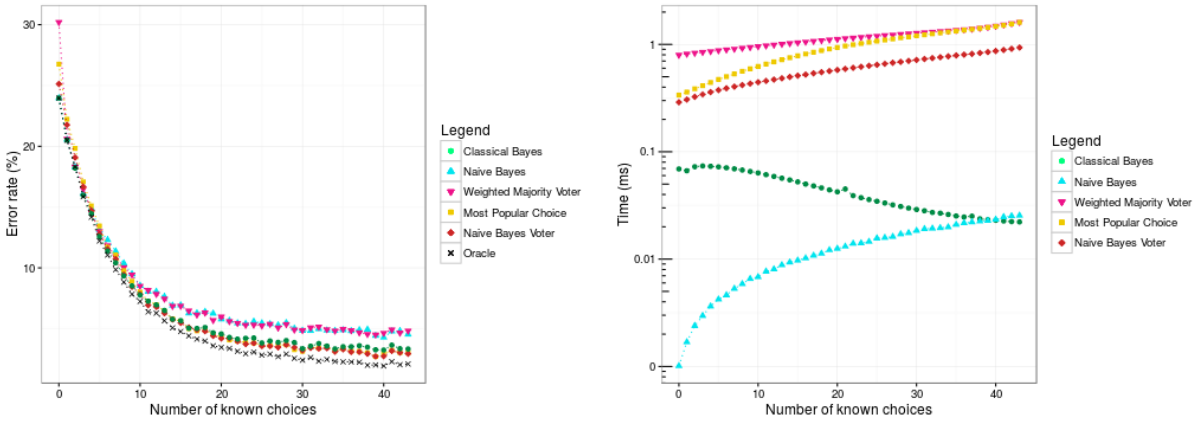


Figure 3. Average error rate and time on dataset "*medium*"

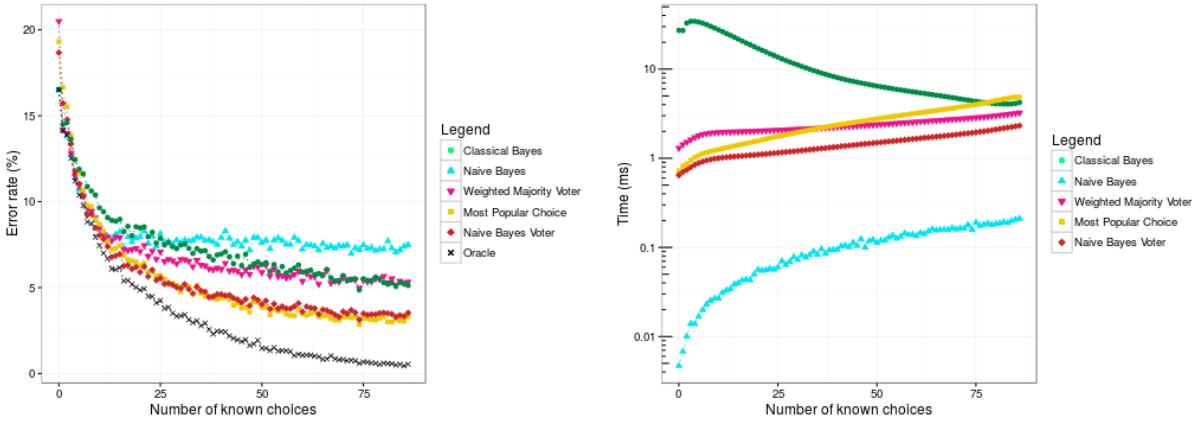


Figure 4. Average error rate and time on dataset "*big*"

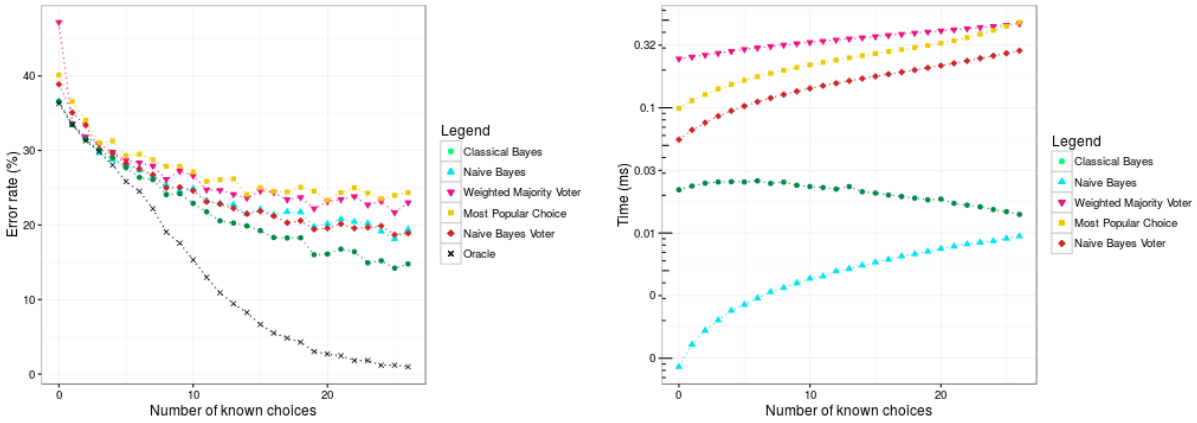


Figure 5. Average error rate and time on dataset "*insurance*"

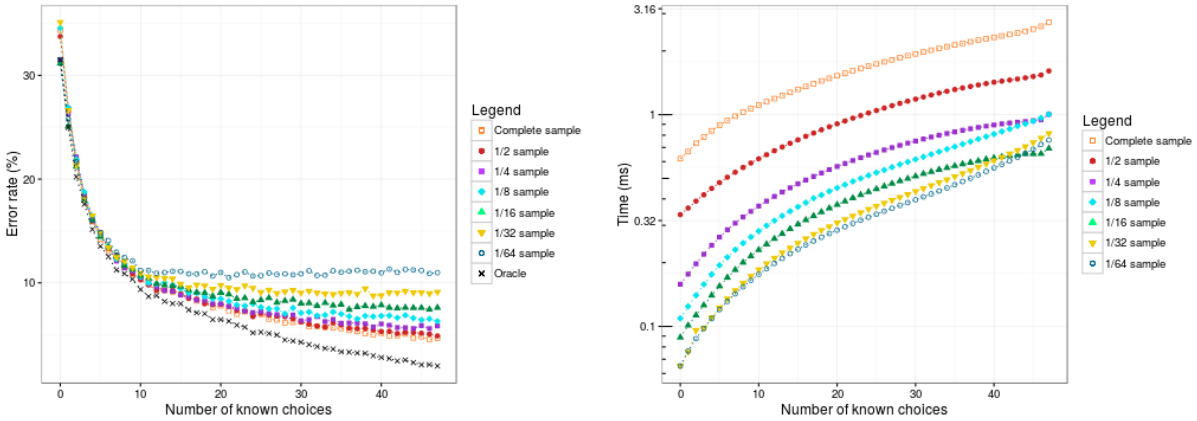


Figure 6. Average error rate and time of Most Popular Choice on dataset "*small*", varying the size of the sample

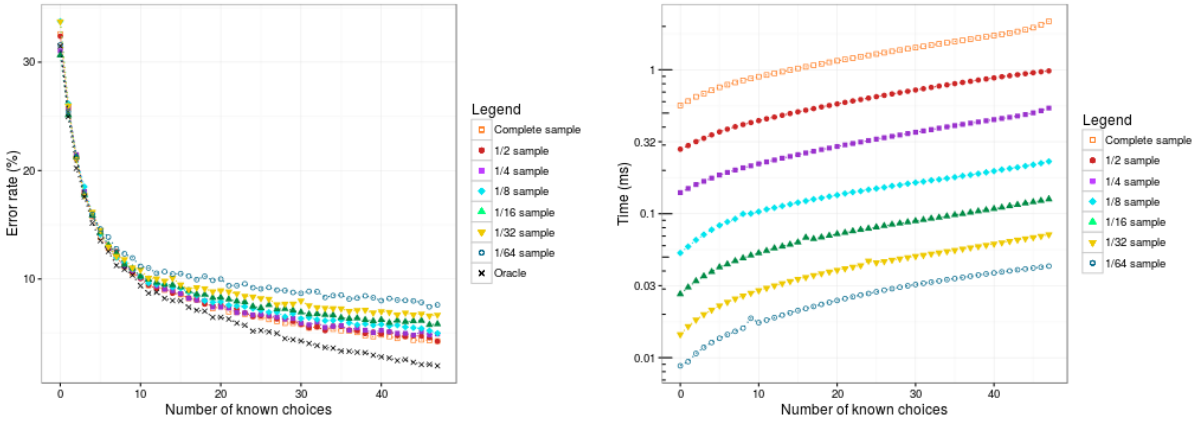


Figure 7. Average error rate and time of Naive Bayes Voter on dataset "*small*", varying the size of the sample

6 Conclusion

This paper has proposed the use of Bayesian nets as a new approach to the problem of value recommendation in interactive product configuration.

Our experiments on real world datasets show that Bayesian Nets are compatible with an on-line context. Classical Bayesian Nets have a success rate close to the best possible one. The naive Bayes approximation is average (about 10 % of error, i.e. twice the minimal error) but very quick. The other approaches proposed by the literature (Naive Bayes Voter and Most Popular Voter) have a success rate similar to the one of Classical Bayesian Nets, and a CPU time that is independent on the size of the instance (1 to 5 ms) - but strongly depend on the size of the sample. They are outperformed by Bayesian net on configuration instances on reasonable size and of course on classical Bayesian benches. We shall thus conclude in favor of the approach based on Bayesian net learning for problems with a large sample but a limited memory resource keeping in mind that naive Bayes shall be an alternative on situations involving very big instances and a very limited memory resource. When it is possible to explicitly memorize the sample, the high accuracy of methods based on a a subsample of close neighbors constitute a simple and accurate solution.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Trans. Knowl. Data Eng.*, **17**(6), 734–749, (2005).

[2] Hirotugu Akaike, 'A new look at the statistical model identification', *IEEE transactions on automatic control*, **19**(6), 716–723, (1974).

[3] Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis, 'Consistency restoration and explanations in dynamic csps application to configuration', *Artificial Intelligence*, **135**(1-2), 199–234, (2002).

[4] Jean-Marc Astesana, Laurent Cosserat, and Hélène Fargier, 'Constraint-based vehicle configuration: A case study', in *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010*, pp. 68–75, Arras, France, (2010).

[5] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi, 'Value elimination: Bayesian inference via backtracking search', in *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence (UAI'02)*, pp. 20–28, (2002).

[6] Christian Bessiere, Hélène Fargier, and Christophe Lecoutre, 'Global inverse consistency for interactive constraint satisfaction', in *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, pp. 159–174, (2013).

[7] David Maxwell Chickering, 'Learning bayesian networks is np-complete', in *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics, AISTATS*, pp. 121–130, Key West, Florida, US, (1995).

[8] Gregory F Cooper and Edward Herskovits, 'A bayesian method for the induction of probabilistic networks from data', *Machine learning*, **9**(4), 309–347, (1992).

[9] Rickard Coster, Andreas Gustavsson, Tomas Olsson, Åsa Rudström, and Asa Rudström, 'Enhancing web-based configuration with recommendations and cluster-based help', in *In Proceedings of the AH'2002 Workshop on Recommendation and Personalization in eCommerce*, pp. 30–40, (2002).

[10] Paul Dagum and Michael Luby, 'Approximating probabilistic inference in bayesian belief networks is np-hard', *Artificial Intelligence*, **60**(1), 141–153, (1993).

[11] Adnan Darwiche, 'Recursive conditioning', *Artificial Intelligence*, **126**(1-2), 5–41, (2001).

[12] Cassio Polpo de Campos and Qiang Ji, 'Properties of bayesian dirichlet scores to learn bayesian network structures', in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*, Atlanta, Georgia, USA, (2010).

[13] Andreas A. Falkner, Alexander Felfernig, and Albert Haag, 'Recommendation technologies for configurable products', *AI Magazine*, **32**(3), 99–108, (2011).

[14] Nir Friedman, Iftach Nachman, and Dana Peér, 'Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm', in *Proceedings of the Fifteenth conference on Uncertainty in Artificial Intelligence (UAI'99*, pp. 206–215. Morgan Kaufmann Publishers Inc., (1999).

[15] Tarik Hadzic and Henrik Reif Andersen, 'Interactive reconfiguration in power supply restoration', in *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pp. 767–771, (2005).

[16] Tarik Hadzic, Andrzej Wasowski, and Henrik Reif Andersen, 'Techniques for efficient interactive configuration of distribution networks', in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 100–105, (2007).

[17] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich, *Recommender Systems - An Introduction*, Cambridge University Press, 2010.

[18] Michael Kutschke, 'Jayes - bayesian network library under eclipse public license', (2013).

[19] Steffen L Lauritzen and David J Spiegelhalter, 'Local computations with probabilities on graphical structures and their application to expert systems', *Journal of the Royal Statistical Society. Series B (Methodological)*, 157–224, (1988).

[20] Judea Pearl, 'A constraint-propagation approach to probabilistic reasoning', in *UAI '85: Proceedings of the First Annual Conference on Uncertainty in Artificial Intelligence, Los Angeles, CA, USA, July 10-12, 1985*, pp. 357–370, (1985).

[21] Judea Pearl, *Probabilistic reasoning in intelligent systems - networks of plausible inference*, Morgan Kaufmann, 1989.

[22] *Recommender Systems Handbook*, eds., Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, Springer, 2011.

[23] Gideon Schwarz, 'Estimating the dimension of a model', *The Annals of Statistics*, **6**(2), 461–464, (1978).

[24] Marco Scutari, 'Learning bayesian networks with the bnlearn R package', *Journal of Statistical Software*, **35**(3), 1–22, (2010).

[25] Peter Spirtes, Clark N Glymour, and Richard Scheines, *Causation, prediction, and search*, MIT press, 2000.

[26] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis, 'The max-min hill-climbing bayesian network structure learning algorithm', *Machine Learning*, **65**(1), 31–78, (2006).

[27] Thomas Verma and Judea Pearl, 'Equivalence and synthesis of causal models', in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI '90:*, pp. 255–270, (1990).

[28] Nevin L Zhang and David Poole, 'A simple approach to bayesian network computations', in *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, (1994).

Recommending and Configuring Smart Home Installations

Gerhard Leitner[1] and **Anton Josef Fercher**[1] and **Alexander Felfernig**[2] and **Klaus Isak**[3] and **Seda Polat Erdeniz**[2] and **Arda Akcay**[2] and **Michael Jeran**[2]

Abstract. In this paper the *Casa Vecchia* smart home planning and configuration system is presented. This knowledge-based application was developed with the goal to support inhabitants of private households in the technical enhancement of their homes. The Casa Vecchia project, in the context of which the presented system was developed, was a longitudinal field study in the research area of active and assisted living (AAL). In a four years period 20 households of elderly people in the rural area of Carinthia, Austria were equipped with smart home technology and the inhabitants' experiences with the technology were researched. Results from the project with regard to needs and requirements for household smartness motivated the development of the system presented in this paper. The system is consisting of a *recommender component* which demonstrates the possibilities and benefits of smart home technology on a general level, and a *configurator component* which is able to deal with specific characteristics of living environments allowing for an individual and custom design of smart home systems.

1 Introduction

Fast technological progress has an impact on all areas of life, also in the residential sector and the *average dweller* is overwhelmed by the possibilities to enhance a home with smart technology, these are systems or components which provide an enhanced level of functionality. They can be, for example, remotely controlled, programmed, combined with other components and integrated into other systems. Today a multitude of smart devices for the home is available, but as [19] points out, a decision is not easier by default, when the number of alternatives to choose from is high. Considering the potential dangers, such as having to deal with a patchwork of incompatible subsystems, which [14] labeled the *remote control anarchy*, it is not surprising that the spread of smart technologies in the private residential sector stays behind expectations up to now. In the professional building sector, smart technology has been more successful. This is probably related to a crucial difference between the public and the private building sector. In the public and industrial building sector, initial installations, changes and enhancements of smart components are typically neither decided and planned nor installed by the users themselves. The basic infrastructures are in the responsibility of professionals, and maintained by qualified personnel, taking into consideration suitability, compatibility issues, etc. Decisions about technical enhancements of private homes are typically made by the inhabitants or they are at least involved in them. But in general, this group of people is characterized by a low level of expertise and knowledge with regard to state-of-the-art technology and by a limited willingness to invest efforts, both contributing to *bounded rationality* [21]. This problem domain therefore constitutes a promising application area for recommendation and configuration technologies [6].

In regard to the problem raised, home owners or tenants today are in a difficult situation. If they are interested in the possibilities of smart technology, they have to collect information which typically is distributed over online and offline resources. To be able to understand if such technology is applicable to their own needs and living circumstances, the existing resources are not appropriate. They are, for example, based on simulations of possible functions and features demonstrated by generic depictions of living environments. It is difficult for technical lay persons to map the presented features to their own needs. To get more precise and serious information, experts have to be consulted. The related efforts could involve inestimable costs, either in terms of financial investments, expenditure of time or both. An appropriate software tool could, on the one hand, support users in learning about the potential benefits and costs of smart home technology. This could be based on, for instance, general examples of what the technology is capable of and in this way support *preference construction* [20]. This is partly covered by existing sources. What is missing is, on the other hand, a tool that is able to demonstrate benefits and possibilities of smart technology to a user in an individualized and customized manner.

To be able to cover both aspects, an appropriate tool has to consist of two parts, whereas the general benefits of smart technology can be conveyed by *recommender technology*. To illustrate possibilities for particular living environments, *configuration technologies* can be used, which are able to deal with, for example, custom product features and connectivity issues. The approach presented in this paper is emphasising the necessity of a combined approach to support users in questions and problems related to smart technology for their homes. It is an outcome of *Casa Vecchia* [12], a research project performed in the domain of active and assisted living (AAL). *Casa Vecchia* constituted a longitudinal field study focusing on the possibilities of smart home technology in a specific field of application. Within the project it was investigated if and how smart technology can support elderly people in rural areas to manage their lives more independently and with an enhanced level of comfort. Around twenty households in the federal state of Carinthia, Austria, inhabited by elderly people in different family constellations were part of the project for the period of four years. The households were equipped with sets of smart components, the participants were observed in using them

[1] Alpen-Adria Universität Klagenfurt, Austria, email: {gerhard.leitner, antonjosef.fercher}@aau.at
[2] Graz University of Technology, Austria, email: {alexander.felfernig, spolater, aakcay, mjeran}@ist.tugraz.at
[3] SelectionArts, Austria, email: klaus.isak@selectionarts.com

and also frequently interviewed regarding their experiences. In order to equip the participants' households with appropriate smart technology, contextual inquiries [2] were conducted and numerous planning and design meetings were carried out. The involved efforts, the planning, design and installation of the customized smart systems could only be realized in a small number of locations. This led to the idea to automate and computerize the process to have the possibility to address broader shares of prospective users in the future. The result was an initial version of the *Casa Vecchia* home planning and configuration system developed with the goal to support the systematic planning of a smart home system for private households, considering the individual requirements of inhabitants as well as the infrastructural characteristics and constraints of their respective living environment.

The remainder of this paper is organized as follows: In Section 2 we discuss work related to the application of *intelligent systems* in the context of smart homes. In Section 3.1 we introduce basic functionalities of the recommender part of the system and also present examples of the corresponding user interface. Thereafter, in Section 3.2 we provide a detailed insight into our smart home configuration tool. With Section 4 the paper is concluded.

2 Related Works on Intelligent Systems in Smart Homes

Decision support systems or recommendation technologies are already used in a variety of contexts. Different approaches are the basis of concepts such as collaborative filtering [18], content-based filtering [15] and knowledge-based recommender systems [4]. However, only a few research works address recommendation technology in the context of smart homes, for example [10] propose the usage of collaborative filtering in (professional) building automation. Knowledge-based approaches, cf. e.g. [8], have been used to support users in smart homes by recommending actions based on historical activity data, [13] illustrate the possibilities of recommender technologies to manage digital contents and services. The applicability of a specific type of recommender technology, however, depends on the problem to be solved. A hybrid approach to address this aspect was proposed by [11], which is based on the combination of different recommendation technologies that can be individually applied depending on the problem at hand. For example, advises for saving energy can be given on the basis of collaborative filtering whereas critical incidents (such as the detection of smoke or fire) are resolved with knowledge-based methods. Summarizing, related work has a strong focus on the *application* of recommendation technology as enhancement of smart home systems that are already available to their users. The problem domain addressed in this paper is the phase of *planning and designing* such a system.

Collaborative and content-based filtering approaches are not applicable to this domain due to the fact that the required rating data are not available in an appropriate granularity. Typically, people do not install smart home equipment very frequently. For this reason a knowledge-based approach was chosen which calculates recommendations on the basis of a predefined set of recommendation rules (*constraints*) rather than on the basis of rating information. One of the challenges is the variety of smart home systems and components, which has to be considered in the development of such knowledge base. This challenge could be addressed by the identification of commonalities of smart systems and components available on the market, for example, in regard to the needs the systems are covering. A basic taxonomy was created by forming categories on the basis of such needs. Related work to build upon has been done, for example, by [16]. The classes of needs the authors differentiate are entertainment, surveillance and access control, energy management, home automation, assistive computing, and health care. In the work of [9] comfort, autonomy enhancement, and emergency assistance are differentiated and [1] distinguish between the quality of living, reducing costs, or providing services for health care. Based on the related work the following categories of needs were seen to be relevant in regard to smart technology:

- *Controllability*: Remote control, combined switching of devices to support certain scenarios, e.g. watching TV (close blinds, dim lights, switch on TV).
- *Cost saving*: Reduction of energy consumption by identifying devices currently not in use. Automatic control of devices based on time parameters or sensor data (e.g., no activity recognized for 10 minutes $\rightarrow$ switch off lights in the respective room).
- *Health support*: Controlling devices which are hard to reach, specifically of interest for people with movement restrictions. Remote health status monitoring by the observation and analysis of activity data.
- *Improving Safety/Security*: Access control by auto-lock mechanisms. Automatic switch-off of potentially dangerous devices (e.g., electric stove, iron). Alerting functions when inhabitants are not at home but activity is recognized.

The second dimension used for forming categories is based on the characteristics of components smart home systems are consisting of. Although providing a high variety of functions and being based on different technical features (e.g. connection via radio, bus or power line) and form factors, the components can be condensed into basic categories based on their features. The categories presented in the following are based on a scheme proposed by [7]:

- *Sensors*: Measuring data or status in the environment they are installed in, e.g. motion sensors.
- *Actuators*: Triggering events on the environment they are installed in, e.g. remote controls.
- *Input Devices*: Providing the possibility to interact with the system on a higher level, e.g. desktop computers, tablets, smart phones.
- *Output Devices*: Enabling the observation of the system's status and the notification of users, for example, embedded computers or environmental displays.
- *Gateway Components*: Building a central point of communication with and between other devices and offering the possibility of parameterization, configuration, and programming.

The features of the two categorization schemes are included in the knowledge base [4] and cross-linked. For example, if a user is interested in enhancing security (need category: Security), this can be managed by observing corresponding devices.

3 Overview of the Smart Home Installation System

The *Casa Vecchia* smart home planning and design system consists of a recommender and a configurator part. The user is guided through the problem domain, whereas the first (*recommender*) part is focused on informing the user about the general possibilities of smart home technology and the elicitation of preferences. Based on the preselections made in these initial steps, the *configurator* enables users to customize a smart home system for their individual living circumstances and needs.

3.1 Casa Vecchia Recommender

After a welcome screen explaining the goal and introducing the upcoming dialogue, the users have to select smart features which are of highest interest for them (see Figure 1). The aspects the recommender part of the system deals with are related to the categories described above and consisting of the following elements:

- *Major interests/goals*: Security, energy, control
- *Technical Requirements*: Stability, emission, ease of installation, installation costs, maintenance efforts.
- *Building characteristics*: Apartment or detached family house, one or more floors.
- *Scope of planned efforts*: Initial installation in a new building, comprehensive renovation, partial modernization, do-it-yourself enhancement.

The recommender takes into account the elements, rules, and constraints in the knowledge base and assures that only information appropriate to the context is presented. The initial dialogue is - depending on the selections made in the previous steps - consisting of 5 steps on average. The primary goal of this stage is to point out potentials of smart home technology in general, to elicit needs and to support the *construction of preferences* in the sense of [20].

The selections made by the users influence the procedures in the back end of the system and the components recommended at the end of the process. For example, if the user states that his living environment only has one floor, stair elements are not shown in the configuration phase.

In this phase the selection of criteria is not limited, potential conflicts are pointed out but not corrected. This is because the user can always change settings during the dialogue. If conflicts persist until the end of the dialogue, they are explained and resolved.

3.2 Casa Vecchia Configurator

After having completed the recommendation part the user is guided to the configuration part of the system. The transition is visually emphasized by a change from a text-based to a graphical interface, the latter enabling the users to sketch the floor plan of their living environments with drag&drop (see Figure 2). The users can use simplified design elements to sketch a variety of rectangular floor plans. In this way rooms, hallways and stairs can be sketched and doors or windows can be positioned. After having finished sketching the floor plan with the basic elements and having labeled the rooms, the next step is to position devices that are currently present in the user's home (3. This constitutes an important advantage of our approach. It is not necessary for the user to identify whether a smart component is available or appropriate for his or her purposes, but the configurator automatically identifies appropriate smart components on the basis of the user's preferences (energy saving, safety, etc.), the floor plan and the devices the user has positioned. This functionality is based on rules implemented in the knowledge base. Examples for such rules are:

```
On Building Level:
Electric smog is an issue (=yes) ⇒
smart-home-system-type (=wired)
Low price relevant (=yes) ⇒ smart-home-system-type
(=wireless)
if Electric smog is an issue (= yes) and Low price
relevant (= yes) ⟹ conflict
```

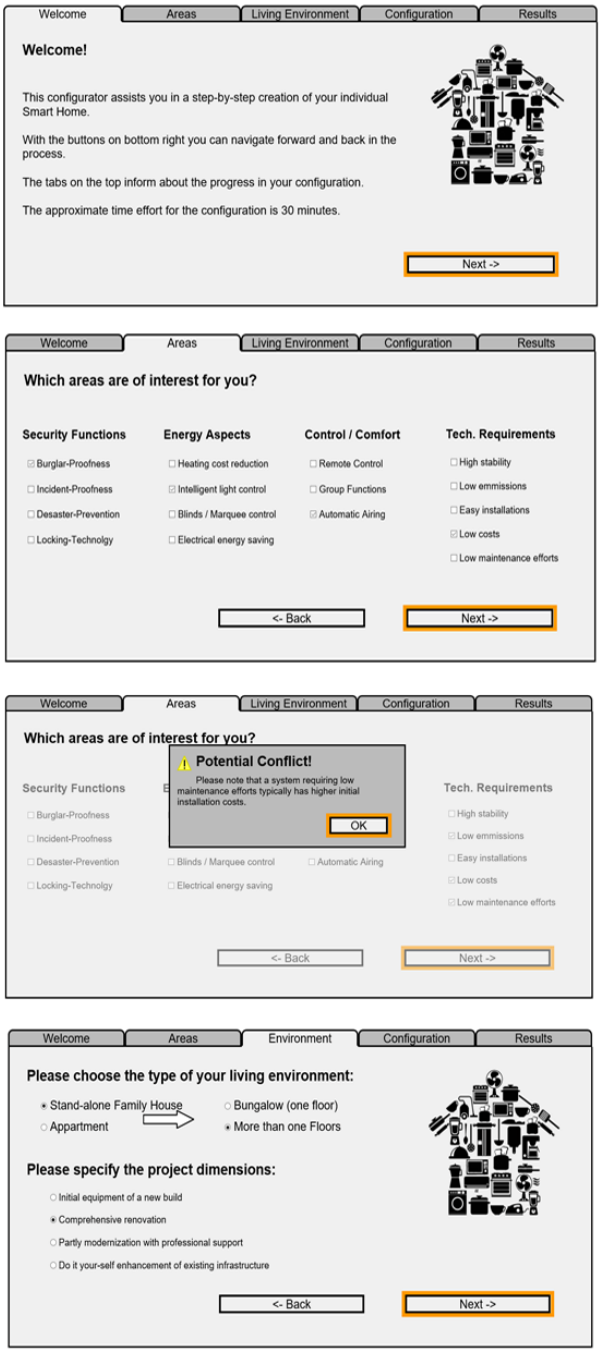


Figure 1. Example screens of the recommender part of the system, illustrating the criteria that can be selected by the user. The pop-up depicted in the third screen shows an example of a potential conflict.

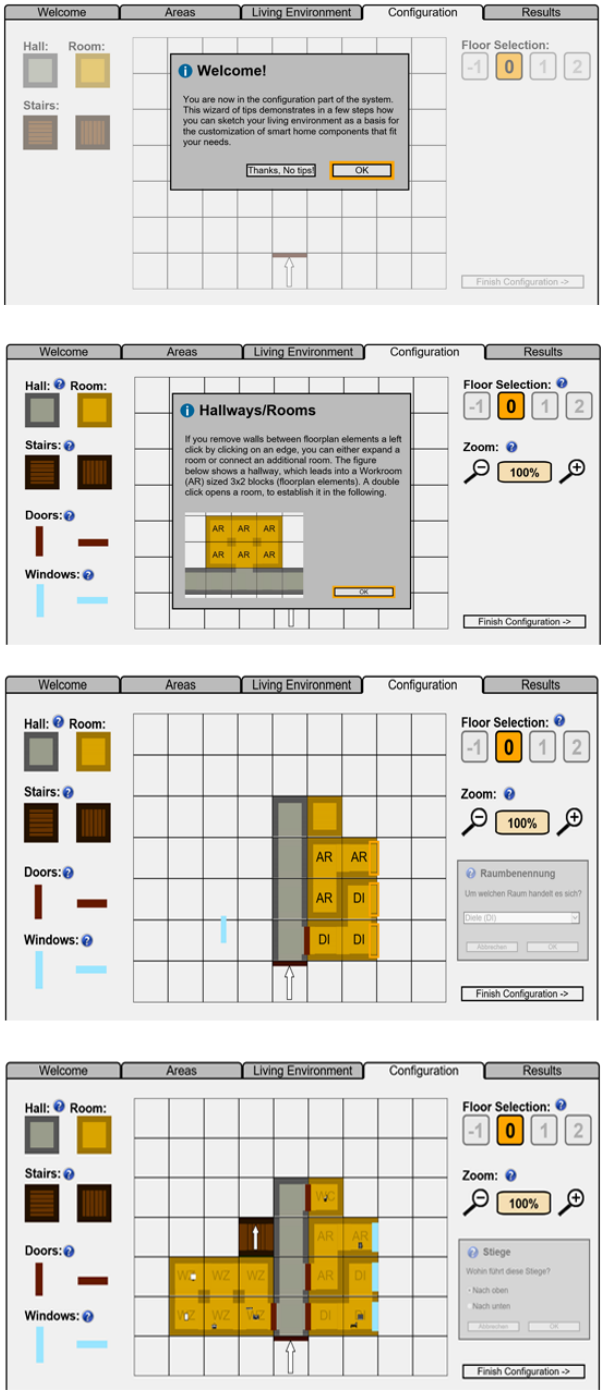


Figure 2. Example screens of the configurator interface. In the top part of the figure, the first screen of a user tutorial is shown. In the tutorial the essential steps of the configurator are demonstrated. The second screen shows a help pop-up which can be accessed via the question mark symbol positioned next to each element. Screens three and four on the bottom show different stages of floor plan design.

```
On Room Level (e.g. Kitchen):
Goal.Security (=yes) and Electric stove (=yes) ⇒
stove-sensor (=yes) and
stove-actuator (=yes)and kitchen-smoke-detector (=yes)
Goal.Support (=yes) and Fridge (=yes) ⇒
fridge-door-contact (=yes)
Goal.Comfort (=yes) and Automated Lights (=yes) ⇒
lights-actuator (=yes) and motion-sensor (=yes)
```

In the phase of configuration potential conflicts are identified and advises to resolve them [5] are given. As an example, the requirement of high stability leads to the recommendation of a wired system whereas requiring a low price would result in the recommendation of a wireless system. In this case the user is informed about the conflict as well as possibilities to resolve it. Other possible conflicts / constraints that can occur are, for example:

Conflict 1: *Remote control AND saving energy*
Remote control requires the system running 24/7 which contradicts the need of energy saving. This can be resolved in different ways, either totally (remote control or energy saving) or partly (permanent operation of specific components only, e.g. heating).

Conflict 2: *No electric smog AND low price*
Low emission can only be ensured with a wired system. This is more expensive than the wireless alternative and causes higher installation costs. This could be resolved by either accepting a higher emission or higher costs.

Conflict 3: *Remote support AND privacy/security concerns*
Support from outside can be provided only if the system is allowed to distribute data. If privacy is important, this form of support might be problematic. The conflict can be resolved, for example, by emphasizing that transferred data is encrypted and only available to a predefined group of persons.

4 Conclusions and Future Work

In this paper, the *Casa Vecchia* smart home planning and configuration system was presented which constitutes a combination of recommendation and configuration technologies. The target user group is private home owners or tenants who could, due to missing domain knowledge, benefit from a system supporting profound decisions related to the technological enhancement of the home. Such a system has to provide both, an adequate knowledge base which is able to match user needs to the functional range of smart home components and the possibility of customizing smartness to an individual living environment. The prototype system presented in this paper was initially developed by [17] and is implemented in HTML5 and other state-of-the-art web technologies and can therefore be used on conventional computers as well as on tablets and smart phones.

Another difference to first implementations is that the new version is rather based on graphical interaction than on textual dialogues. An outcome from the evaluations of the first prototype has been, that questions regarding the numbers and positions of devices present in a household, such as TVs, are more difficult to answer in a textual manner than by positioning them on a sketch of a floor plan. The graphical representation enabling drag & drop significantly increases the acceptance, usability, and convenience of the system.

The presented recommender/configurator combination has many advantages. Beside end users, other stakeholders could also benefit from such an approach, for example, service providers. The results generated by the system represent a structured and more precise description of user needs which could lead to lower costs for the prepa-

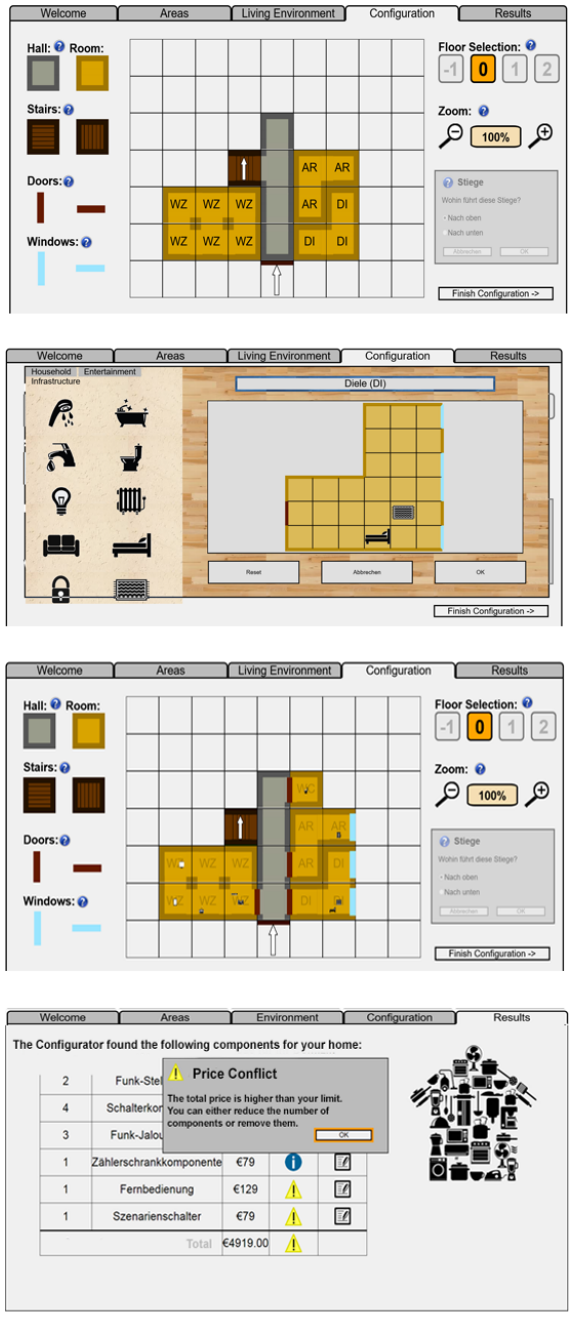


Figure 3. Example screens of the drag & drop interface of the configurator in the top part of the figure. On the top a final floor plan sketch is shown which contains rooms, a hallway, and stairs. By double-clicking on the respective room shape (in this case the "l-shaped" room located bottom right) a detail view of the room is opened which enables the positioning of devices. The user can choose different electric devices as well as installation components (e.g. faucets) and furniture and drag them on the room shape. When the user has finished the selection, he gets back to the floor plan overview and the devices just positioned in the respective room are shown as miniature symbols. The bottom figure shows the result page with the list of smart components the system finally recommends, with the possibility to edit and change in case of existing conflicts.

ration and adaption of offers, more cost efficient installations due to clearer requirements (in the form of a floorplan), and a reduction of errors in the planning as well as in the installation phase. On the side of the customer, easier preference elicitation and a better understanding of the system and its components can be expected. A detailed empirical evaluation of the presented smart home planning environment is the central focus of future work.

REFERENCES

[1] A. Aztiria, A. Izaguirre, R. Basagoiti, J. Augusto, D. Cook. Automatic Modeling of Frequent User Behaviours. In: Intelligent Environments. In: *Proc. of IE*, pp 7–12, 2010.

[2] H. Beyer and K. Holtzblatt. Contextual Design. *Interactions*, 6(1), pp. 32–42, 1999.

[3] R. Burke. Hybrid web recommender systems,In: *The adaptive web*, pp 377–408. 2007.

[4] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An Integrated Environment for the Development of Knowledge-based Recommender Applications, Int. Journal of Electronic Commerce, 11(2), pp 11–34, 2006.

[5] A. Felfernig, M. Schubert, C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets, Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM), 26(1):53-62, 2012.

[6] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. Knowledge-based Configuration - From Research to Business Cases, Elsevier/Morgan Kaufmann, 1st ed., 2014.

[7] M. Hitz, G. Leitner, H. Groß(Hsg.): Das Haus als Gegenstand interdisziplinärer Forschung. Profil Verlag, 2012.

[8] N. Kushwaha, M. Kim, D. Y. Kim, and W.-D. Cho. An intelligent agent for ubiquitous computing environments: Smart home ut-agent. In: *Proc. of SEUS*, pp 157–159, 2004.

[9] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, P. Müller. Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces, LNCS 4555, pp 103-112, 2007.

[10] M. LeMay, J. J. Haas, and C. A. Gunter, Collaborative Recommender Systems for Building Automation, In: *System Sciences, 2009. HICSS 09.*, pp 1–10, 2009.

[11] G. Leitner, F. Ferrara, A. Felfernig, C. Tasso. Decision Support in the Smart Home. in: *Decisions@RecSys'11*, Chicago, IL, 2011.

[12] G. Leitner, A. Felfernig, A. Fercher, M. Hitz. Disseminating Ambient Assisted Living in the Rural Area, Sensors Journal, 14(8):13496-13531, 2014.

[13] S. Mennicken, J. Vermeulen, and E.M. Huang, From Todays Augmented Houses to Tomorrows Smart Homes: New Directions for Home Automation Research, Proc. 2014 ACM Intl Joint Conf. Pervasive and Ubiquitous Computing, 2014, pp. 105115.

[14] J. Nielsen. Remote control anarchy. https://www.nngroup.com/articles/remote-control-anarchy/

[15] M. J. Pazzani, D. Billsus. Chapter Content-based recommendation systems, In: *The adaptive web*. pp 325–341, 2007.

[16] T. Perumal, A.R. Ramli, C. Y. Leong, S. Mansor, and K. Samsudin. Interoperability for Smart Home Environment Using Web Services, Int. Journal of Smart Home, vol. 2, no. 4, Oct., pp 1–16, 2008.

[17] M.Pum. Configurator-Umgebung fr die Unterstützung der Erstellung individualisierter Smarthome Systeme. Diploma Thesis, Alpen-Adria Universität Klagenfurt.

[18] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Chapter Collaborative Filtering recommender systems, In: *The adaptive web*, pp 291–324. 2007.

[19] B. Schwartz. The Paradox of Choice: Why More Is Less. Ecco, 2004.

[20] P. Slovic. The Construction of Preference. American Psychologist, 50(5) pp 364–371. 1995.

[21] T. Wittmann, R. Morrison, J. Richter, T. Bruckner: A Bounded Rationality Model of Private Energy Investment Decisions, in: *Proc. of the 29th IAEE*, Potsdam, 2006.

Concurrent configuration of product and process : moving towards ETO and dealing with uncertainties.

A. Sylla[1,2], **E. Vareilles**[1], **M.Aldanondo**[1], **T. Coudert**[2], **L. Geneste**[2], **P. Pitiot**[1,3]

Abstract - *Product configuration is a well-known technique that allows safe and reliable customization of products in assembly to order (ATO) or make to order (MTO) industrial situations. When dealing with engineer to order (ETO) situations, the required design activities cannot be handled by conventional configuration techniques. The first goal of this paper is to show how constraint based configuration techniques can be extended towards ETO situations for both product or system and their realization process. As ETO situations requires some design activity, the confidence in the configured item or offer proposed to the customer is lower compared with ATO-MTO situations. The second goal of this paper is to propose a set of indicators that characterize the confidence of the supplier in the configured system and process and therefore in the offer provided to the customer.*

Keywords – *Configuration, ATO-MTO-ETO, Constraint satisfaction problem, Confidence, TRL, SRL*

1. INTRODUCTION

The proposed paper concerns the assistance of a supplier in a customer/supplier relationship. More accurately, it aims at aiding the definition of a commercial offer for both system (product, system or service) and realization process. The presented contribution belongs to the stream of works that deals with the set-up of knowledge-based tools aiding the system-process definition (that can include some design activities) and supporting the quotation of performance, cost and cycle time [1].

In this offer definition context, the system-process definition can vary from a very routine activity up to a highly creative and so far much less routine one [2]. For example let us consider a computer system or a truck, the definition of an offer consists mainly in selecting some options and components in a catalogue, checking their consistency and computing a cost and a standard delivery time. At the opposite, the definition of an offer for a crane or for a specific machine-tool can require significant engineering or creative design activities for both system solution and realization process. Given these elements, the customer/supplier relationship can be characterized, according to [3], as:

- very routine assembly-to-order (ATO) or make-to-order (MTO) offer definition,
- much less routine engineer-to-order (ETO) offer definition.

For 20 years now, configuration software have been recognized as very efficient tools for aiding suppliers in their offer definition activity in ATO-MTO situations [4]. When dealing with ETO, it is less the case because the design activity is more consequent and thus Computer Aided Design software must be used. It is important to note that ATO-MTO or ETO is not a binary issue. A system composed of three sub-systems can have two of them in ATO-MTO and one of them in ETO. For instance, a crane can have its engines in ATO-MTO while its structure is in ETO.

In an ATO-MTO situation, all design problems for both system solution and realization process have already been studied and solved in advance before launching the activity of the offer definition. Therefore, the level of uncertainty in the offer characteristics is rather low and the supplier feels very confident in the fact that the defined offer matches the customer's expectations (including price and due date). When the situation begins to move from ATO-MTO towards ETO, design or engineering activities are more significant. Two kinds of approaches can be seen in companies for the offer definition activity.

- The first one relies on a detailed design of offers for both system solutions and realization processes. Thus uncertainties are low and supplier's confidence is high but this approach is time and resources consuming.
- On the opposite, the second one tends to just clarify the main ideas or concepts about offers avoiding detailed design, but leaving a great deal of uncertainty and a scant confidence.

Given these elements, the goal of this paper is to propose a theoretical approach and a knowledge-based tool aiding suppliers to define promising offers:

- for "rather" routine design situation in order to be able to collect knowledge,
- for situation "between" ATO-MTO and ETO, when more than 50% of system sub-assemblies and process tasks are entirely defined,
- that avoids the entire detailed design of offers by saving time and resources' commitment,
- and strengthens the confidence in the main ideas or concepts about system-process offers.

[1] University Toulouse – Mines Albi, France, emails: first-name.family-name@mines-albi.fr
[2] University Toulouse – ENI Tarbes France, emails: first-name.family-name@enit.fr first-
[3] 3IL – CCI Aveyron, France, , email : p.pitiot@cci-aveyron.fr

Our main and original contribution is to add a new characteristic or indicator to system-process offers that can quantify a kind of "confidence level" (in a similar sense as the one proposed by [5]). This means that each sub-assembly, each realization process activity and resulting system-process is characterized with its own "confidence level". This new indicator allows the supplier to compare competing solutions on: performance, cost, lead time but also, and we have never seen that in the scientific literature, confidence. The suppliers feel now more self-confident to decide about the offer to propose to the customer whatever the stage of its development.

The remaining of the paper is organized in three sections. In a second section, the main ideas about concurrent configuration of system and process for ATO-MTO and ETO situations are recalled and the support provided by the Constraint Satisfaction Problem (CSP) framework is explained. The third section is dedicated to the proposition of the "confidence level" indicator with various aggregation mechanisms for both system solutions and realization processes. A crane example runs all along the paper.

2. CONFIGURATION IN ATO-MTO AND ETO SITUATIONS

In a first sub section, we draw the parallel between the product configuration [4] and systems configuration and we extend ATO-MTO situations towards ETO situation.

2.1 – Configuration in ATO-MTO: Products to Systems

When dealing with concurrent configuration of product and process problem, [6] or [7] have shown that the product can be considered as a set of components and its production process as a set of production operations (dotted lines in Fig 1). According to the customer's expectations, the configuration of a product is achieved either by selecting components in product families (as an engine in a catalogue) or choosing values of descriptive attributes (power and weight of an engine) represented with dotted lines in left part in Fig 1. Of course all combinations of components and attribute values are not allowed (a low power engine is incompatible with a high crane). Thus, as explained by many authors [8] or [9] the product configuration problem can be considered as a discrete constraint satisfaction problem (CSP), where a variable is a product family or a descriptive attribute and constraints (solid line in Fig 1) specify acceptable or forbidden combinations of components and attribute values. Some kind of product performance indicators can characterize the product, thanks to some mixed constraints (symbolic and numerical domains) that link the most important product characters (for example : crane performance function of crane height and acceptable load).

For process configuration, a similar approach is proposed by [10] or [11]. According to the configured product characteristics (selected components and attributes values), the resources for each production operation can be selected in families of resources (small assembly-table for a small crane in a list of assembly tables), and in some case a quantity of resource can be specified too (2 operators for a large crane, 1 for a small one). Of course, selected components and values (for products) and selected resources and quantities (for operations) impact operation durations and therefore the production process delivery time or cycle time of the configured product. As for product, process configuration can be considered as a CSP, where each operation gathers variables corresponding to resource families, resource quantities and operation duration. Constraints (solid line in Fig 1) restrict possible associations.

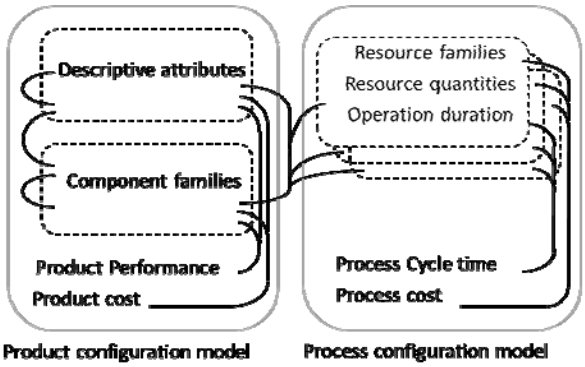


Figure 1 - Concurrent configuration of product and process

For both product and process, all variables can be linked to cost indicators (one for product and one for process). With the previous problem descriptions, [11] suggested (i) to gather these two problems into a single concurrent problem and (ii) to consider this concurrent problem as a CSP. Considering this problem as a CSP, allows the use of propagation or constraint filtering mechanisms as an aiding tool. Each time a customer's expectation is inputted (mainly in the product and less in the process), constraints propagate this decision and prune variables values for all problem variables. For a detailed presentation with an easy to understand example, we deeply suggest to consult [11].

This kind of problem modeling is the ground basis of configuration problems. All commercial websites and conventional configuration software that run interactive configuration or customization processes rely on such problem models. The key point is that all possible solutions have been studied in advance, the configuration process is infinitely routine and there is absolutely no design or creative task. Thus the supplier is fully confident in his/her ability to achieve his/her commitments, with no unnecessary stress.

Moving from products to systems is not a big deal. We assume for systems: (i) a system is a set of sub-systems (ii) a sub-system is represented by a set of descriptive attributes and one family of technical solutions (equivalent to a component family). For processes, the model is absolutely the same. Same indicators, performance, cycle time and cost are kept. From now, we will speak only of configuration of systems (and not only products) and processes.

2.2 – Configuration: from ATO-MTO to ETO

Our goal is to update the previous problem and solution in order to handle ETO system and process configuration. Moving from ATO-MTO to ETO means that some engineering activities either to design new sub-systems or to finalize the design are necessary in order to satisfy the customer's requirements.

For the system side, moving from ATO-MTO to ETO means that the system has never been designed completely because: (1) at least, one of its sub-systems has to be designed (Value-new) in order to answer to the customer's requirements, leading to a new system or (2) the system is composed of a set of existing sub-systems which have never been assembled together, leading to a new system. Let us focus only on the first point, as the second one works exactly the same but for the system level.

Let go back to the crane example. In this example, only the sub-system jib moves from ATO-MTO to ETO. Assume that until now only four jib technical solutions (Jts-1, Jts-2, Jts-3, Jts-4) corresponding with two lengths (4 and 8 meters) and two load capacities (low-load, high-load) have been already designed, manufactured, integrated in a crane and supplied to a customer (upper part of Fig 2). If the supplier wants to satisfy a customer that requires a high-load crane with a jib length different from 4 and 8, the configuration model should be updated with a new possible value for the descriptive attributes jib-length (Value-new) and a new technical solution (Jts-new) in the family of jib technical solutions (lower part of Fig 2). In the two models in Fig 2, the solid lines represent allowed combinations of descriptive attributes and technical solutions.

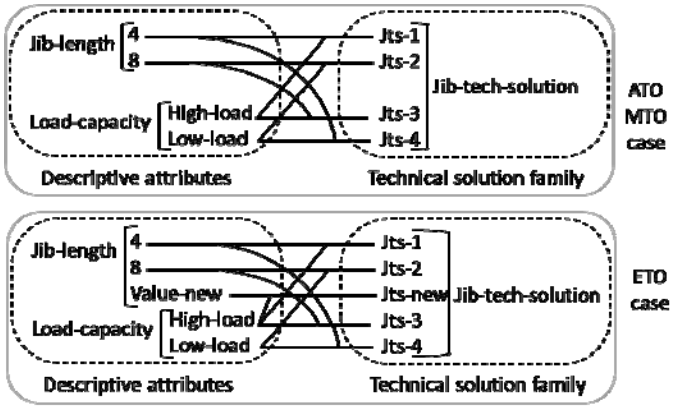


Figure 2 – Sub-System configuration models

For the realization process side, moving from ATO-MTO to ETO means that some engineering activities have to be carried out in order to design or finalize the design of the system therefore: (1) new engineering activities can be added to the realization process and tuned or (2) the process durations (design and production activities) can be updated (Value-new) to take into account the engineering activity. Let us consider the crane example. On the left side of Fig 3, a conventional ATO-MTO process model gathering two processing tasks (noted Sourcing and Production). Each task is characterized by a resource family and duration, resource quantities are unary. A very simple constraint modulates duration according to the selected resource. On the right side of Fig. 3, an ETO model with a new engineering task (noted Engin.). The existence of this task is triggered according to the selection of a new technical either for a system or a sub-system. Any operation can have: (i) its duration, (ii) its resource, (iii) its resource quantity, updated by the user according to his/her knowledge about the creative level of the new technical solution. Fig 4 just show duration possible update (with "value-new"), f or simplicity, constraints (i) between system and process (ii) computing process cost and (iii) computing process lead time are not shown.

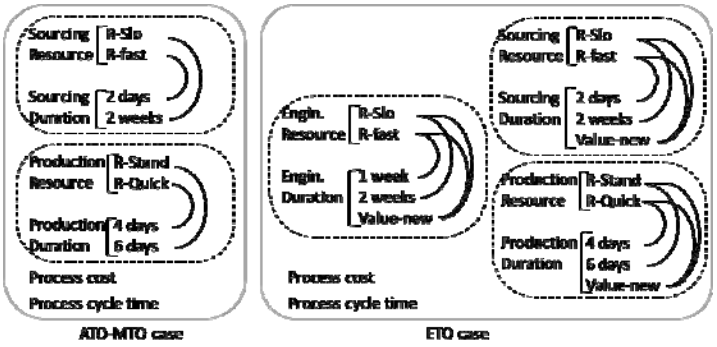


Figure 3 – Process configuration models

2.3 – Conclusion

We have shown how the problem of concurrent configuration of systems and processes can be extended from ATO-MTO towards ETO situations with respect to the constraint satisfaction framework. This extension leads to the generic model of Fig 4 that shows the different kinds of variables and their valuation domains. This allows mixing ATO-MTO and ETO sub-systems with relevant production process in a conventional interactive constraint based aiding tool. Thus, this can be used to assist the offer definition in a customer/supplier relationship. Next section deals with confidence issues of offer definitions in ETO situations.

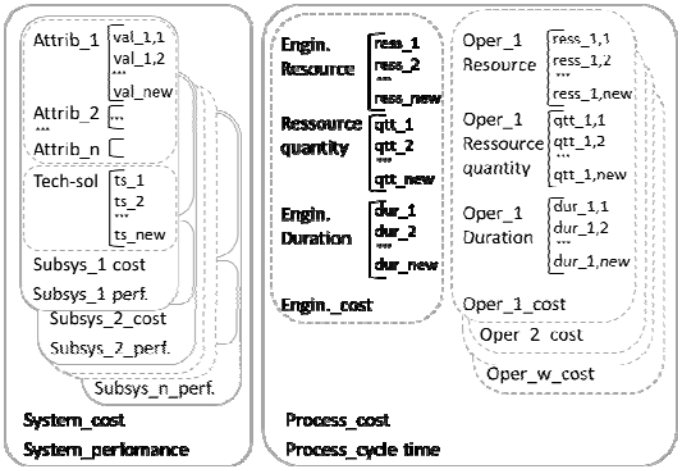


Figure 4 – ETO generic configuration model

3. OVERALL CONFIDENCE DEFINITION

The third section is dedicated to the definition of the offer overall confidence indicator. We propose that this new and original indicator relies on two pair of specific indicators, one pair characterizing the system solution, and the other one, the realization process. Each pair of indicators is composed of one objective indicator and its pre-defined scale whereas the second one is much more subjective and supplier-dependent. First, objective indicators are presented for the system and process sides, then, are the subjective ones. This section finishes with the first aggregation mechanisms in order to compute the offer overall confidence, and how this information can help supplier in decision making.

3.1 – Objective Indicators for Solution and Process

Objective indicators give reliable unbiased information on system solutions and realization processes and characterize the readiness of technology used for the system solution and the risks handling for the realization process. We propose to add to each sub-system of the system solution and each activity of the realization process, these new objective indicators.

Let's start with the system side. The offer overall confidence relies at least partially on the readiness of technology used in the system solution. Indeed, the technology readiness level or TRL indicates how much a system is ready to be deployed. TRL is a systematic metric/measurement developed by [12] (at US NASA) for the measure of the maturity of technologies. TRL is based on a scale from 1 to 9 with 9 being the most mature, as shown in Table 1. In our proposal, for each sub-system, we associate to each technical solution (of its family of technical solutions) a TRL. Therefore, selecting a technical solution for a sub-system leads to the identification of the correct TRL.

Table 1. TRL Scale

Level	TRL
9	Actual system proven through successful mission operations
8	Actual system completed qualified through test demonstration
7	System/sub prototype demonstration in a relevant environment
6	System/sub model demonstration in a relevant environment
5	Component and/or breadboard validation in relevant environment
4	Component or breadboard validation in laboratory environment
3	Analytical and experimental critical function and/or characteristic proof of concept
2	Technology concept and/or application formulated
1	Basic principles observed and reported

Let's now move to the process side. The offer overall confidence relies also on the risks taken by the supplier in case of success, meaning that he/she has won the tender. Indeed, every business is exposed to risks all the time and such risks can directly affect day-to-day operations, decrease revenue or increase expenses. Their impact may be serious enough for the business to fail. As far as we know, there is no way to characterize the risk handling level for each activity of a realization process. Therefore, based on the CCMI (Software Engineering Institute) and TRL, we propose a first version of ARL, for Activity Risks Level, based on a nine-level scale. This nine-level scale is dedicated to the main risk of an activity and relies on (i) risk probability of occurrence (high or low), (ii) risk impacts (serious or marginal) and (iii) risk treatments (existence of action plans or not). Table 2 presents the nine levels of ARL. In our proposal, for each activity, we associate an ARL. Depending on the model and knowledge, ARL can be modified by the selection of adequate resources and valuation of their quantity.

Table 2. ARL Scale

Level	ARL
9	Risk with low probability, marginal impact and treatments
8	Risk with high probability, marginal impact and treatments
7	Risk with low probability, serious impact and treatments
6	Risk with high probability, serious impact and treatments
5	Risk with low probability, marginal impact and no treatment
4	Risk with high probability, marginal impact and no treatment
3	Risk with low probability, serious impact and no treatment
2	Risk with high probability, serious impact and no treatment
1	No risks management

3.2 – Subjective Indicators for Solution and Process

Subjective indicators reflect more the supplier feelings about the offer and rely on his/her skill, expertise and point of view on the whole situation as well as his/her risk aversion. Indeed, the fact that all the technologies selected for the system solution are ready to be deployed does not guaranty that the system solution matches customer expectations. Moreover, certainly, not all sub-systems need a maximum readiness level as a prerequisite for an application and inversely, a given readiness level is not sufficient for selecting a technical solution. Following the same reasoning for the process, the fact that all the activities of the realization process have their main risk at level 9 with low probability of occurrence, marginal impact and plenty of treatments does not guaranty that the realization process will run correctly, without any hazard and any delay or additional cost. We therefore propose a first version of SFL, for Supplier Feeling Level, with a three-level scale. This scale corresponds to the feeling (bad, neutral or good) of the supplier about the offer. Table 3 presents the three levels of SFL. In our proposal, we associate an SFL to each sub-system of the system solution and each activity of the realization process.

Table 3. SFL Scale

Level	SFL
3	Good: customer's requirements seem achievable with no difficulty
2	Neutral: difficulty to make a decision on the requirements achievement
1	Bad: customer's requirements seem unachievable

3.3 – Aggregation Mechanisms

The offer overall confidence relies at the same time on TRL and SFL of the system side and ARL and SFL of the process side. Some aggregation mechanisms are needed at each level of the bill-of-material for the system solution, for the complete set of activity for the realization process and for the overall offer.

Let's start with the system side. When a system is composed of several sub-systems, its readiness level (SRL) depends on the TRL of each of it sub-systems and of the readiness of their integration or IRL. Several SRL calculation methods have been proposed in the literature and the most used is the one proposed in [13] or [14] and it is the calculation method adopted in this paper. This method leads to a five-level scale for SRL, as shown in table 4. We propose to use the same aggregation method for the objective indicators SFL of the system by taking into account the SFL of each sub-system as well as the SFL of their integration.

Table 4. SRL Scale

Level	SRL
9	Execute a support program that meets operational support performance requirements [0.9-1]
8	Achieve operational capability that satisfies mission needs [0.8-0.89]
7	Develop a system or increment of capability; reduce integration and manufacturing risk [0.5-0.79]
6	Reduce technology risks and determine appropriate set of technologies into a full system [0.2-0.79]
5	Refine initial concept; develop system/technology development strategy [0.1-0.19]

Let's continue with the process side. After determining the ARL of each activity of the realization process, the risk level of the whole realization process or PRL has to be computed. It is important to recall here that the phenomenon of integration as described in a system does not exist in the realization process as there is not yet any decomposition of activity into sub-activities. As a first stage, we propose to use an average method based on ARL to compute the PRL as well as its subjective indicators SFL of the activities.

Let's finish with the offer overall confidence. The offer overall confidence relies on both system solution and realization process and therefore should weight them equally. Therefore, as a first stage, we propose a two-step approach to compute the offer overall confidence. First, the objective indicators SRL and ARL are modulated by the subjective ones SFL: a good feeling increases the indicator, a bad feeling decreases it and a neutral one has no impact. The supplier has to specify how much it goes up and down. Second, the offer overall confidence is computed as the average of the modulated indicators.

3.4 – Offer Overall Confidence and Decision Making

When an offer is defined, the proposed original indicators (TRL, ARL, SFL) together with the proposed aggregation mechanisms allow a company to quantify a confidence that characterizes the offer. This new indicator associated with the traditional ones (cost, lead time and performance) enable the supplier to select the most promising offers with less stress and a better confidence.

The complete offer definition process is formalized as a constraints satisfaction problem. The use of filtering algorithm makes it possible to see the impacts of each choice on the whole offer. These choices can be for the system side, the selection of a technical solution or the valuation of an attribute, and for the process side, the selection of the adequate resource and its quantity. As previously said, choices have a direct impact on the TRL of the system solution and on the ARL of the realization process activities.

In addition, as constraints do not have any orientation, it is possible to force the observance of the customer's requirements, such as a minimal level of readiness for the technologies used in the system (TRL greater than 6 for instance), a maximum cost and the maximum lead time for the whole offer.

4. CONCLUSIONS

In this paper, we have proposed the first ideas in order to assess confidence in offers while bidding, from the supplier or bidder point of view. Our proposals are based on the extension of configuration process from ATO-MTO towards ETO situation. This extension is necessary as some configurations have never occurred and require to be specifically designed then produced. In order to cope with ETO situation, specific values have been added to the configuration models with a specific meaning.

Then, we have proposed the three new indicators to measure the degree of confidence in the overall offer. Two of them are objective and independent of the supplier (TRL and ARL). They characterize the readiness level of each sub-system and the risk level of each activity and are both based on a nine-level scale. The last one is more subjective and relies on the supplier feelings (SFL) about the offer and rely on his/her skill, expertise and point of view on the whole situation as well as his/her risk aversion.

Aggregation mechanisms have been proposed in order to compute the SRL of the system solution, the PRL of the whole realization process and the SFL for both system and process. In order to compute the offer overall confidence, objective indicators SRL and PRL are modulated by their respective SFL. Then, the offer overall confidence is computed as the average of modulated SRL and PRL.

Our proposals have been confirmed by several companies in system and service sectors, even the relative simplicity of the aggregation mechanisms. We have now to test it on real cases and to improve it with much more sophisticated aggregation methods. The use of Case-Based Reasoning and experience feedbacks could surely help the supplier in the valuation of the subjective indicators and the model updates.

REFERENCES

[1] W. Verhagena, P. Bermell-Garciab, R. van Dijkc, R. Curran - A critical review of Knowledge-Based Engineering: An identification of research challenges – Adv. Engin. Informatics Vol. 26, n° 1, pp 5–15, 2012.

[2] B. Chandrasekaran - Design problem solving : a task analysis. In Artificial Intelligence Magazine, Vol. 11, pp 59-71, 1990

[3] J. Olhager - Strategic positioning of the order penetration point – Int. J. of Prod. Economics – Vol. 85, n° 3, pp 319–329, 2003.

[4] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen - Knowledge-based Configuration From Research to Business Cases, Morgan Kaufmann, 2014.

[5] MR Endsley, D.G Jones – Chapter 7 Confidence and Uncertainty in situation awareness and decision making - Designing for situation awareness, Taylor & Francis, pp 113-121 – 2004.

[6] S. Mittal, F. Frayman - Towards a generic model of configuration tasks, in: Proceedings of IJCAI, pp. 1395–1401, 1989.

[7] M. Aldanondo, E. Vareilles - Configuration for mass customization: how to extend product configuration towards requirements and process configuration – J. of Intel. Manufacturing, Vol 9, n° 5, pp 521–535, 2008.

[8] T. Soininen, J. Tiihonen, T. Mannisto , R. Sulonen - Towards a general ontology of Configuration - AIEDAM Vol 12 n°4, pp 357–372, 1998.

[9] D. Sabin et R. Weigel - Product configuration frameworks - A survey - IEEE Intell. System, Vol 13, n°, pp 42-49, 1998.

[10] L.Zhang, E.Vareilles, M.Aldanondo - Generic bill of functions, materials, and operations for SAP2 configuration - IJPR, Vol. 51, n°2, pp 465-478, 2013.

[11] P.Pitiot, M.Aldanondo, E.Vareilles - Concurrent product configuration and process planning : Some optimization experimental results - Computers in Industry, Vol. 65, pp 610-621, 2014

[12] J.C. Mankins - Technology readiness level : A White Paper - Advanced Concepts Office - NASA April 6, 1995

[13] BJ Sauser, D. Verma, J. Ramirez-Marquez, R. Gove - From TRL to SRL: The Concept of Systems Readiness Levels. Conf. on Syst. Engineering Research, April 7-8, Los Angeles, 2006.

[14] W. Tan, B.J. Sauser, and J. Ramirez-Marquez - Analyzing component importance in multifunction multicapability systems developmental maturity assessment, IEEE Trans Eng Management, Vol 58, n° 2, 2011

Assessing the configurators user need for social interaction during product configuration process

Chiara Grosso*, Cipriano Forza*, Alessio Trentin*

Abstract. Commercial websites, including online sales configurators, are increasingly implementing technologies that enable social interactions, in order to provide users with social interaction possibilities that characterize retail shopping. One of these implementation strategies refers to connecting commercial websites to social software. Social software are web-based applications that support web users in social networking, interacting, sharing content and thus, in collecting feedback from various referents (e.g. user's online contacts, other web users, company representatives). Given the variety of possible connections between social software and configurators, mass customizers need to choose which connection(s) to implement, if any, to fulfil the configurator user need for social interaction. To make this choice, it is useful to be able to: (a) identify the various facets of the social interaction as user/consumer's navigation behaviour (b) measure the strength of configurator users' need for social interaction. Being able to assess the user need for social interaction could help mass customizers enhance the proactive support provided to the user during his/her configuration process. Accordingly, this study presents an exploratory analysis (a) by examining various facets of the social interaction need and subsequently (b) by proposing a multi-item scale to measure the social interaction need. The present paper aims at contributing research into the key role of feedback delivered to users from different referents during the configuration process and sheds further light on new online customers' needs.

1 INTRODUCTION

The social characteristic of the Web [1-3] is pushing companies to adopt selling strategies coherent with the social dimension of shopping on the web. Online vendors face a significant challenge in making their virtual storefronts socially rich [4-5]. However there are multiple ways of increasing sociability through the web interface of commercial web sites to positively impact consumer attitudes towards online shopping [6].

To engage consumers in an interactive and socially rich online shopping experience, commercial Web sites are implementing technologies that enable social interactions [7]. Social interaction refers to all action involving two or more people in which the behaviour of each person is in response to the behaviour of the other [8].

In particular, commercial web sites are increasingly using a set of web-based technologies called social software applications (hereafter addressed SocSW). Social software applications are defined as web-based software applications that enable people to connect, collaborate, create online networks and manage contents in a social and bottom-up fashion [9].

* *Università degli studi di Padova, Dipartimento di Tecnica e Gestione dei sistemi Ind.li, Vicenza, Italy. Corresponding author: C.Grosso, Ph.D., e.mail: chiara.grosso@unipd.it*

The same interest in social software application is increasingly growing between mass customizers that sell their product through online configurators and have started to connect their configurator to SocSW. The configurators are connected to social software applications in different modalities. Each modality enables various social interaction tools (e.g. text messages, image sharing, chat) that support the user enabling them to receive social feedback during his/her configuration process [10].

The importance of feedback during the configuration process has already been investigated in literature [11-13]. In particular, previous research showed the importance of peer feedback during the configuration process [11]. Also, research has shown that feedback influences the feeling of regret or satisfaction deriving from decision outcomes [14-16]. People are motivated to avoid post-decisional regret. The risk is that if the need for feedback is not identified and satisfied, it can lead the client to abandon the shopping process, in the online environment where the customer is more sensitive to small obstacles that can cause the termination of the shopping process, thus the configuration process [17].

The strength of the configurator user need for feedback has not been investigated yet. To what extent the implementation of SocSW responds to the user need for social interaction and which connection modality (if any) better fulfils the user's need for social feedback, is still unexplored. To understand the usefulness of the social interaction support provided by the configurator-social software connection, it raises the question of assessing how strong the need is for social interaction experienced by the user. In particular, to investigate the need for social feedback experienced by a user during the configuration process. A measure to evaluate the user's social interaction need is still missing.

This study presents an exploratory analysis in order to understand in more detail which factors are linked to the user's need for social interaction, and consequently, proposes a multi-item scale to measure this need. The aims of the present paper are both (i) to move forward understanding of customer's behavior in the specific shopping process via online configurators and (ii) to provide MCs with insights to evaluate if and to what extent configurator-SocSW connections fulfil the user's need for social interaction.

2 THEORETICAL BACKGROUND

2.1 Consumer interaction behavior while shopping

2.1.1 Factors that drive the customer to shop

Consumers' behaviour research studies three distinct activities: (i) shopping, (ii) buying, (iii) consuming [18]. The literature on consumer behavior underlines that shopping is driven by different factors. More specifically, there are two class of factors that impact on customer intention to shop: functional and non-functional [18-21]. Functional factors are linked to product acquisition (actual

buying of products) for example: time, place and product possession needs. Non-functional factors refers to the satisfaction of additional non product-related needs, for example social, emotional and epistemic needs [19-20]. The present study is interested in non-functional factors. Non-functional factors are divided into two categories: personal and social factors which drive the customer to shop [18-20].

Personal factors refer to customer-specific factors that determine the customer's intention to shop across a wide range of product types. Personal factors manifest the customer's shopping style, for example: economic shopper, personalizing shopper, ethical shopper style [19]. Personal factors: *(a) Individual role playing:* a factor determined by the customers' interest to act conforming to a certain position or role in society. *(b) Diversion from the routine of daily life:* a factor determined by customers' interest in recreation and diversion from daily life. *(c) Self-gratification:* a factor determined by customers' interest in buying something just for the pleasure of rewarding him/herself. *(d) Learning options about new trends:* a factor determined by customers' interest to learn and get new ideas about trends and symbols related to specific products. *(e) Physical activity:* a factor determined by customer's interest in doing physical exercise (e.g. go for a walk in a shopping street). *(f) Sensory stimulation:* a factor determined by customers' sensory benefits while shopping (for example background music, video or visual stimuli, even scent) [18,20].

Social factors refer to the social situation that determines the customer's intention to shop, for example social situations such as the presence of friends and relatives at the time of shopping [19]. Social factors: *(a) Social experience outside the home:* a factor determined by customer interest in being engaged in social interactions during shopping. The shopping experience provides a specific time and place for social interaction; *(b) Communication with others having a similar interest:* a factor determined by customers' interest for sharing the shopping experience with others with the same interest (for example, other customers). Also, interest in interacting with others who provide special information while shopping (for example, sales personnel). *(c) Peer group attraction:* a factor determined by customers' interest in the companion of peers or members from his/her reference groups while shopping. *(d) Status and authority;* a factor determined by customers' interest in commanding attention and respect for example, by shopping in a specific place or buying a particular product, or choosing a brand. *(e) Pleasure of bargaining:* it's a factor determined by customers' interest in enjoying the process of bargaining [18,20].

Research on online consumer behavior confirmed that non-functional factors have the same impact on consumer behavior during shopping both off-line and online [20]. Thus, consumers motivated by social interaction may choose to shop within a conventional retail store format as opposed to the online context [22]. Therefore, online retailers may find it more challenging to attract also shoppers who may be less predisposed to shopping online.

2.1.2. Consumer socialization process and socialization agents

Shopping is an activity that includes social interaction with others [23-24]. There is a strong relationship between consumer decision-making and the consumer socialization process [25-26]. Consumer socialization refers to the process by which individual consumers learn skills, knowledge, and attitudes from others through communication, which then assist them in functioning as consumers in the marketplace [27]. Consumer socialization theory states that communication among consumers affects their cognitive, affective, and behavioural attitudes [25, 27-28].

The socialization process can take three forms: (i) modelling, (ii) reinforcement, and (iii) social interaction. Each form represents a different mechanism by which the individual is socialized. Each socialization form has a different impact on the specific behaviour that an individual adopts to interact with others and participate in a social environment [29-30].

- *The modelling process* implies a mechanism of imitating or mimicking socialization agents because the agent's behaviour appears meaningful or desirable to the learner (Moschis and Churchill 1978). Socialization agents are those who have direct or indirect influence on an individual's behaviour (e.g. family, friends, peers, media, school) [29-30].
- *The reinforcement process* implies that the learner is motivated to adopt (or not) some behaviour or intentions because of a reward (or punishment) offered by the socialization agent [25, 27-29]. In particular, communication among consumers affects their cognitive, affective, and behavioural attitudes [25].
- *The social interaction process* implies interactions with socialization agents in social contexts, which may combine modelling and reinforcement [29].

Customers are interested in engaging relationships with different actors (socialization agents) while shopping to reduce their availability of choice, simplify their buying and consuming tasks, simplify information processing, reduce perceived risks, and maintain cognitive consistency and a state of psychological comfort [26].

Following the social learning approach, the socialization theory suggests that a consumer develops consumption attitudes and behaviour by learning from socialization agents through interactions with them even while shopping [31]. Research highlights consumer socialization agents who deeply influence the consumers' purchase decision: family, friends, peers, reference groups [32].

Peers are recognized as the most influencing socialization agents, beyond family members [7,27-29,33]. Consumers learn values, attitudes, and skills by observing others. Consumers tend to interact with peers regarding consumption matters, which greatly influence their attitudes toward products and services. Communication between peers is the strongest predictor of product placement attitudes and behaviour [30].

Beyond social interaction between customers and family, friends, peers, reference groups, the shopping process also includes social interaction between the consumer and company representatives (e.g. sales persons) [22-23]. 'Consumers have always been interested in relationships with marketers' [26:265]. Technological advantages, especially digital devices and social software application are facilitating the process of engaging and managing relationships with individual consumers [9].

2.1.3 Social software application as tools to support consumer social interaction

New interaction possibilities for Web users are changing user consumption behavior [34-41]. More specifically, social software applications have changed how consumers communicate because SocSW allow their users to interact and exchange information

about products/services with known and unknown people connected through social networks, virtual communities, blogs [34,37-39, 41].

Social Software applications provide virtual spaces for users to be connected in networks thus interaction is facilitated particularly among reference groups and peer groups [34, 38-39]. Research highlights that communication with reference groups and peers through Social software applications enable a form of consumer socialization that has a profound impact on consumer decision making [34, 40].

The socialization process enabled by social media is based on learning processes that simultaneously involve the three socialization mechanisms of (i) modelling, (ii) reinforcement, and (iii) social interaction [31, 38].

- *Modelling* - the ownership of a certain product or service owned by peers enables a modelling process.Thus, the consumer can buy the same product or avoid the product depending on whether s/he wants to be like peers or not.
- *Reinforcement* - pressure from peer and reference groups motivates the consumer to endorse a product or to purchase it because once a purchase via social media is shared it can be a source of rapid social rewards.
- *Social interactions* - SocSW provide communication tools that make the social interaction process easy and convenient (even costless) (e.g. blogs, instant messaging and social networking site). For example, in virtual communities new members can interact easily with virtual groups through electronic communication and quickly learn task-related knowledge and skills through their interactions with other members [42].

SocSW facilitate learning about products and trends by supporting information exchanges among multitudes of friends or peers (socialization agents) who provide different and numerous product information and enable, as well, quick evaluation of products [43].

Previous research suggested implementing SocSW in corporate websites to allow consumers not only the exchange of information about products or services but also to engage both current and potential consumers through participative and socializing experiences [41].

2.2 Online configurators and social software applications

2.2.1 Shopping experience via online configurators

One particular shopping process is shopping for personalized products [44]. This process happens more and more through online configurators [45]. Online configurators are defined as knowledge-based software applications that support a potential customer, or salespersons interacting with the customer, in completely and correctly specifying a product solution within a company's product offerings [44-48]. The selling approach through configurators has proven to be beneficial to both mass customizers [44, 48] and their customers [49-51].

Even the purpose of configurators is to support potential customers in choosing, within a company's product offering, the product solution that best fits their needs, configuration systems often outstrip user capability to identify a proper solution [46-48]. The more complex individualization possibilities are, the more information gaps increase [52] thus customers may experience uncertainty during the design process or have no clear knowledge of what solution might correspond to their needs. A customer may find him/herself in some circumstances (e.g. choice complexity, lack of knowledge, lack of experience) that enhance his/her uncertainty thus the need to receive feedback. The customer may feel overwhelmed by the number of product configurations available and leave the configuration process before purchasing [53]. This happens mostly when the customer find him/her self in a condition of choice complexity.

Choice complexity is defined as the amount of information processing necessary to make a decision and it's one determinant of the product variety paradox [54]. Another determinants of the choice complexity is post-decisional regret. In addition to the perceived risk of online shopping [55] regret aversion negatively influences consumer decisions, because the possibility of regret is anticipated, and subsequently experienced during decisions-making [14].

Recent studies suggest that a promising method for configurators to provide feedback would be to include a function that allows users to submit their (interim) design solutions for rapid social feedback from other users who are online [56]. The integration of social feedback during product configuration, more specifically, feedback from peers, stimulated favorably the customer's problem-solving process because 'MC toolkit users can assist each other during the development of the initial idea and during the design process and by giving each other constructive feedback on interim design solutions' [11:556].

2.2.2 Social interaction mechanisms provided for the user by connecting configurators to social software

Previous studies observed that a growing number of social media provide different supports to customers by sharing their created products and the possibility to share configured product via social media can foster customer-perceived benefits [10, 57-59].

Configurators are connected to social software through various modalities [10,57]. The focus of this paper is on those connection modalities integrated into the configurator. In particular, those modalities that provide social interaction support for the user in the configuration environment.

Table 1 reports a brief description of integrated-based modalities (M2.1, M2.2, M2.3, M3, M4, M7.1, M8), a synthesis of the configuration stage supported by each integrated-based modality (columns 1-3), the characteristics of social feedback provided to the user, in specific, from whom and when, the user is supported by each modality in receiving social feedback (columns 4-7). We adopted technical terminology provided by Franke et al. [11] to address three configuration stages, namely: initial idea development; intermediate evaluation; configuration evaluation. Accordingly, by partial product configuration, we mean a product configuration that has not been completed. By intermediate product configuration, we mean a preliminary product configuration that has not yet been selected as the preferred one. By final product configuration, we mean the product configuration that the user has chosen, possibly after considering various intermediate configurations. We adopted the following terminology to refer to the individuals with whom a configurator user can interact: online circles, that is, people that the user already knows, trusts, and is also in connection with via SocSW; peers, that is, unknown people of equal standing, such as other configurator users or other customers; expert sources, that is, unknown people that the user recognizes as experts, such as company representatives.

Table 1. Synthesis of configurator –social software integrated-based connections

Connection modalities	**CONF. stages**			**Social feedback characteristics**			
				From whom			***When***
	1	**2**	**3**	**Exp**	**OC**	**Peers**	**Real time**
M2.1 *Social media (SM) icons enable user to automatically publish the configurator link on his/her social profiles. E.g. of configurator: Puget Systems - https://www.pugetsystems.com/echo.php*	X				X		Yes
M2.2 *SM icons enable user to automatically share a complete configuration in user social profile(s). E.g. of conf: Tesla-Motors - http://my.teslamotors.com.*			X		X		Yes
M2.3 *SM icons enable user to automatically share a partial configuration in users social profile(s) while configuration is in process. E.g. of conf.: Nike- http://www.nike.com/us/en_us/c/nikeid*		X	X		X		Yes
M3 *Direct browse/upload into the configurator of files shared in the user's SM profile(s) E.g. of conf.: Personal Wine https://www.personalwine.com.*	X	X			X		NO
M4 *Simplified configurator embedded into company SM profile. E.g. of conf.: Vauxhall- https://www.facebook.com/vauxhall/*	X			X	X	X	Yes
M7.1 *Email to send complete configuration to user's online circles. E.g. of conf.: Colorware http://www.colorware.com/p-477-playstation-4.aspx*			X		X		NO
M8 *Instant message services to connect configurator users to company representatives E.g. of conf.: Ecreamery https://www.ecreamery.com/create-your-own-flavor*	X	X	X	X			Yes
Configuration process. *1: initial idea development; 2: intermediate evaluation; 3: configuration evaluation.* **Social-interaction characteristics.** ***From whom:*** *Exp.: expert sources (e.g., company representatives); OC: online circles; Peers: other configurator users or customers.* ***When:*** *Yes: in real time; No: not in real time.*							

Support at the configuration stages. Each integrated modality differently supports the configurator user at different stages of his/her configuration process.

The support provided by M2.1, M3, and M4 focuses on the early stages of the configuration process. M2.1 supports the user in sharing only a configurator link on social platforms; M3 in uploading items from online social folders into the configurator; M4 in making the first step of configuration on social media platforms (SM) because a basic configurator is integrated into a dedicated page in the company's SM profile. The support provided by M2.2, M2.3, and M7.1 focuses on the intermediate and final stages of configuration. M2.2 supports the user in sharing a partial configuration and M2.3 a final configuration on social platforms; M7.1 in sending the final configuration by email. Finally, M8 supports the user during the entire configuration process by providing a chat channel to configurator users.

Social feedback characteristics. Integrated-base modalities support the user in collecting social feedback from different referents and with different timing, depending on the interaction mechanisms enabled by each modality.

From whom. With the exception of M8, all integration-based connection modalities support the user in interacting with his/her online circles, thus, in receiving social feedback from already known people. Modality M4 allows users to share information also with peers (i.e. unknown people of equal standing) and expert sources (i.e. company representatives).

When. With the exception of M3, which does not support social interactions and M7.1 that supports a sharing option by email, the feedback process enabled by the integration-based modalities can be delivered to the user in real-time. Only M8 and M4 provide real-time feedback in the configuration environment. M2.1-3 enable real-time feedback delivered to the user only on social platforms.

3 RESEARCH AIMS & METHOD

In this study we first present an exploratory analysis (i) to identify the various facets of user social interaction need and (ii) to understand in more detail which factors are related to the need for social interaction (e.g. social feedback, referents to interact with). Secondly, we propose a multi-item scale to measure the need for feedback to assess the strength of the need for social interaction perceived by the configurator user.

Research method for exploratory analysis. A questionnaire was submitted to a panel of 34 (24 Male, 10 Female). The participants in the study were engineering students from the University of Padua with experience in the design of configuration system whom voluntary took part to the survey. The respondents also attended tutorials on: the configuration systems and its capabilities, the benefits a user can derives from configuration process, the different configurator-SocSW connection modalities, the different social interaction features enabled by each connection. During the seminar respondents were provide with materials on the explained topics (e.g.ppt slides, examples from previous researches).

To run a preliminary analysis of the respondents experience of the various facets of social interaction need, respondents perform a configuration process in groups of three to identify if the configurators were implemented with social software applications. Afterwards a questionnaire with five structured questions and multiple-choice answers, was provided to respondents. Items of multiple-choice answers were measured by a 5-point Likert scale (5 = totally agree, . . ., 1 = totally disagree). Positive statements have been proposed as negatively worded questions with an agree–disagree response format are often cognitively complex [60]. In order to identify the various facets of social interaction needs during the configuration process, we selected a sub-panel of 27 (20 Male, 7 female) selected for being web users always connected to social software applications both via mobile phone or personal computer.

Research method for measure development. The proposal of a measure for the need for social interaction is based on both the literature background (section 2.1 of the present paper) and the results collected from explorative analysis previously performed with the subsample of 27 respondents. To assess the quality of the measure it will be considered: to adopt procedure validated in previous research on configurator capabilities [61], to realize a construct validity and reliability of items selected to measure social interaction need and finally to realize a nomological analysis to test the existence of significant relationships with variables that are expected to be causally related to the need for social interaction.

4 RESULTS OF EXPLORATORY ANALYSIS

4.1 The level of connection of participants

From a total of 34 respondents, 27 were always connected (hereafter addressed always-on). To identify the various facets of social interaction need, we focus on the always-on subsample.

Always-on respondents represent the new generation of Internet user also named, millennials. Millennials are young people who are always connected to the web through web-connected devices (e.g. smartphone, tablets, pc) that communicate and even work mostly through those devices [62-64]. Young people adept at using Internet also represent the majority of business-to-consumer sales configurator users [51].

4.2 The view of the "always on" configurator users

4.2.1 Benefits from configurators implementing social interaction

In order to explore the always-on respondents opinion on the benefits deriving from configurator implement with social software, the following question was provided: "Which benefits can the user derive from a configuration experience on a configurator that implements social interaction?". The items refer to benefits already researched in mass customization literature, namely: creative achievement, hedonic benefits [65,50], uniqueness and self-expressiveness benefits [49,65-66]. Distribution of the levels of agreement with the proposed answers is reported in table 2. Percentages are grouped in three levels: 1-2 (totally and partially disagree, 3 (nor agree neither disagree), 4-5 (partially and totally agree). The 3rd point - neither agree nor disagree - was introduced into the scale, consistent with the option that respondents could not have a clear perception of the new proposed scenario.

Tab. 2 – Which benefits can the user derive from a configuration experience on a configurator that implements social interaction?

A configuration system that implements social interaction features:	**1&2**	**3**	**4&5**
Could motivate the user to be more creative	19%	15%	67%
Could provide a funny experience	4%	37%	59%
Allows the user to assert his/her uniqueness	11%	41%	48%
Allows the user to express his/her own personality	22%	37%	41%
Increases the user's pride of authorship	37%	19%	44%

Table 2 shows that there is wide consensus of benefits that can derive from configuration experience on a configurator that supports social interaction. Respondents agreed on the possibility of making the configuration an experience that inspires the user tobe more creative (i.e. creative achievement benefit). An interesting result is the respondents' agreement on considering the support of social interaction as a source of fun (i.e. hedonic benefit). Thus the user will benefit from an enjoyable configuration experience. Excluding the 41% of respondents with no clear preference, respondents agreed on uniqueness benefit. Similar consensus was manifested about self-expressiveness as a benefit that the configurator user can derive thanks to the social interaction support. The possibility of providing pride of authorship doesn't achieve a well-defined consensus from respondents.

4.2.2 The request for social interactions at different stages of the configuration process

In order to explore the respondents' opinion on the link between the configuration stages and the implementation of social interaction, the following question was provided: "when can social interaction features be a key factor during the configuration experience?". The items provided in the answers set refer to three stages of the configuration process: initial idea development; intermediate configuration evaluation; final configuration evaluation. [11]. The answers are summarized in table 3.

Tab. 3 - When can social interaction features be a key factor during the configuration experience?

	1&2	**3**	**4&5**
Evaluating his/her final configuration to increase his/her confidence about the final configured solution.	0%	26%	74%
Evaluating his/her intermediate configuration to improve his/her configuration while it's in process	7%	26%	67%
Developing his/her initial configuration idea development.	26%	26%	48%

It's interesting to note the wide agreement expressed by respondents on the key role played by social interaction features in supporting the user in the evaluation of his/her product configuration once it's completed. High is also consensus on the key role of social interaction features to support the user during his/her configuration experience. Not well defined is agreement on the stage of the development of the initial configuration idea.

4.2.3 The request for interactions with different social actors

In order to collect the respondents' opinions on the possibility of interacting with different actors during the configuration experience, the following question was formulated: "With whom do you think the user will prefer to interact during his/her configuration experience?" (Tab.4). Question 3 was meant to go in deeply into the respondents' opinion thus we propose a set of close answers with different degrees of user interest in the interaction options. Specifically, to explore in detail respondent preferences, instead of the scale of agreement (1 totally disagree... 5 totally agree) respondents were provided with a scale of interest in interacting. The scale to measure the interest in interacting was also from 1 to 5, where each level refers to (1) no interest in interacting with them; (2) interest in interacting if they are the only referent available; (3) sporadic interest in interacting with them; (4) interest in interacting with them; (5) strong interest in interacting with them.

Tab. 4 – With whom do you think the user will prefer to interact during his/her configuration experience?

User will prefer to interact with:	**1**	**2**	**3**	**4&5**
Other company customers because s/he considers them as experienced consumers of the company's products	4%	22%	26%	48%
User's online circles because s/he is confident about their interest in supporting him/her and are trustworthy sources of suggestions	0%	26%	30%	44%
Company representatives because s/he considers them as sources of professional feedback even if they are interested in selling company products	11%	15%	37%	37%
Other configurator users because s/he considers them experts of the configuration process	0%	19%	48%	34%

As reported in columns 1 and 4-5 respondent preference is to interact with user's friends/online circles, similar consensus is registered on interaction with other configurators users.
Company representatives are referents whom the user can be interested in interacting with if they are the only referent available or for occasional interaction or for interest.

4.2.4 The request for social interactions with different referents

In order to explore the respondents' preferences for interacting with different referents the following question was provided: "How can social interaction features be a key factor during the configuration experience?" and answers are summarized in table 5. Respondents evaluate the items of multiple choice answer on a 5-point Likert scale (5 = totally agree, . . ., 1 = totally disagree).

Tab. 5 – How can social interaction features be a key factor during the configuration experience?

If social interaction is enabled with:	**1&2**	**3**	**4&5**
company representatives, it has to be provided at each stage of the configuration process	26%	22%	52%
other configurator users, it has to be provided while the configuration is in process	19%	33%	48%
his/her online circles, it has to be provided in the configuration environment	19%	37%	44%
his/her online circles, it has to be provided at each stage of the configuration process	30%	48%	22%

Even if respondents have low interest in interacting with company representatives (tab.4), answers from table 5 show that interaction with company representatives can deploy a key role if it supports the user at each stage of his/her configuration process. Excluding the 33% of respondents with no clear preference, the majority of respondents considers a key factor interaction with other configurator users. Interactions with online circles do not constitute a key factor if provided at each stage of the configuration process. This percentage is consistent with respondents' preference for social interaction features that support the user at the final stage of the configuration process (tab. 3). Respondents agreed on the key role of the social interaction features if provided in the same environment where configuration takes place (configurator environment).

4.2.5 Sharing configuration experience with online friends

In order to explore respondents' opinions on links between social feedback and product sharing options with trustworthy referents, the following question was provided: "How do you expect the configuration experience to be on a configurator that supports the user in sharing his/her configuration experience with online circles?" The answers are summarized in table 6.

Tab. 6 – How do you expect the configuration experience to be on a configurator that supports the user in sharing his/her configuration experience with online circles ?

On a configurator that supports the user in sharing his/her experience:	**1&2**	**3**	**4&5**
The configuration experience will reduce the user's uncertainty about his/her purchase decision because the user could receive feedback about his/her configuration solution from people s/he knows and trusts	11%	26%	63%
The configuration experience will be entertaining	15%	22%	63%
Thanks to feedback provided by people s/he knows and trusts the user could collect suggestions to learn about his/her preferences about his/her configuration	15%	22%	63%
Thanks to interaction with people s/he knows and trusts the user could collect hints to learn about the product s/he is configuring	15%	41%	44%
The configuration experience will make the user more confident about his/her configuration because s/he could act in accordance with people s/he knows and trusts	26%	30%	44%

Results show the respondents' agreement about the reduction of user uncertainty on his/her purchase decision if s/he receives feedback from known and trusted people. High consensus is registered on the possibility of an entertaining configuration experience if shared with friends/online circles. Respondents agreed on the learning option enabled by a configuration experience shared with friends/online circles. Consensus of opinion is on the learning process linked to user configuration preference. A lower level of agreement is registered for the learning process linked to user knowledge of the configuration product. Respondents don't express a clear consensus on the confidence the user can derive by acting in accordance with people s/he knows.

4.3 Results overview

Even with its limitations, exploratory analysis provides useful hints to understand users' need for social interaction during the configuration process. Results show various facets of social interaction that configurator users always connected to social media platforms, expect from the implementation of configurator with social interaction features.

Configuration process. The implementation of the social interaction feature could inspire the user to be more creative (tab.2) and provide entertaining configuration experiences (tab.6). Social interaction features could reduce a user's uncertainty about his/her purchase decision (tab.6) and provide the user with insights to learn about his/her configuration preferences (tab.6)

From whom. Respondents expect the above-mentioned outcomes whether social interaction features support the user in collecting feedback from people s/he knows and trusts (tab.4). Beyond online circles, users prefer to interact with peers as experienced consumers of company products (tab.4).

When. Based on respondents' answers, social interaction features have to support the user in evaluating both his/her intermediate configuration in order to improve his/her configuration while it is in process (tab.3), and also his/her final configuration in order to increase his/her confidence about the final decision (tab.3). Social interaction features have to be provided for the user while the configuration is in process, thus in real time (tab.5)

5 TOWARDS MEASUREMENT OF THE CONFIGURATOR USER NEED FOR SOCIAL INTERACTION

5.1 Identifying measure items

5.1.1 Hints from exploratory analysis for measure development

Hints from explorative analysis on user social interaction preferences point out that the need for social interaction is perceived at different levels depending on the stage of the configuration process. The need for social interaction is perceived as a need to be satisfied at each stage of the configuration and mainly at the final stage of the configuration experience. Thus, a measure for social interaction need has to cover the need experienced during the entire configuration process.

The need for social interaction is linked to possible interactions that the user can establish with different actors (e.g. online circles, peers, expert sources: other customers, company representatives) during the shopping/configuration process. Exploratory analysis showed that users prefer to interact mostly with referents like their friends and online circles but at specific stages of the configuration process, such as the final stage.

Exploratory analysis showed that the satisfaction of the need for social interaction is linked to the user's uncertainty about his/her purchase decision, and his/her learning process about his/her configuration preferences (see subsection 4.2.5). Thus, the measure of the social interaction need has to consider the sharing option of the configured product before its purchase. Also, the measure has to consider the possibility to reduce user uncertainty and the learning option enabled by social interaction.

5.1.2.Generation items to measure user social interaction need

A review of previous research was undertaken to identify construct definitions and any existing measures. Based on the review, we identified seven items to measure social interaction need. Each identified item characterizes the construct of social interaction.

To develop a multi-item measure we can consider the items as defining facets of the construct [67] of social interaction. Those facets are reflected in the need for feedback experienced by the user during the configuration process. Whereby changes in social interaction (latent variable) are reflected (i.e. manifested) in changes in observable items [68]. Each item reflects (i.e. manifests) a specific function of the latent variable (social interaction) by considering the user's need for feedback from different actors that can impact on consumer behaviour during the shopping experience.

As introduced in the previous background section the consumer prefers to interact in particular with social agents as for example: family, friends, reference groups, peers. Items were selected in order to measure the need for social interaction as a need for feedback from those specific socialization agents during the configuration process. Proposed items to measure the user social interaction need:

- *During the configuration process I felt the need for feedback to reassure me in my choice*
- *Right from the beginning of the configuration process I felt the need to see other user's configuration choices*
- *During the configuration process I wanted to be able to confront my choices with those of other users*
- *During the configuration process I wanted online support from an expert operator who could guide me in decision-making*
- *During the configuration process I would have liked to receive feedback from some of my contacts*
- *If I had gone on to purchase a configured product, in making my final purchase decision I would have liked to confront my choices with those of other users*
- *Once the configuration process was terminated I experienced the impulse to share the product configuration I had created with other users*

5.2 A proposal for validation of the measure

5.2.1. Procedures for data collection and analysis to asses the quality of the measure

Data for measure validation could be collected through a number of configuration experiences performed by around 50 configurator users on roughly different configurators by using also a wider set of product types to increase the generalization of results. Every user could perform different experiences and every configurator should have been used by roughly 3-4 different users. After their experiences the users could answer questions on the social interactions needs they perceived during their experiences.

Through construct validity and reliability analysis it will be assessed whether the set of items proposed to measure social interaction need similarly reflect a single underlying latent construct. This analysis will guide researchers to deep understanding of the construct of social interaction need during product configuration.

In order to assess nomological validity we should test for the existence of significant relationships with variables that are expected to be causally related to the need for social feedback. We can focus on choice complexity within the company's product offer because choice complexity is a determinant that inhibits the user from investing the requisite time and effort in seeking the best option for him/her and interferes in his/her evaluation of the decision outcome itself [54, 69]. Social interaction during the configuration-shopping experience can enable recommendation dynamics based on interactions with others (e.g. peers, users 'online circles, company representatives). Those dynamics can provide the user with social feedback from trustworthy sources [70] that guide the consumer in his/her shopping for personalized products on configurators. Thus, social feedback can support the user in positively concluding with his/her configuration process and also support him/her in reducing his/her cognitive efforts caused by determinants of choice complexity (e.g. uncertainty, anticipated and/or post-decisional regret) [14].

CONCLUSIONS

The present study, firstly, explores the various facets of social interaction and subsequently proposes a multi-item scale to measure the user's need for social interaction during his/her configuration process. This study highlights that social interaction need is definitely perceived by users and this need depends on various drivers, such as: *from whom* social feedback is provided, thus with whom interaction is enabled by social software connected to configurators; *when* social interaction is supported (e.g. at which configuration stage) and *how* interaction occurs if social software are connected to configurators (e.g. in real-time while

configuration is in process, or not).

Based on our results the integrated-based connections M2.2 and M2.3 present the characteristics to fulfil the user need for social interaction with online circles while configuration is in process. Modality M4 responds to user interest in receiving social interaction while configuration is in process, but interactions supported by M4 can be only between users and company representatives. Feature research is needed to generalize results from exploratory analysis and to validate the proposed measure.

Once validated, the proposed measure aims at supporting MCs in assessing the configurator user need for social interaction and also in evaluating which social software connection (if any) implement configurators to effectively fulfill this need. By fulfilling the users' need for social interaction, mass customizers could both proactively support the user and also respond to social factors that drive customers to shop.

REFERENCES

[1] T. O'Reilly, 'What is Web 2.0: Design patterns and business models for the next generation of software', *Communications & Strategies*, **65**,17-37, (2007).

[2] T. Berners-Lee, J. Hendler and O. Lassila, 'The semantic web, *Scientific American*, **284,** 28-37, (2001).

[3] I. Maignan and B. A. Lukas,'The nature and social uses of the Internet: A qualitative investigation', *The Journal of Consumer Affairs, **31**, 346-371,* (1997).

[4] I. Benbasat and N. Kumar,'Para-social presence and communication capabilities of a web site: a theoretical perspective', *E-service Journal*, **1**, 5-24, (2002).

[5] N. Kumar and I. Benbasat,'Shopping as experience and website as a social actor: web interface design and para-social presence', *ICIS 2001 Proceedings*, **54**, (2001).

[6] K. Hassanein and M. Head, 'Manipulating perceived social presence through the web interface and its impact on attitude towards online shopping', *International Journal of Human-Computer Studies*, **65**, 689-708, (2007).

[7] X. Wang, C. Yu and Y. Wei, 'Social media peer communication and impacts on purchase intentions: A consumer socialization framework', *Journal of Interactive Marketing*, **26**, 198–208, (2012).

[8] H.T. Reis and L. Wheeler, 'Studying social interaction with the Rochester Interaction Record', *Advances in Experimental Social Psychology*, **24**, 269-318, (1991).

[9] W.A. Warr, 'Social Software: fun and games, or business tools?', *Journal of Information Science*, **34**, 591–604, (2008).

[10] C. Grosso, C. Forza and A. Trentin,'Support for the social dimension of shopping through Web Based Sales Configurators'

[11] N. Franke, P. Keinz, and M. Schreier,'Complementing Mass Customization Toolkits with User Communities: How Peer Input Improves Customer Self-Design', *Journal of Product Innovation Management*, **25**, 546-559, (2008).

[12] L.B. Jeppesen, 'User toolkits for innovation: Consumers support each other', *Journal of product innovation management*, **22**, 347-362, (2005).

[13] L.B. Jeppesen and M.J. Molin, 'Consumers as co-developers: Learning and innovation outside the firm', *Technology Analysis & Strategic Management*, **15**,363-383, (2003).

[14] M. Zeelemberg, 'Anticipated regret, expected feedback and behavioral decision making', *Journal of Behavioral Decision Making*,**12**, 93–106, (1999).

[15] T.L. Boles and D.M. Messick,'A reverse outcome bias: The influence of multiple reference points on the evaluation of outcomes and decisions' Organizational Behavior and Human Decision Processes, **61**, 262-275, (1995).

[16] K.Taylor, 'A regret theory approach to assessing consumer satisfaction', Marketing letters, **8**, 229-238, (1997).

[17] T. Rogoll and F. Piller, 'Product configuration from the customer's perspective: A comparison of configuration systems in the apparel industry', *In International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems*, Denmark, (2004).

[18] E.M Tauber, 'Why do people shop?', *The Journal of Marketing*, **36**, 46-49, (1972).

[19] J.N. Sheth, *An integrative theory of patronage preference and behavior*. College of Commerce and Business Administration, Bureau of Economic and Business Research, University of Illinois, Urbana-Champaign, 1981.

[20] A.G. Parsons, 'Non-functional motives for online shoppers: why we click.' *Journal of Consumer Marketing*, **19**, 380-392, (2002).

[21] A.J. Rohm and V. Swaminathan, 'A typology of online shoppers based on shopping motivations', Journal of business research, **57**, 748-757, (2004).

[22] J. Alba, J. Lynch, B. Weitz, C. Janiszewski, R. Lutz, R., A. Sawyer, and S. Wood, 'Interactive home shopping: consumer, retailer, and manufacturer incentives to participate in electronic marketplaces', *The Journal of Marketing*, **61**, 38-53, (1997).

[23] M.R. Solomon, *Consumer behavior: buying, having, and being*, Engelwood Cliffs, NJ: Prentice Hall, 2014.

[24] R. Dholakia, 'Going shopping: key determinants of shopping behaviors and motivations', *International Journal of Retail & Distribution Management*, **27**, 154-165 (1999).

[25] S.Ward, 'Contributions of socialization theory to consumer behavior research', *The American Behavioral Scientist*, **21**, 501, (1978).

[26] J.N. Sheth and A. Parvatlyar, 'Relationship marketing in consumer markets: antecedents and consequences', *Journal of the Academy of marketing Science*, **23**, 255-271, (1995).

[27] S.Ward, D.M. Klees and D.B. Wackman, 'Consumer Socialization Research: Content Analysis of Post-1980 Studies, and Some Implications for Future Work', *Advances in consumer research,* **17**, (1990).

[28] S.Ward, D.M. Klees and T.S. Robcnson, 'Consumer Socialization in Differcnt Settings: An lntemational Perspective', *Advances in consumer research*, **14**, 468-72, (1987).

[29] G.P. Moschis and G.A. Jr. Churchill,'Consumer socialization: A theoretical and empirical analysis', *Journal of marketing research*, **15**, 599-609, (1978).

[30] F. De Gregorio and Y. Sung, 'Understanding attitudes toward and behaviors in response to product placement', *Journal of Advertising*, **39**, 83-96, (2010).

[31] J.E. Lueg and R.Z. Finney, 'Interpersonal communication in the consumer socialization process: scale development and validation', *Journal of Marketing Theory and Practice*, **15**, 25-39, (2007).

[32] W.O. Bearden and M.J. Etzel, 'Reference group influence on product and brand purchase decisions', *Journal of consumer research*, **9**,183-194, (1982).

[33] S. Shim, J. Serido and B.L. Barber, 'A consumer way of thinking: Linking consumer socialization and consumption motivation perspectives to adolescent development', *Journal of Research on adolescence*, **21**, 290-299, (2011).

[34] S. Okazaki, 'Social influence model and electronic word of mouth: PC versus mobile internet', *International Journal of Advertising*, **28**, 439-472, (2009).

[35] T. Hennig-Thurau, E.C. Malthouse, C. Friege, S. Gensler, L. Lobschat, A. Rangaswamy and B. Skiera, 'The impact of new media on customer relationships', *Journal of service research*, **13**, 311-330, (2010).

[36] B. Cova, R. Kozinets and A. Shankar, (2012) *Consumer tribes*. Routledge

[37] P. Nambisan and J.H. Watt, 'Managing customer experiences in online product communities', *Journal of Business Research*, **64**, 889-895, (2011).

[38] W.C. Martin and J.E. Lueg, J.E., 'Modeling word-of-mouth usage', *Journal of Business Research*, **66**, 801-808, (2013).

[39] J. Zhang and T. Daugherty, 'Third-person effect and social networking: implications for online marketing and word-of-mouth communication', *American Journal of Business*, **24**, 53-64, (2009).

[40] T.S. Teo and Y.D. Yeong,'Assessing the consumer decision process in the digital marketplace', *Omega*, 31, 349-363, (2003).

[41] R.D. Mersey, E.C. Malthouse and B.J. Calder, 'Engagement with online media', *Journal of Media Business Studies*, **7**, 39-56, (2010).

[42] M.K. Ahuja and E.G. John. 'Socialization in virtual groups', *Journal of Management*, **29**,161-185, (2003).

[43] A.D. Gershoff and G.V. Johar. 'Do you know me? Consumer calibration of friends' knowledge', *Journal of Consumer Research*, **32**, 496-503, (2006).
[44] F.S. Fogliatto, G.J.C. da Silveira, and D. Borenstein, 'The mass customization decade: an updated review of the literature', *International Journal of Production Economics*, **138**, 14–25, (2012).
[45] M. Heiskala, J. Tiihonen, K.S. Paloheimo and T. Soininen, *Mass customization with configurable products and configurators: a review of benefits and challenges*, in: T. Blecker, G. Friedrich (Eds.), Mass Customization Information Systems in Business, IGI Global, London, UK, 1–32, (2007).
[46] A. Falkner, A. Felfernig and A. Haag, 'Recommendation technologies for configurable products', *AI Magazine*, **32**, 99-108, (2011).
[47] J. Tiihonen and A. Felfernig,'Towards recommending configurable offerings', *International Journal of Mass Customisation*, **3**, 389-406, (2010).
[48] C. Forza and F. Salvador, 'Application support to product variety management', International Journal of Production Research, **46**, 817–836, (2008).
[49] C. Grosso, A Trentin and C. Forza,'Towards an understanding of how the capabilities deployed by a Web-based sales configurator can increase the benefits of possessing a mass-customized product'. *In 16th International Configuration Workshop*, **21**, 81-88, (2014).
[50] A. Trentin, E. Perin, and C. Forza, 'Increasing the consumer-perceived benefits of a mass-customization experience through sales-configurator capabilities', *Computers in Industry*, **65**, 693-705, (2014).
[51] N. Franke, M. Schreier and U. Kaiser, 'The 'I designed it myself' effect in mass customization', *Management Science*, **56**, 125–140, (2010).
[52] N.Franke and F.T.Piller,'Key research issues in user interaction with user toolkits in a mass customisation system', *International Journal of Technology Management*, **26**, 578-599, (2003).
[53] C. Huffman and B.E. Kahn, Variety for sale: mass customization or mass confusion?', *Journal of Retailing*, **74**, 491–513 (1998)
[54] A. Valenzuela, R. Dhar and F. Zettelmeyer, 'Contingent response to self-customization procedures: implications for decision satisfaction and choice', *Journal of Marketing Research* **46**,754–763, (2009).
[55] G. Pires, J. Stanton and A. Eckford, 'Influences on the perceived risk of purchasing online', *Journal of Consumer Behaviour*, **4**, 118–131, (2004).
[56] N. Franke and C. Hader, 'Mass or only "niche customization"? Why we should interpret configuration toolkits as learning instruments', *Journal of Product Innovation Management*, **31**,1214-1234, (2014).
[57] P. Blazek, M. Kolb, M. Part and C. Streichsbier C, 'The usage of social media applications in product con_gurators', *International Journal of Industrial Engineering and Management,* **3**, 179-183, (2012).
[58] F.T.Piller and P. Blazek (2014) Core capabilities of sustainable mass customization. In: Felfernig A, Hotz L, Bagley C, Tiihonen J (ed) Knowledge based Configuration From Research to Business Cases. Morgan Kaufmann Publishers Inc., San Francisco, pp107-120.
[59] F.T. Piller, A. Vossen and C. Ih (2012) From social media to social product development: the impact of social media on co-creation of innovation. Die Unternehmung 65. http://ssrn.com/abstract=1975523.
[60] F.J. Fowler (1993) *Survey Research Methods*. Sage Publications, Newbury Park, CA
[61] A. Trentin, E. Perin and C. Forza, C., 'Sales configurator capabilities to avoid the product variety paradox: Construct development and validation'. *Computers in Industry*, **64**, 436-447, (2013).
[62] O. Peters and S.B. Allouch, 'Always connected: a longitudinal field study of mobile communication', *Telematics and Informatics*, **22,** 239-256, (2005).
[63] M.M. Tso, D.J. Gillespie and D.A. Romrell, 'Always on, always connected mobile computing'. *In Universal Personal Communications*, 5th IEEE International Conference, **2**, 918-924, (1996).
[64] J. Kubátová,'Innovative managerial principles for current knowledge economy', *Economics and Management*, **17**, 359-364, (2012).
[65] A. Merle, J.-L. Chandon, E. Roux and F. Alizon, 'Perceived value of the mass-customized product and mass customization experience for individual consumers', *Production & Operations Management*, **19**, 503–514, (2010).
[66] N. Franke and M. Schreier,'Product uniqueness as a driver of customer utility in mass customization', *Marketing Letters,* **19**, 93–107, (2008).
[67] J.R. Rossiter, 'The C-OAR-SE procedure for scale development in marketing', *International journal of research in marketing*, **19**, 305-335, (2002).
[68] A. Diamantopoulos and J.A. Siguaw, 'Formative versus reflective indicators in organizational measure development: A comparison and empirical illustration', *British Journal of Management*, **17**, 263-282, (2006).
[69] G.J. Fitzsimons, 'Consumer response to stockouts', *Journal of Consumer Research,* **27**, 249–266, (2000).
[70] C.N. Ziegler and J. Golbeck, 'Investigating interactions of trust and interest similarity', *Decision Support Systems*, **43**, 460-475, (2007).

Improved Performance and Quality of Configurators by Receiving Real-Time Information from Suppliers

Katrin Kristjansdottir, Sara Shafiee, Martin Bonev, Lars Hvam[1], Morten H. Bennick and Christian S. Andersen

Abstract. Companies providing customized products are increasingly applying configurators in order to support the sales and design activities. Yet, especially for engineer-to-order (ETO) companies such activities are often divided across different organizations, where throughout the configuration process product specification has to be retrieved across the supply chains. Therefore, it is required that relevant information from suppliers is included in the configuration process, either as sub-models or by integrating configurators across the supply chains. This study investigates the challenges associated with including suppliers' product specifications as sub-models and how these can be addressed by integrating configurators across supply chains to receive real-time information from suppliers. Based on established literature on the illustrated technical integration of configurators across the supply chains, this paper contributes with empirical evidence on the overall impact of its implementation. The results presented are based on a case study in an ETO company where it is supported that the complexity of the configuration models can be significantly reduced as well as the time devoted for the modelling and maintaining the systems. Furthermore, with the ability of receiving accurate and up-to-date information from suppliers, the quality of the specifications can be improved, which leads to reduced cost of the overall design.

1 INTRODUCTION

The ability to provide customized products has become more important across a wide range of industries [1]. To effectively guide communication with the customers and increase the quality of the product specifications, configurators are being applied to greater extent when defining product variants within the chosen scope of variety [2]. Such systems utilize formally expressed product architectures, i.e. knowledge bases, consisting of a set of components, their relationships, and constrains to prevent infeasible designs [3].

In engineer-to-order companies (ETO) the supply chains can be characterized by being tailored and complex [4], where manufacturing tends to be vertical integrated, including both internal manufacturing processes and outsourced supply [5]. Furthermore, the dynamic and segregated character of the early sales and engineering processes limits the availability of design information and increases the uncertainty of project's profitability [6]. As a result to this there is a high dependency of receiving information across the supply chains in the early sales design phases.

To address the complexity and the vertical integrated supply chains in ETO companies, the configurator's knowledge base needs to cover up to date product information related to the companies' own designs and of outsourced components/modules from suppliers. By including the suppliers' information as sub-models in the configurators there are some limitations, as the information are often confidential and sensitive for sharing outside the companies. Therefore, critical design detail and cost structures, which are often considered as confidential information, are not shared from the suppliers' side. This can result in insufficient level of detailed information being provided that can affect the overall quality of the configuration. Furthermore, rapidly changing components and modules supplied internally or externally drastically increase the effort for maintaining the configurator's knowledge base. This increases the risk of operating with outdated prices and variant designs and thereby decreasing the overall quality of the systems and the generated output. This underlines that centralized knowledge base is not desired, which emphasis the need of having distributed configurators across the supply chains [7].

The recent advancement of cyber-physical systems has enabled a closer integration of supply chains relationships [8], allowing for efficient ways of information management across multiple organizations. However, to make such an e-business environment possible, the established knowledge base needs to account for high degree of tailoring and dependency from suppliers [9]. Academia has proposed a technical approach that enables real-time information sharing across the supply chain by integrating configurators [7]. However, it's successful implementation and the actual impact from receiving the information directly from suppliers in the configuration processes has not been addressed in previous literature.

This paper aims to capture that research opportunity by analysing the overall impact from establishing the supplier integration to retrieve more accurate and up-to-date information across the supply chains in ETO companies. This includes description of the gained benefits, the challenges companies are faced within the process and directions for further improvements. Aligned with the focus of the research, the following propositions have been developed.

> **Propositions 1:** By integrating configurators across supply chains, the complexity in terms of business rules, tables, parts and values of the configurator model, and consequently the modelling and development effort can be reduced.

[1] Engineering Management Department, Technical University of Denmark, Denmark, email: katkr@dtu.dk, sashaf@dtu.dk, mbon@dtu.dk, lahv@dtu.dk

Propositions 2: By integrating configurators across the supply chains, the quality of the product specifications in terms of increased accuracy, more detailed and up-to-date, can be improved.

Propositions 3: The more detailed specifications from the supplier make it possible to improve the overall designs, which lead to cost optimization both for the component in focus and for other related components.

Aiming to investigate the impact of integrating configurators across the supply chains, a case study is introduced in an ETO company, which has established this integration with one of their supplier. The company operates globally and provides their customers with highly engineered and complex products and is thought to be a good representative of other ETO companies. The results of the case study are based on the in-depth interviews with the configuration engineers and managers at the case company as well as related supplier.

The paper is organized as follows. First, relevant literature is reviewed to identify the key constructs of the research model. In the next section the results in connection with the propositions and the managerial implications are presented. Finally, the main findings are discussed and concluded, and directions for further studies are elaborated.

2 LITTERATURE REVIEW

In this section the related literature is explored. The theoretical foundation for this article consists of configurators' main benefits and challenges and integrative information technologies in supply chains.

2.1 Configurators benefits and challenges

Configurators are used to support design activities throughout the customization process, where a set of components along with their connections are pre-defined and where constrains are used to prevent infeasible configurations [3]. The main technical component of the configurator is the knowledge base, which includes a database where the different components and their instances are stored along with the configuration logic representing constrains how different components can be combined [10].

Configurators have been considered as one of the key success factors in order to achieve the benefits from the mass customization approach [11], [12]. The main benefits of using configurators can be listed in terms of reduced lead times, improved quality of product specifications, preservation of knowledge, use of fewer resources, optimization of product designs, less routine work, improved certainty of delivery, reduced time for training new employees and increased customer satisfaction [13]–[15].

Even though configurators have proven to be beneficial and provide various benefits, there are some challenges concerned with utilizing such a system. The main challenges can be described in terms of supporting the customer in the customization process where the configuration process should be simple and short [10]. As a result of insufficient tools and methods, it can be difficult to guarantee consistency, completeness and formal documentation of the models and the long term management of interfaces and data can as well be a challenge [16]. Structuring and modelling product information [17], product characteristics, customer relations and long time span of the projects, and product complexity are also considered as one of the main challenges especially in ETO companies [18]. Lack of documentation which can lead to confusion about the variation possibilities [16], [19] and finally acceptance of the systems and change management as employees might see the implementation of the configurators as a threat to their job security [20] has also been named in relations to the challenges related to configurators.

2.2 Integrated information technologies across supply chains

Supply chain management involves the activities concerned with flow information and the transformation of raw materials to the end users [21]. In order to develop an integrated supply chain, a detailed top down approach is important, however successful achievement of integrated supply chain is more likely to happen through bottom up approach through a number of stages as shown in Figure 1 [22].

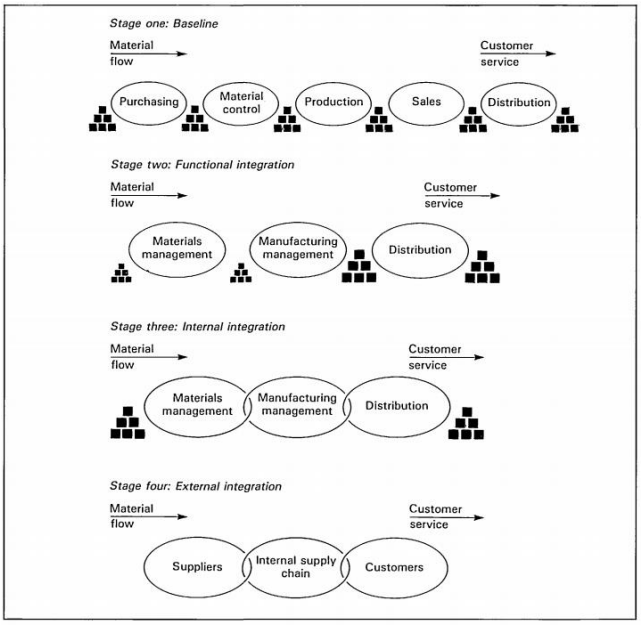


Figure 1. Achieve integrated supply chain [22]

There are a number of research that have explored the hypothesis "the higher the degree of integration across the supply chain, the better a firm performs" [22]–[27]. Ragatz et al. [28] identify the linked information systems applicability as a key success factor for integrating suppliers into the new product, process or service development process. Tallon et al. [29] point out that any positive impact of IT comes from its ability to coordinate value adding activities. A linkage between integrative IT and supply chain is a key aspect of supply chain integration. Stroeken [30] examines the link between IT and supply chain innovation in six industry sectors in order to show the importance of IT to develop the process oriented structure of the supply chain needed for the integration [30].

Mukhopadhyay and Kekre [31] quantify both strategical and operational impacts for Electronic Integration which leads to efficient procurement processes. The strategic benefits concerning the supplier and the operational benefits are in respect to both parties, or the suppliers and the customers. It should though be

noted that the operational benefits are generated by Electronic Data Interchange (EDI) through reengineering of the internal processes of an organization, unlike strategic benefits, which result from changes in the buyer-supplier trading relationship [31]. A supply chain strategy recognizes that integrated business processes create value for the companies' customers if these processes reach beyond the boundaries of the firm by drawing suppliers and customers into the value creation process [22], [32]. Vickery at al. [33] explain this linkage as the relationship between where one value activity is performed and the cost or performance of another is then introduced as the core purpose of supply chain integration as optimizing linkages amongst value activities.

IT development can lead to process innovation, or more broadly, supply chain integration, followed by cheaper, more diverse and customer-specific products. By considering organizations and markets, information processes makes the economic role of computers clearer [34]. To be successful, firms need to be able to adopt to computers as part of a system or cluster for reinforcing organizational changes [35]. Additionally, the extent clients achieve real time, or direct access to information maintained by service providers constitutes a goal of customization efforts efficiently and economically attainable through newly developed Internet-based technologies [36]. Suppliers utilize information specific to client requirements for global optimization of plans and adaptive execution of processes and these clients integrating logistics applications, enable suppliers to plan capacities for peak periods and exhibit requisite scalability of operations [9].

Configurators have been proven to be useful in distributed supply chains, where information from sub suppliers are retrieved in the configuration processes. Ardissono et al. [7] express the development of configuration services which offers personalized user interactions and distributed configuration and services in the supply chain. In Figure 2, the architecture for configurators setup integrated to the suppliers is demonstrated. The approach suggested is thought to support further cooperation, where the exchange of orders, publishing of product catalogues and the billing processes is supported in the supply chain [7].

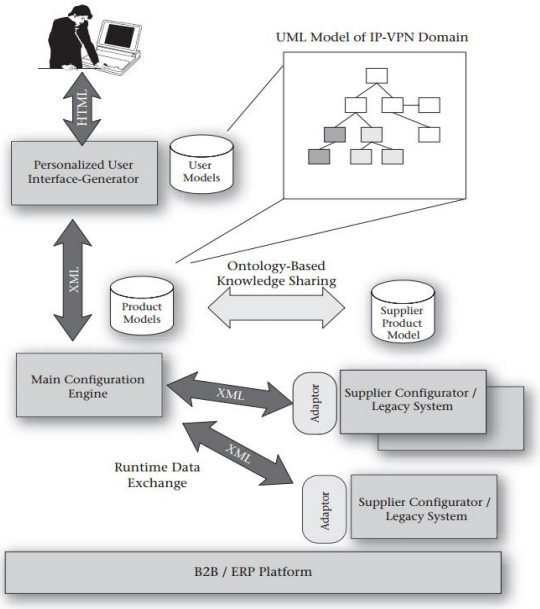


Figure 2. Architecture overview [7]

2.3 Summary of the literature

Based on the current literature in the field, the research highlights the importance of achieving greater integrations across the supply chains where IT plays a key role. Furthermore, for companies providing customized products, there is a need for having up-to-date information across the supply chains. Therefore, by integrating configurators across the supply chains, it allows companies to further integrate the flow of information and at the same time solve some of the main challenges concerned with mass customization and configurators. However, the impact from increased integration across the supply chains by enabling interactions of configurators across the supply chains has not been addressed previously in the literature.

3 CASE STUDY

3.1 Background information

The case company introduced in the study has a world leading position in providing cement plants and equipment for the minerals and cement industry. The company has utilized configurators since 1999 and has currently 136 operational configurators[2], which support the specification processes in the sales and the engineering at the company. The configuration setup at the case company has been addressed in previous researches where Hvam [37] describe the benefits and Orsvarn and Bennick [38] provide explanation of the overall configurations setup, integrations, output and the benefits. Even though, the company has been very successful in applying configurators to support their specification processes in the past, receiving up-to-date and accurate information from suppliers to use in the overall configuration process has proven to be a challenge.

The case company has a great number of suppliers providing it with customized products to be used in the overall design. Therefore, there is a close dependency of receiving relevant product information and prices from suppliers in the configuration process. In many cases products are sourced from several suppliers and it has to be considered which supplier is the most suitable one for a particular project. The initial strategy for past years was to include high-level product specifications from each supplier in form of sub-models, modelled and maintained directly in the configuration system. This additional responsibility requires a regular follow up activity with the suppliers to ensure the correctness and validity of the product specifications. There are several challenges reported using this approach, as the knowledge is not available in-house it can be difficult to access and validate it. Furthermore, with no mechanism in place for the required supplier updates to be communicated, the company has to compromise on the overall configuration quality and generated specification outputs.

In order to overcome these challenges, the company has made an integration to one of their gear supplier's configurator via API web services as suggested by [7]. Through this integration, information can be retrieved directly during the configuration process, thereby leaving the modelling and maintenance task to their suppliers. Through that the suppliers can obtain the

[2] A configurator is defined as model based expert system with it is own knowledge base and inference engine.

confidentiality of sensitive product data while increasing the level of details and optimization and ensuring up-to-date provided specifications.

In this chapter, first the procedure to include the suppliers' information before the supplier integration and the main limitations to those procedures will be elaborated. Secondly, the technical setup and the protocols will be explained in order to give more understanding of the overall technical setup for this specific case. Thirdly, the impact from integrating the configurators across the supply chains will be explained in relation with the propositions. Finally, the suppliers' incentives for providing the integrations and the main organizational challenges with establishing the setup will be addressed.

3.2 The prior documentation of the suppliers' information

To include the suppliers' information in the internal configurators used at the case company, three different methods have been used over the years. The method selected to document the supplier's information each time depends on the product complexity and the availability of the product information. Following is a brief description of those methods.

- The first method includes making a list of all possible configuration of the supplied product. In cases where highly complex product with great number of possible configurations, it will become impossible to map down all different configurations. Therefore, a limited number of possible combinations of the products and pre-calculated ranges of values are included in the configurator for the product.
- The second method includes building a configuration model based on the supplier's documentation, which allows covering all different configurations even for complex products. However, the main limitations can be traced to the knowledge not being available for the programmers, which makes it difficult to access and validate the models. Furthermore, changes over the time are not always communicated, which can result in invalid or inaccurate configurations of obsolete supplier designs.
- Finally, the third method is to integrate with .DLL[3] files provided by the supplier. The .DDL files can contain both codes and data, which enables that the program division into separate modules. Therefore, the .DDL files from the suppliers can be incorporated into the configuration system as separate components of the program. In these cases, where .DDL files are used, it has to be assured that in case of any changes, the supplier will send an updated file to the company. Furthermore, the suppliers are in most cases not willing to share company critical information. Therefore, these files are often missing product related information concerning the sensitive aspect of the design and the overall cost structure.

Even though, these approaches have been used at the company to include the suppliers' information, they are not without limitations. The main limitation is the insufficient level of detail of the included product specification and its availability in an up-to-date form. In order to overcome these limitations, the suppliers could be contacted every time an input or a proposal from them is required. However, that would delay the overall process, as the lead-time for receiving input or proposal can take weeks. Furthermore, this requires resources being available both at the company and the supplier to request and send the information. This scenario is therefore regarded being unfeasible or impractical. With the current technological progress, an alternative approach to receive up-to-date and accurate products' information from suppliers is to establish integration that allows data exchange in automatic and efficient way. Here, the case company has decided to connect its internal configurator via API web services to the supplier's configurator. During the configuration process input parameters configured in prior steps are sent to the supplier's configurator, which calculates possible solutions within the given criteria in 0,1 - 0,2 seconds and send back the requested product specifications. This setup enables the company to use the correct and up-to-date designs. Besides, suppliers have the ability to optimize the design for the particular customer requirements with a greater level of detail, instead of using a fixed range of pre-calculated calculations. The technical setup used in this case study is further described in next section.

3.3 The technical setup and the protocols at the case company

The case company and the supplier both had operational configurators used for internal operation to support the sales and engineering processes. The technical setup allows the configurators at both companies to interact (business-to-business communication) in order to retrieve real-time and accurate product configuration from the supplier. In Figure 3, the setup of the supplier integration in the case company is demonstrated. The company has currently established integration with one of their suppliers but has planned to expand the numbers of suppliers in close future as is shown on the figure below. By expanding the number of suppliers it both allows expansion of the parts that can be configured via the integration and also by including number of suppliers providing the same product the most desirable supplier can be found each time in automatic way, which is done manually today.

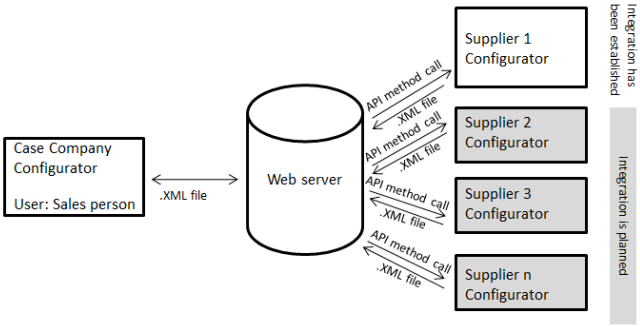


Figure 3. The technical setup at the case company: the supplier integration via API web services

[3] Dynamic link library (DDL)

3.3.1 The setup for transferring data from one system to another system

Confidential data are transferred across the companies and therefore special security methods are required. In this specific case, the confidential part is limited to the pricing logic as different product designs are already accessible for customers in product catalogues. Therefore, by establishing the integration the supplier does not have to revile the logic behind the pricing as only the final price for the specific configurations are reviled. In order to reduce the risk from the supplier's site of sharing confidential information, several methods have been established. Those method are not only limited to the prices but to the overall access of the information that can be gathered from the supplier's configurator.

In order to prevent spying collection, data tracking and men in the middle attack, a third party is not used for transferring the data and the data communication is directly established between the two companies. The case company has special access rights to the supplier's server, which can be used without identification after login. The initial login therefore only enables persons having access to the configurators at the case company to access the supplier's configurator as the server is not accessible without the login. In addition at the case company, the access rights are not shared with the whole company as it is only available for the employees, which needs to work with the specific configuration/product model. These security methods should therefore protect the supplier from misusages of the integration both from the case company and from other external threats.

3.3.2 Input and output parameters

The data exchange between the case company and the supplier is done via .XML files. The case company sends 20 design parameters (such as min/max torque, what the reduction should be in the gearbox, gear factors), which are defined in the previous steps of the configuration process. The request is to find a design within these parameters, where the supplier's configurator, based on their logic and business rules, find all possible design solutions, which can be around 100 and the prices for the different designs. It is highly unlikely that the supplier's configurator will not be able to find feasible solution. However, if that situation comes up either parameters have to be changed in the configuration at the case company or the supplier has to be contacted. The design solutions are sorted according to prices (from lowest to highest) and sent back on an .XML format via the web API web services. For this specific product, the prices are most important and therefore the cheapest solution is automatically selected by the case company's configurator. It should though be noted that other parameters can be used to sort after, such as in terms of quality, lead-time etc. The information retrieved from the supplier is then used in the further steps of the configuration as the dimensioning of the product, will affect the overall design under configuration at the case company.

3.4 The impact from integrating configurators across the supply chains

3.4.1 Reduced complexity of the configuration model

The configurator models operated at the case company contain a number of sub-models that in turn include parts and modules bought from suppliers (as described in section 3.2). Outsourcing these sub-models, the complexity of the configuration model has been reduced. By reducing the complexity, in terms of business rules, tables, parts and values, of the configurators' models, the development and maintenance effort can simultaneously be reduced as the supplier's configurator is accessed in the configuration process. The supplier therefore becomes responsible of developing and maintaining his own products' information. In Table 1, it is summarized how the supplier integration affects the complexity of one of the configurator's model operated at the case company and the impact is has on the development time.

Table 1 Summary of reduction of complexity in the configuration at the case company

Characteristics of the configurator	Before the supplier's integration	After the supplier's integration
Business rules	86	0
Tables	13	0
Parts	17	1
Values	18.836	20
Development time of the system	8+ days	2 days
Specialist time spent on the development	8+ days	0 days

3.4.2 Improved quality of the specifications in terms of updated and more detailed product information

An important aspect of the proposed approach is improved quality of the products' specification as they are based on real-time, optimized and more detailed information. This secures a valid solution, right dimensioning of the product under question and exact and up-to-date prices are used in the overall configuration process.

For the product provided by the supplier addressed in this case study that is gears, the numbers of possible configurations for a product are 25-26 millions. When having so many possible combinations, it is not feasible to include them all by using Excel sheets or preliminary databases as it will take too long time to look up and affect the time it takes to start up the configurators. Therefore, for the product in question in this case study only 20 different configurations were included (out of 25-26 millions) in the configurators before the integration. As a result to this, the company was not using the most optimal design of the supplier's product (as feasible solution is selected based on limited number of configurations). The solution that was chosen was always scaled up to the predefined range, which means that surrounding systems also needed to be scaled up. As if one part of the design is over dimensioned other parts have to be adjusted accordingly, which will cause a snowball effects in the overall design. In Figure 4 this is demonstrated where the blue line represent the predefined configuration that would have been selected prior to the supplier integration and the red line represent the exact configuration, which can be selected as a result to more detailed information retrieved after the supplier integration was established. The product' dimensions for this specific product are determined based on required kilo watts (kW).

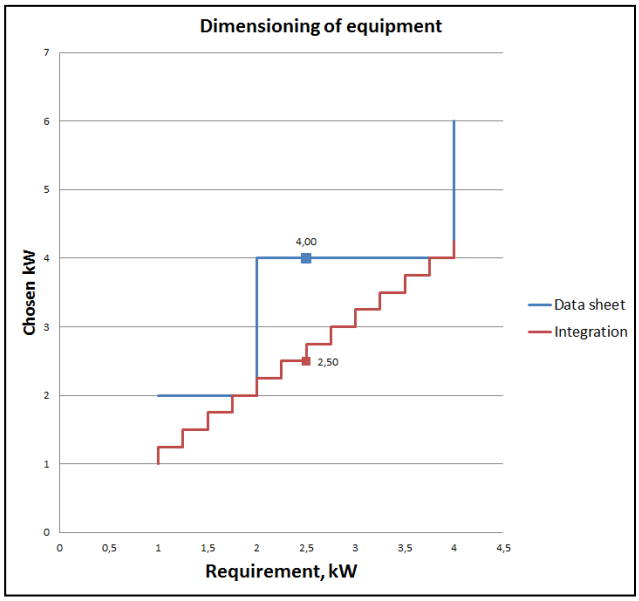


Figure 4 Dimensioning intervals of the equipment before and after the supplier integration

Having the precise dimensions of the supplier's product in the configuration process has proven to improve the accuracy of the generated specifications and reduce over-dimensioned surrounding systems. Therefore, the company has archived both immediate and in-direct cost savings as a result to more detailed product information. The immediate cost saving, for the example presented in Figure 4, is the difference between the 4,00 kW and 2,50 kW gear while the in-direct cost savings represent the related systems, or the frame as the gear is positioned on and again the platform area, weight of supporting building and etc. It is estimated that the company saves up to 20% in material cost in the overall design by having more detail information in the design phase.

3.5 Supplier incentive for providing integration

From a supplier perspective this approach provides additional benefits as it allows the supplier to protect sensitive product information, as these are considered as a secure black box in the configuration process. The supplier also saves resources for generating and sending proposals to their clients and thereby drastically reducing lead-times across the supply chains. Finally, the supplier hopes to increase their business share in long term with the case company as when this integration has been established it can easily be expanded to include additional products provided by the supplier.

3.6 Challenges with the approach

The main challenges can be related to legal barriers from both parties and to identifying suppliers that have the capabilities for the suggested collaboration with respect to operating with configurators.

For the companies addressed in this case study this is the new way of doing business, which needs the management and power to be able to execute it in a bigger scope so both parties can get some substantial gains from it. The main challenges can therefore be described in terms of organizational and not in terms of technical challenges. From the technical aspect, the whole programming was done in 2 days for the first time and afterwards for other integrations it was even less than 1 day, which highlights that the integration can be established without great effort.

4 DISCUSSIONS

The supplier integration used in the customization process where configurators are connected via API web services has proven to improve the overall process and provide substitutional benefits both for the case company and their supplier. This can be traced to accuracy of the suppliers' data, where more detailed and optimize information are provided, which are constantly up-to-date. This has enabled the case company to save up to 20% of the overall material cost in the overall design. Furthermore, the complexity of the configuration models can be reduced and the time consuming task of modelling and maintenance are delegated to the supplier. Finally, with this setup the supplier does not have to revile the actual logic behind the designs and the pricing strategy as the supplier's configurator is treated as a black box in the configuration process.

As the application of the configurators is constantly increasing, this integration to supplier's configurators becomes more realistic. That is since the requirement for making the integration is limited to the suppliers having operational configurators or willing to develop a configurator, which is capable of covering the required configurations. In addition to the integration that has been established at the case company four other suppliers have been identified that fulfil these requirements and have approved to participate in the project.

Further work at the case company with this approach will therefore include establishing the integration to greater number of suppliers, where comparisons capabilities of the configurator are used to identify the most suitable supplier. As for each product bought at the company there are several suppliers able to provide the product. For plant equipment, the aim is to have 2-3 suppliers for each of the products and the most favourable supplier each time will get the quote. The criterion for selecting the most desirable supplier has to be selected in the system for different products. In many cases the cheapest supplier would get the quote but it could also be lead-time, quality etc. The configurations retrieved from the suppliers are then sorted based on the selected criteria and the best one is selected by the system. This will automate the processes of comparing different suppliers' offers, which is done manually in the company today. For configurations on plant level there are preferred suppliers and therefore this cannot be applied in these cases. However, the comparison capabilities can be used to analyse the impact from changing the preferred suppliers to see the effect it has on prices, delivery-time etc.

The company has also made plans to increase the amount of documents retrieved from the suppliers in the configuration process. Therefore, further work will include making it possible to retrieve documents such as, 3D models and technical specifications as now only prices and dimensions of the product are received. Furthermore, currently the integration is only used to receive data as input in the configuration process, where the procurement will then contact the supplier to make the actual order purchase. In close future it is anticipated to automate that as well, so that the product can be requested from the supplier via the integration.

5 CONCLUSION

The present paper analyses the impact from having integrated configurators in the supply chains in an ETO company. The approach suggests the involvement of configurators that retrieve accurate sub-product information in real-time from suppliers during the customization process. The results indicate an improved quality of the product specifications and reduced complexity of the configurator model. Three propositions were developed to analyse the impact from integrating configurator across the supply chains to retrieve more accurate, detailed information and optimized in the configuration processes.

The first proposition investigates if by applying this approach the complexity of the configurator model can be reduced. The modelling and development effort proved to be reduced at the case company as they are not responsible for modelling the supplier's product information. Thereby the modelling and maintenance effort is moved to the supplier. The findings support this proposition as the complexity, which is defined in numbers of business rules, tables, parts and values is reduced to almost zero. This also effects the development time of the system which is reduced from 8+ days to 2 and the specialist time spent on the development has been reduced from 8+ to 0.

The second proposition questions if by integrating configurators across the supply chains, the quality of the specifications generated by the configurators will increase. The quality of the configurators model in this article is defined in terms of improved accuracy as the information retrieved via the supplier integration are optimized, more detailed and up-to-date. The findings support this as over dimensioning of different parts is not required as a result to improved quality of the products' specifications.

Finally, the third proposition is concerned with the improved quality of the specifications will lead to cost savings at the company. The result indicate that the company can save up to 20% of material cost as a result to immediate and in-direct savings gained from over dimensioning both the supplier's product and the surrounding systems. The results based on this study indicate that significant benefits can be gained from increased supply chains integrations in ETO companies where integrated configurators are distributed across the supply chains.

REFFERNCE

[1] F. Salvador and C. Forza, 'Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities', *International journal of production economics*, **91**, 273-291, (2004).

[2] C. Forza and F. Salvador, 'Application support to product variety management', *International Journal of Production Research*, **46**, 817-836, (2008).

[3] A. Felfernig, G. E. Friedrich, D. Jannach, 'UML as Domain Specific Language for the Construction of Knowledge-based Configuration Systems', *International Journal of Software Engineering and Knowledge Engineering*, **10**, 449-469, (2000).

[4] P.A. Konijnendijk, 'Coordinating marketing and manufacturing in ETO companies', *International Journal of Production Economics*, **37**, 19-26, (1994).

[5] C. Hicks, T. McGovern and C. Earl, 'Supply chain management: A strategic issue in engineer to order manufacturing', *International Journal of Production Economics*, **65**, 179-190, (2000).

[6] N. H. Mortensen, L. Hvam, A. Haug, P. Boelskifte, C. Lindschou and S. Frobenius, 'Making Product Customization Profitable', *International Journal of Industrial Engineering: Theory, Applications and Practice*, **17**, 25-35, (2010).

[7] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schafer and M. Zanker, 'A Framework for the Development of Personalized, Distributed Web-Based Configuration Systems', *Ai Magazine*, **24**, 93-108, (2003).

[8] L. Petnga and M. Austin, 'An ontological framework for knowledge modeling and decision support in cyber-physical systems', *Advanced Engineering Informatics*, **30**, 77-94, (2016).

[9] R. Klein, 'Customization and real time information access in integrated eBusiness supply chain relationships', *Journal of Operations Management*, **25**, 1366–1381, (2007).

[10] T. Blecker, N. Abdelkafi, G. Kreuter and G. Friedrich, 'Product configuration systems: State-of-the-art, conceptualization and *extensions' in: A.B. Hamadou, F. Gargouri, M. Jmail (eds.): Génie logiciel & Intelligence artificielle, Eigth Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI), Sousse, Tunesia*, 25–36, (2004).

[11] F. Piller and P. Blazek, *Core capabilities of sustainable mass customization*, 107-120, Knowledge-Based Configuration From Research to Business Cases, (eds.) A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, Morgan Kaufmann Publishers, Waltham, 2014.

[12] B. J. Pine, *Mass customization: the new frontier in business competition*, Harvard Business Press, 1999.

[13] F. Piller, K. Moeslein and C. Stotko, 'Does mass customization pay? An economic approach to evaluate customer integration', *Production planning & control*, **15**, 435-444, (2004).

[14] L. L. Zhang, 'Product configuration: a review of the state-of-the-art and future research', *International Journal of Production Research*, **52**, 6381–6398, (2014).

[15] L. Hvam, N. H. Mortensen and J. Riis, *Product customization*, Springer, Berlin Heidelberg, 2008.

[16] J. Tiihonen, T. Soininen, T. Männistö and R. Sulonen, State of the practice in product configuration–a survey of 10 cases in the finnish industry, 95-114, Knowledge intensive CAD, Springer US, 1996.

[17] C. Forza and F. Salvador, 'Product configuration and inter-firm co-ordination: an innovative solution from a small manufacturing enterprise', Computer in Industry, **49**, 37–46, (2002).

[18] T. D. Petersen, *Product Configuration in ETO Companies*, 59-76, Mass customization information systems in business, (eds.) T. Blecker, Igi Global, 2007.

[19] T. Soininen, J. Tiihonen, T. Männistö and R. Sulonen,'Towards a general ontology of configuration', AI EDAM. **12,** 357–372, (1998).

[20] C. Forza and F. Salvador, 'Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems', *International journal of production economics*, **76**, 87–98, (2002).

[21] R.. Handfield and E. Nichols, *Introduction to supply chain management*, NJ: prentice Hall, Upper Saddle River, 1999.

[22] G. Stevens, 'Integrating the supply chain', *International Journal of Physical Distribution & Materials Management*, **19**, 3-8, (1989).

[23] R. Metters, 'Quantifying the bullwhip effect in supply chains', *Journal of operations management*, **15**, 89-100, (1997).

[24] H. Lee and V. Padmanabhan, S. Whang, 'Information distortion in a supply chain: The bullwhip effect', *Management science*, **50**, 1875-1886, (2004).

[25] M. Frohlich and R. Westbrook, 'Arcs of integration: an international study of supply chain strategies', *Journal of operations management*, **19**, 185-200, (2001).

[26] P. Hines, N. Rich and J. Bicheno, 'Value stream management', *The International Journal of Logistics Management*, **9**, 25-42, (1998).

[27] R. Johnston and P. Lawrence, 'Beyond vertical integration–the rise of the value-adding partnership', *Harvard business review*, 94-101, (1991)

[28] G. Ragatz, 'Success factors for integrating suppliers into new product development', *Journal of product innovation management*,**14**, 190-202, (1997).

[29] P. Tallon and K. Kraemer, 'Multidimensional Assessment of the Contribution of Information Technology to Firm Performance', *Center for Research on Information Technology and Organizations*, (1996).
[30] J. Stroeken, 'Information technology, innovation and supply chain structure', *International Journal of Services Technology and Management*. **2**, 269-288, (2001).
[31] T. Mukhopadhyay and S. Kekre, 'Strategic and operational benefits of electronic integration in B2B procurement processes', *Management Science*, **48**, 1301-1313, (2002).
[32] K. C. Tan, 'Supply chain management: supplier performance and firm performance', *Journal of Supply Chain Management*, **34**, 2, (1998).
[33] S. Vickery and J. Jayaram, 'The effects of an integrative supply chain strategy on customer service and financial performance: an analysis of direct versus indirect relationships', *Journal of operations management*, **21**, 523-539, (2003).
[34] J. Galbraith, *Organizational Design*, MA: Addison-Wesley, 1977.
[35] P. Milgrom and J. Roberts, 'The economics of modern manufacturing: Technology, strategy, and organization', *The American Economic Review*, 511-528, (1990).
[36] E. Brynjolfsson, L. Hitt, 'Beyond computation: Information technology, organizational transformation and business performance', *The Journal of Economic Perspectives*, **14**, 23-48, (2000).
[37] L. Hvam, 'Mass customisation of process plants', *International Journal of Mass Customisation*, **1,** 445–462, (2006).
[38] K. Orsvarn and M. H. Bennick, *Tacton: use of Tacton configurator at FLSmidth*, 211-218, Knowledge-Based Configuration From Research to Business Cases, (eds.) A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, Morgan Kaufmann Publishers, Waltham, 2014.

Deriving Tighter Component Cardinality Bounds for Product Configuration

Richard Taupe and Andreas A. Falkner and Gottfried Schenner[1]

Abstract. Product configuration is the task of specifying products given a set of components and constraints for their combination. Although the types of components do not vary while solving a configuration problem, the exact component cardinalities (i.e. numbers of components of each type) are typically not known beforehand. Automatic configurators benefit from tight lower and upper bounds for these cardinalities, especially in large-scale domains.

In this paper, we show how to generically derive additional constraints on component cardinalities from object-oriented configurator models. This is achieved by utilizing information contained in the multiplicities of associations. In order to calculate tight bounds, we introduce new inequalities based on association specialization. We show that this leads to tighter cardinality bounds compared to earlier approaches. These bounds can increase the performance of a constraint solver significantly, as we demonstrate using a MiniZinc encoding of object-oriented configuration models.

1 INTRODUCTION

Complex products consist of many components. For some of them, the multiplicities are fixed (e.g. a car has 4 wheels), whereas for others the number of components of a special type is configurable, depending on technical constraints or preceding decisions made by the user. An example of this are railway interlocking systems. Real-world examples of such systems often consist of hundreds of different part types and tens of thousands of configured parts for hardware, software, user interfaces, and communication equipment [12].

Almost all knowledge representation languages used for product configuration allow to express constraints on cardinalities. Object-oriented formalisms provide association multiplicities [21], Description Logics (OWL) offer qualified number restrictions [3], and in classical component-port models the number of ports of a type restricts the number of connected components (e.g. 4 wheel-ports for a car) [22].

In order to optimally support a user or an algorithmic solver to find a consistent solution, it is helpful to know lower and upper bounds on the numbers of components. These bounds should be as tight as possible. Sometimes the cardinality information alone is sufficient to decide if a configuration exists at all, e.g. one cannot configure n cars with less than $4n$ wheels.

Most of previous work on reasoning about UML class diagrams has focused on their satisfiability and verification in the context of software engineering [4, 5, 11, 18]. However, a product configurator's fundamental reasoning task is not to decide satisfiability but to actually find an instantiation of the object model which corresponds to a buildable artefact in the real world.

In this article, we show how to derive additional cardinality restrictions from a given configuration model. We assume the configurator knowledge base to be defined in a UML-like object-oriented formalism, i.e. component types correspond to classes and port-to-port connections to associations.

Since we focus on UML, the basis of our analysis are association multiplicities and additional information (e.g. singletons) found in UML class diagrams. This information is translated to linear inequalities, thereby generating optimization problems whose solutions correspond to the sought-after cardinality bounds.

Furthermore, we address *association specialization* (i.e. subset relations between associations) and necessarily empty ("zero-zero") associations which have not been studied before. We show that specialized associations increase the expressive power of UML class diagrams in terms of tightening bounds on class cardinalities. Association specialization is frequently used in product configuration to specify cardinalities of specific types. For example: Vehicles can have any number of wheels, but cars have 4 wheels, bicycles have 2 wheels, etc.

This paper is organized as follows: First, we introduce a running example in Section 2. In Section 3 we propose a formal description of a subset of the language of UML class diagrams and explain how to extract linear inequalities from them, including our new inequalities for association specialization. In Section 4, we show how to compute class cardinality bounds from the generated inequalities and how to utilize this knowledge in automatic configurators. After an evaluation in Section 5 and addressing related work in Section 6, the article is concluded with a discussion in Section 7.

2 MOTIVATING EXAMPLE

The UML [21] class diagram in Fig. 1 constitutes a configuration problem that shall serve as our running example.

The class diagram represents a company whose network infrastructure and real estate we want to manage. It consists of eight classes and various associations between them. The configuration entry class, of which exactly one instance exists, is *Company*. The network consists of up to 100 *Device*s which are either *PC*s or *Printer*s. Each PC must have a default printer. The company's real estate consists of up to ten

[1] Siemens AG Österreich, Corporate Technology, Vienna, Austria
firstname(.middleinitial).lastname@siemens.com

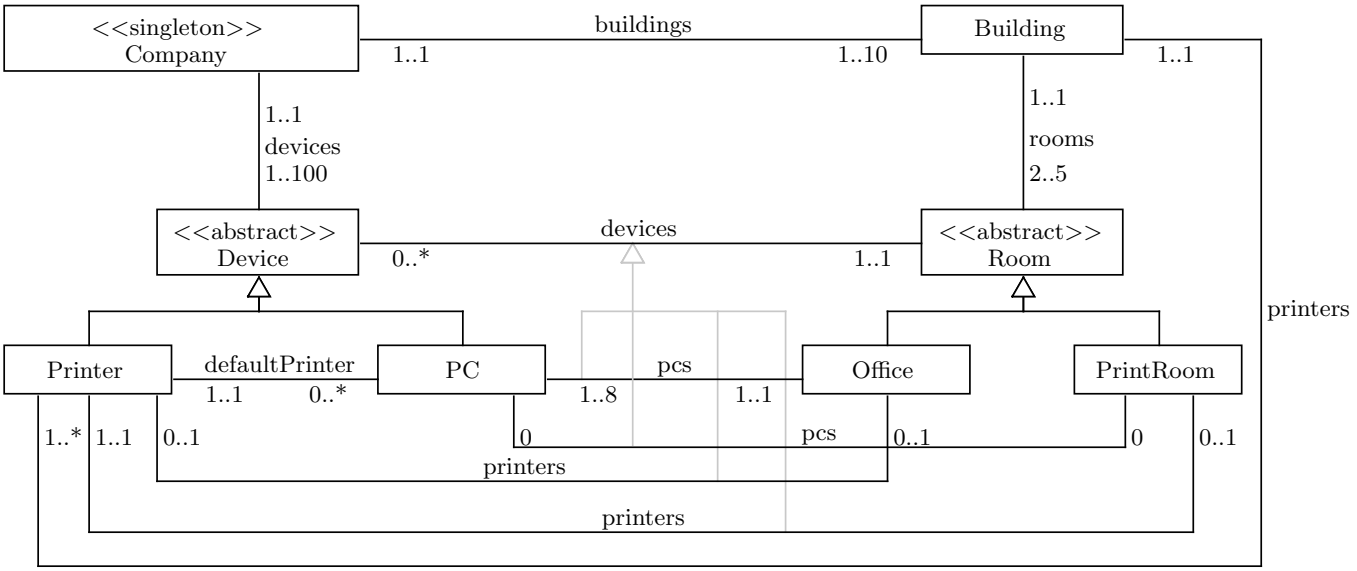


Figure 1. Running example: Network and real estate of a company (UML class diagram)

*Building*s, each of which has between two and five *Room*s. A room is either an *Office* or a *PrintRoom*. Each device knows which room it is in. Each room contains an arbitrary number of devices, but this number is constrained in the subclasses: Each office contains between one and eight PCs and at most one printer, while a print room contains one printer but no other devices. To guarantee easy access to printers for the employees of our company, we require at least one per building.

The generalization arrows drawn between associations realize the UML concept of association specialization [10], which will be formalized in Section 3. Intuitively, this means that the set of *PC*s in an *Office* is a subset of the set of *Device*s in the same *Office* etc.

In the remainder of the article, class names will often be abbreviated by only their upper-case letters.

Usually, configuration problems contain additional domain-specific constraints. Such constraints, which are not shown here due to lack of space, might enforce that the *Printer*s associated with a *Building* are the same ones that are associated with the building's *Room*s, or that a *PC* and its default *Printer* must be in the same *Building*.

A solution to a configuration problem constitutes a *configuration*, i.e. an instantiation of the class diagram that satisfies all constraints (the domain-specific ones as well as the cardinality constraints on which we concentrate from now on). Now imagine that, while searching for such a configuration, one instance of *PC* is created. Because there exists an implicit constraint that it must be associated with its default *Printer*, the question arises whether an existing printer should be used or a new one should be created. If the maximum number of devices in the company's network is already reached or there is a printer in the computer's room, no new printer should be instantiated. This is obvious to a human (at least in this toy example), but not necessarily to an automatic solver.

3 CARDINALITY CONSTRAINTS IN CLASS DIAGRAMS

To determine lower and upper bounds of class cardinalities in UML class diagrams, we extract inequalities from them to generate integer linear programs (ILPs). Solving those leads to the desired results.

This approach was first used with Entity-Relationship diagrams [9, 17] and later adapted for object-oriented models such as UML class diagrams [4]. We build upon the approach for UML class diagrams, because this language is the de-facto standard for describing object-oriented systems in general and configuration problems in particular [15].

A UML class diagram[2] consists of classes, binary associations and integrity constraints. A class represents a set of individuals, called the instances of the class, sharing common properties. Associations are relationships between classes. We support the following integrity constraints:

- *Is-a constraints* (class hierarchy constraints) between classes force the set of instances of one class to be a subset of the instances of another class.
- *Subset constraints* between associations force the pairs of class instances in one association to be a subset of the pairs of class instances in another association. This realizes the UML concept of association specialization [10].
- *Cardinality constraints* (multiplicities) defined on each side of an association restrict the number of class instances involved in the association.

We now define these notions more formally.

[2] We define only a subset of the language as far as needed for our purposes. Our definition of a class diagram is inspired by the definition of an ER-diagram in [9]. A different set-theoretic definition of class diagrams is provided by [5].

Definition 1 *For our purposes, a **class diagram** CD is a tuple* $(\mathcal{C}, \preceq_c, \mathcal{A}, \preceq_\mathcal{A}, lb, ub)$*, where*

- $\mathcal{C}$ *is a set of class symbols,*
- $\preceq_\mathcal{C} \subseteq \mathcal{C} \times \mathcal{C}$ *is a non-circular binary relation between class symbols representing the set of is-a constraints, the reflexive transitive closure*[3] *of which is denoted by* $\preceq^*_\mathcal{C}$*,*
- $\mathcal{A} \subseteq \mathcal{C} \times \mathcal{AS} \times \mathcal{C}$ *is a set of bidirectional associations between two classes (*$\mathcal{AS}$ *being a set of valid association symbols, e.g. the set of all strings over a given alphabet),*
- $\preceq_\mathcal{A} \subseteq \mathcal{A} \times \mathcal{A}$ *is a non-circular binary relation between associations representing the set of subset constraints between associations, the reflexive transitive closure*[3] *of which is denoted by* $\preceq^*_\mathcal{A}$*,*
- *lb:* $(\mathcal{A} \times \{1, 2\}) \to \mathbb{N}_0$ *is a total function mapping each end (1 and 2) of an association to the lower bound on its multiplicity, which is a natural number, and*
- *ub:* $(\mathcal{A} \times \{1, 2\}) \to \mathbb{N}_0 \cup \{*\}$ *is a total function mapping each end of an association to the upper bound on its multiplicity, which is a natural number or the asterisk.*

Additionally, we define the following auxiliary functions:

- *For all classes* $C \in \mathcal{C}$*, their set of direct descendants is denoted by* $children(C) = \{C' \in \mathcal{C} \mid C' \preceq_\mathcal{C} C\}$*.*
- *For all classes* $C \in \mathcal{C}$*, the set including them and all their descendants is called* $subfamily(C) = \{C' \in \mathcal{C} \mid C' \preceq^*_\mathcal{C} C\}$*.*
- *For all classes* $C \in \mathcal{C}$*, the set of leaves of the generalization tree rooting at* C *is denoted by* $leaves(C) = \{C' \in subfamily(C) \mid children(C') = \emptyset\}$*.*
- *For all associations* $A = (C_1, AS, C_2) \in \mathcal{A}$*, we define the two class access functions* $c_1(A) = C_1$ *and* $c_2(A) = C_2$*.*

To illustrate Definition 1, we apply these concepts to a subset of our running example (cf. Fig. 1):

$$\mathcal{C} = \{C, D, P, PC, B, R, O, PR\}$$
$$P \preceq_\mathcal{C} D, PC \preceq_\mathcal{C} D, O \preceq_\mathcal{C} R, PR \preceq_\mathcal{C} R$$
$$\mathcal{A} = \{(R, devices, D), (O, pcs, PC), \dots\}$$
$$(O, pcs, PC) \preceq_\mathcal{A} (R, devices, D), \dots$$
$$lb((R, devices, D), 1) = 1, ub((R, devices, D), 1) = 1,$$
$$lb((R, devices, D), 2) = 0, ub((R, devices, D), 2) = *, \dots$$

An instance of a class diagram is a finite collection of instances of the involved classes and associations that satisfies the integrity constraints inherent in the diagram. Formally, this is based on the notion of interpretation.

Definition 2 *An **interpretation** $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ of a class diagram CD consists of a set $\Delta^\mathcal{I}$ (the domain of $\mathcal{I}$) and a function $\cdot^\mathcal{I}$ (the interpretation function of $\mathcal{I}$), the latter of which maps*

- *every class* $C \in \mathcal{C}$ *to a subset* $C^\mathcal{I}$ *of* $\Delta^\mathcal{I}$ *and*
- *every association* $A \in \mathcal{A}$ *to a subset* $A^\mathcal{I}$ *of* $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$*.*

[3] The reflexive transitive closure R^* of a relation $R \subseteq S \times S$ is the subset-minimal relation fulfilling the following three conditions:

1. $R \subseteq R^* \subseteq S \times S$
2. R^* is reflexive (i.e. $\forall s \in S\colon sR^*s$)
3. R^* is transitive (i.e. $\forall a, b, c \in S : (aR^*b \wedge bR^*c) \Rightarrow aR^*c$).

The elements of $C^\mathcal{I}$ and $A^\mathcal{I}$ are called *instances* of C and A, respectively.

Definition 3 *An interpretation $\mathcal{I}$ of a class diagram CD is said to be a **model** of CD if it satisfies the following conditions:*

- *For all* $C_1, C_2 \in \mathcal{C}$ *it holds that* $C_1 \preceq_\mathcal{C} C_2 \implies C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$*.*
- *For each association* $A = (C_1, AS, C_2) \in \mathcal{A}$ *it holds that* $A^\mathcal{I} \subseteq C_1^\mathcal{I} \times C_2^\mathcal{I}$*.*
- *For all* $A_1, A_2 \in \mathcal{A}$ *it holds that* $A_1 \preceq_\mathcal{A} A_2 \implies A_1^\mathcal{I} \subseteq A_2^\mathcal{I}$*.*
- *For each association* $A = (C_1, AS, C_2) \in \mathcal{A}$ *it holds that its cardinality constraints are respected, i.e.*
  - $\forall c_1 \in C_1\colon lb(A, 2) \le |\{(c_1, c_2) \in A^\mathcal{I} \mid c_2 \in C_2^\mathcal{I}\}| \le ub(A, 2)$ *and*
  - $\forall c_2 \in C_2\colon lb(A, 1) \le |\{(c_1, c_2) \in A^\mathcal{I} \mid c_1 \in C_1^\mathcal{I}\}| \le ub(A, 1)$*,*

  where $|\cdot|$ *denotes the cardinality of a set and* $n \le *$ *is trivially satisfied for every* $n \in \mathbb{N}_0$*.*

In the remainder of this paper, we assume all interpretations to fulfill these conditions.

3.1 Extracting inequalities from class diagrams

The original motivation to derive inequality systems from class diagrams was to check their correctness. In this sense, a database schema without is-a constraints is *strongly satisfiable* if each of its classes is non-empty in at least one instance of the schema [14, 17]. In the presence of is-a constraints, a schema is *satisfiable* if it has a non-empty model, and it is called *finitely satisfiable* if it has a finite model. The latter property is more relevant in practice because both in databases and in knowledge representation we are interested in finite models only [9].

Regarding UML class hierarchy concepts (cf. [4]), we restrict generalization sets to be *complete* and *disjoint*, because these properties are required in all configuration problems that we see in practice[4]. A generalization is complete if every instance of the superclass belongs to at least one subclass; and it is disjoint if the instance sets of the subclasses are disjoint. In Fig. 1, this means that every *Device* is a *PC* or a *Printer*[5] but not both, with similar constraints for *Rooms*. More formally, a generalization with root $C \in \mathcal{C}$ is complete in an interpretation $\mathcal{I}$ if $C^\mathcal{I} = \bigcup_{C' \in children(C)} C'^\mathcal{I}$ and it is disjoint if for all $C_1, C_2 \in children(C)$ it holds that $C_1 \neq C_2 \implies C_1^\mathcal{I} \cap C_2^\mathcal{I} = \emptyset$.

Also, we admit only *single inheritance* (in contrast to multiple inheritance), i.e. $C \preceq_\mathcal{C} P_1 \wedge C \preceq_\mathcal{C} P_2 \implies P_1 = P_2$.

Associations are restricted to be *unique*, which means that the same pair of objects cannot occur multiple times in the same association [11, 14]. This is trivially satisfied in our case due to the definition of association mappings in Definition 2.

For each class C, we have a variable $|C|$ which denotes the class' *cardinality*, i.e. $|C|$ is a shorthand for $|C^\mathcal{I}|$ in any

[4] Where completeness does not hold, it can easily be established by introducing an additional subclass.

[5] This is additionally represented by the <<abstract>> stereotype on the *Device* class.

interpretation $\mathcal{I}$. Each variable's domain is the set of non-negative integers, i.e. $|C| \in \mathbb{N}_0$ for all $C \in \mathcal{C}$.

The authors of [11, 14] describe a method to translate associations from UML class diagrams to linear inequations[6].

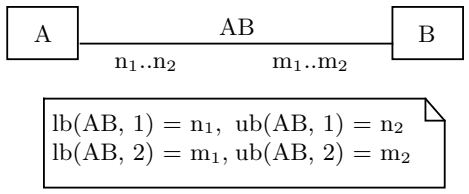


Figure 2. A binary association with multiplicities

In the small class diagram shown in Fig. 2, two classes A and B are associated with each other by association AB. Each instance of A is associated with at least m_1 and at most m_2 instances of B and each instance of B is associated with at least n_1 and at most n_2 instances of A. Based on the assumptions above, the inequalities in Theorem 1 are generated.

Theorem 1 (proven in [14][7])

$$lb(AB,2) \cdot |A| \leq ub(AB,1) \cdot |B| \quad (1)$$
$$lb(AB,1) \cdot |B| \leq ub(AB,2) \cdot |A| \quad (2)$$
$$|A| > 0 \implies |B| \geq lb(AB,2) \quad (3)$$
$$|B| > 0 \implies |A| \geq lb(AB,1) \quad (4)$$

For $(PC, pcs, O) \in \mathcal{A}$ from our running example, we obtain the following inequalities:

$$1 \cdot |PC| \leq 8 \cdot |O|$$
$$1 \cdot |O| \leq 1 \cdot |PC|$$
$$|PC| > 0 \implies |O| \geq 1$$
$$|O| > 0 \implies |PC| \geq 1$$

Inequalities (3) and (4) are so-called conditional inequalities, i.e. the inequality to the right of the implication is only required to hold if the condition to the left of the implication holds. In [11], the generated inequalities are used to detect inconsistencies as well as to construct minimal configurations. CLEWS[8] is a tool that implements this approach [20].

Object-oriented models typically allow classes to be organized in hierarchies based on a containment relation (is-a constraints). The presence of such constraints makes reasoning about a schema much harder [9].

The authors of [4] also process class diagrams and suggest a translation of is-a constraints to associations with multiplicity 1 at the parent class and 0..1 at the child class. Additionally, they propose adding a constraint stating that the number of instances of the parent class must be equal to the sum of instances of the child classes in case the hierarchy is disjoint and complete. This amounts to the following equation for general class hierarchies.

[6] However, slightly different notations are used in both articles.
[7] (1) and (2) correspond to $My \geq nx$ and $Nx \geq my$ and (3) and (4) correspond to $xy \geq nx$ and $xy \geq my$ in Theorem 9 in [14].
[8] `http://logic.at/clews/`

Theorem 2

$$|P| = \sum_{C \in children(P)} |C|, \quad \textit{for all } P \in \mathcal{C} \textit{ where } children(P) \neq \emptyset \quad (5)$$

Proof 1 *Equation* (5) *is a direct consequence of the assumption that class hierarchies are complete and disjoint.* ■

Applying this to our running example in Fig. 1, we obtain the following additional equations:

$$|D| = |PC| + |P|, \qquad |R| = |PR| + |O|$$

3.2 Increasing expressiveness through association specialization

To ease the modeling of different multiplicities and partner types for specialized associations in subclasses, our approach supports association specialization. We use the class diagram depicted in Fig. 3 to explain the general setting. The UML feature of association specialization[9] [10, 21] is used to describe *subset* constraints between associations. This means that the association AB' between A' and B', which are (direct or indirect) descendants of A and B, specializes the association AB between A and B, i.e. $AB' \preceq_{\mathcal{A}} AB$. This, in turn, means that every instance of AB' is also an instance of AB. Of course, class hierarchies can be of arbitrary breadth and depth. It is allowed that $subfamily(A) = \{A\}$ and/or $A' = A$ (or similarly with B).

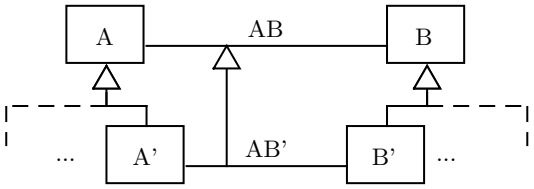


Figure 3. A class hierarchy featuring association specialization

As a special case, we admit so called "zero-zero" associations whose interpretation necessarily needs to be empty. This is used to disallow certain combinations of classes to be associated with each other[10]. Such an association exists between PC and $PrintRoom$, i.e. no print room contains a PC.

Because of the assumptions we make on class diagrams, only leaf classes can be instantiated. Therefore, we "propagate" associations for which no specialization exists down to the leaf classes. To define the notion of association specialization more formally, we introduce the concept of *leaf associations.*

Definition 4 *For an association $A \in \mathcal{A}$, the set A^* defines the* ***leaf associations*** *of A. For $A = (C_1, AS, C_2)$, $A^* =$*

[9] UML provides two additional concepts which are similar to association specialization, called *association subsetting* and *association redefinition* [10]. The subtle differences between these constructs and association specialization are out of scope of this article.
[10] By that, zero-zero associations have the same effect as cross-tree constraints of the *excludes* type in Feature Models [6].

$leaves(C_1) \times \{AS\} \times leaves(C_2)$, i.e. all possible associations between a leaf of C_1 and a leaf of C_2. The definitions of lb and ub must be extended accordingly for associations $A' \in A^*$ that are not explicitly defined in $\mathcal{A}$:

- *The lower bounds of such associations must be zero, because we cannot make any claims on the minimal cardinality of subclasses which are not part of an association: $lb(A', 1) = lb(A', 2) = 0$.*
- *The upper bounds, on the other hand, can be inherited from an association on the path to the root of the class hierarchy, because they are valid for all descendants of their defining class.*
  More formally: For a class $C_1^{A'} \in \{c_1(A') \mid A' \in A^*\}$ that does not occur in an association specializing A ($\nexists A' \in \mathcal{A}$: $A' \preceq^*_{\mathcal{A}} A \wedge c_1(A') = C_1^{A'}$), we define the nearest definition of A as $nearest(A, C_1^{A'}) := A' \in \mathcal{A}$ s.t. $A' \preceq^*_{\mathcal{A}} A$ and there is no A'' in-between ($\nexists A'' \in \mathcal{A}$: $A' \neq A'' \wedge A'' \preceq^*_{\mathcal{A}} A \wedge C_1^{A'} \preceq^*_{\mathcal{C}} c_1(A'') \preceq^*_{\mathcal{C}} c_1(A')$). Then, $ub(A', 1) = ub(\bar{A}'_1, 1)$ where $\bar{A}'_1 = nearest(A', c_1(A'))$ (and accordingly for C_2).

We use this to derive the additional inequalities in Theorems 3 and 4.

Theorem 3 *Using the symbol AB' as an abbreviation for the triple $(A', AS, B') \in AB^*$ and the symbol $\sum$ as an abbreviation for $\sum_{AB' \in AB^*}$, the following additional inequations hold:*

$$lb(AB,1) \cdot |B| \leq \sum ub(AB',1) \cdot |B'| \quad (6)$$

$$lb(AB,1) \cdot |B| \leq \sum ub(AB',2) \cdot |A'| \quad (7)$$

$$lb(AB,2) \cdot |A| \leq \sum ub(AB',1) \cdot |B'| \quad (8)$$

$$lb(AB,2) \cdot |A| \leq \sum ub(AB',2) \cdot |A'| \quad (9)$$

$$\sum lb(AB',1) \cdot |B'| \leq ub(AB,1) \cdot |B| \quad (10)$$

$$\sum lb(AB',1) \cdot |B'| \leq ub(AB,2) \cdot |A| \quad (11)$$

$$\sum lb(AB',2) \cdot |A'| \leq ub(AB,1) \cdot |B| \quad (12)$$

$$\sum lb(AB',2) \cdot |A'| \leq ub(AB,2) \cdot |A| \quad (13)$$

Proof 2 *For every instance of A, there are at least $lb(AB, 2)$ different tuples in the association, i.e. $lb(AB, 2) \cdot |A| \leq |AB|$. For every leaf association $AB' \in AB^*$, similar inequalities can be constructed, e.g. $|AB'| \leq ub(AB', 1) \cdot |B'|$. This can be summed up over all leaf associations, i.e. $\sum |AB'| \leq \sum ub(AB', 1) \cdot |B'|$. Because the set of (interpreted) leaf associations defines a partition over the interpreted root association, it holds that $|AB| = \sum |AB'|$, which proves (8). The other inequations can be proven analogously.* ■

Intuitively, (8) and (12) correspond to (1), and (7) and (11) correspond to (2), where one side of the inequality is replaced by a sum over an entire class hierarchy, respectively. In other words: (8), for example, states that there cannot be more instances of A than can be associated with instances of leaf classes of B; and (6) limits instances of B by the number of times that its leaf classes can be associated with instances of A's leaf classes.

The two conditional inequalities (3) and (4) could also be adapted for association specialization, but this does not lead to any additional information.

Theorem 4 *For all $A' \in subfamily(A)$, the following additional inequations hold, where $A'B^*$ stands for $\{AB' \in AB^* \mid c_1(AB') = A'\}$:*

$$lb(AB,2) \cdot |A'| \leq \max_{AB' \in A'B^*} \big(ub(AB',1)\big) \cdot |B| \quad (14)$$

$$\min_{AB' \in A'B^*} \big(lb(AB',1)\big) \cdot |B| \leq ub(AB,2) \cdot |A'| \quad (15)$$

For all $B' \in subfamily(B)$, the following additional inequations hold, where AB'^* stands for $\{AB' \in AB^* \mid c_2(AB') = B'\}$:

$$lb(AB,1) \cdot |B'| \leq \max_{AB' \in AB'^*} \big(ub(AB',2)\big) \cdot |A| \quad (16)$$

$$\min_{AB' \in AB'^*} \big(lb(AB',2)\big) \cdot |A| \leq ub(AB,1) \cdot |B'| \quad (17)$$

Proof 3 *From the proof for Theorem 1, we know that $lb(AB, 2) \cdot |A| \leq |AB|$ holds. For any $A' \in subfamily(A)$ it holds that $|A'| \leq |A|$, thus $lb(AB, 2) \cdot |A'| \leq |AB|$ holds. On the other hand, A cannot be associated with a higher number of Bs than any of its leaves, so $|AB| \leq \max_{AB' \in A'B^*} (ub(AB', 1)) \cdot |B|$. Combining these findings, we prove (14). Inequations (15) to (17) can be proven analogously.* ■

Applying Theorems 3 and 4 to our running example[11], we obtain the following additional inequations[12]:

$$1 \cdot |D| \leq 2 \cdot |P| + 1 \cdot |PC| \quad (6)$$
$$1 \cdot |D| \leq 9 \cdot |O| + 1 \cdot |PR| \quad (7)$$
$$0 \cdot |P| + 1 \cdot |PC| \leq 1 \cdot |D| \quad (10)$$
$$1 \cdot |O| + 1 \cdot |PR| \leq 1 \cdot |D| \quad (12)$$
$$1 \cdot |P| \leq \max(1,1) \cdot |R| \quad (14)$$
$$\min(0,1) \cdot |R| \leq 1 \cdot |P| \quad (15)$$
$$1 \cdot |PC| \leq \max(0,8) \cdot |R| \quad (14)$$
$$\min(0,1) \cdot |R| \leq 1 \cdot |PC| \quad (15)$$

4 DETERMINING AND UTILIZING CLASS CARDINALITY BOUNDS

Having defined all inequalities that are extracted from class diagrams, we now describe how they are used to find lower and upper bounds for cardinalities of classes.

4.1 Generating and solving linear programs

As already mentioned, some information on the cardinalities must be given. Otherwise, not much information on lower and upper bounds can be obtained from the linear program. For example, configuration problems are often constructed to have a root class of which exactly one instance exists, or the user could provide more information, e. g. that at most two hard drives are allowed in a computer configuration. Such information is translated to additional (in)equalities.

[11] Inequations (8), (9), (11), (13), (16) and (17) are omitted here, because they cannot contribute anything to cardinality bounds due to the unboundedness of *r_devices* near D. For Theorem 4, only the inequations for $A' \in \{P, PC\}$ are given.

[12] The inequations in the example have already been simplified. The full version of (6) reads: $1 \cdot |D| \leq 1 \cdot |P| + 1 \cdot |PC| + 1 \cdot |P| + 0 \cdot |PC|$.

Table 1. Computed cardinality bounds for the running example

	C	D	P	PC	B	R	O	PR
Associations	0..∞	0..∞	0..∞	0..∞	0..∞	0..∞	0..∞	0..∞
There is only one company	**1..1**	**1..100**	**1**..∞	0..∞	**1..10**	**2..50**	0..∞	0..∞
Generalizations	1..1	**2**..100	1..**100**	0..**97**	1..10	2..50	0..**50**	0..**50**
Association specialization	1..1	2..100	1..**50**	0..97	1..10	2..50	0..50	0..50
Fixed B, PR, O	1..1	**40**..100	**10..40**	**30..90**	**10..10**	**40..40**	**30..30**	**10..10**

To calculate lower and upper bounds of individual classes' cardinalities, we generate two optimization problems per class: One maximizes the value of the variable representing the cardinality of the class, the other minimizes it. In this step, an external ILP solver is employed to solve the generated linear program. We used *lp_solve*[13] and *SCIP*[14] [1] interchangeably in our experiments.

The conditional constraints (3) and (4) make it possible to derive further inequalities after an optimization problem is solved. Because of this, we solve each problem iteratively until no further inequalities are added.

Algorithm 1 presents procedure ComputeBounds that computes the lower and the upper cardinality bounds of a single class. It calls ComputeBound twice, once for each bound. The conditional inequations (3) and (4) are generated in line 6, and all unconditional inequalities in line 7. Then, iteratively, the linear program is either minimized or maximized by the external ILP solver in line 10.

After being solved, the linear program is extended by the conditional inequalities whose premises are satisfied in line 11. When no new conditional inequations can be satisfied[15], the value corresponding to the relevant class is extracted from the solution and returned in line 13.

Algorithm 1 Cardinality bounds computation for one class

```
1:  procedure ComputeBounds(CD, C ∈ 𝒞)
2:      min ← ComputeBound(CD, "minimize", C)
3:      max ← ComputeBound(CD, "maximize", C)
4:      return (min, max)
5:  procedure ComputeBound(CD, optimization, C ∈ 𝒞)
6:      CI ← ExtractConditionalInequations(CD)
7:      I ← ExtractUnconditionalInequations(CD)
8:      repeat
9:          I′ ← I
10:         S ← Solve(I′, optimization, C)
11:         I ← I′ ∪ GetSatisfiedCondIneqs(CI, S)
12:     until I′ = I
13:     return ExtractValue(S, C)
```

Table 1 shows the computed cardinality bounds for our running example (see Fig. 1). From the top to the bottom of the table, information is successively added to tighten the cardinality bounds: In the row labeled *Associations*, (1) to (4) are generated for all associations, which gives us no bounds at all. In the next row, there is one and only one company, which makes all bounds tighter except for classes *PC*, *PR* and *O*.

[13] `http://lpsolve.sourceforge.net/5.5/`
[14] `http://scip.zib.de/`
[15] Algorithm 1 is guaranteed to terminate because *CI* is finite and thus *I* will eventually reach a fixpoint.

Next, the constraint for complete and disjoint generalizations (5) is added, which leads to several interesting new bounds. Because we know that there are at least two rooms and that every room must contain at least one device, the lower bound for *D* becomes 2. The *PC* case is a bit more complex: In case there were more than 80 *PC*s, we would need at least 3 buildings to accommodate them. Since every building needs a printer, 3 of the maximum number of 100 devices must be printers, so there can be at most 97 *PC*s.

When considering the constraints generated for association specialization (cf. Theorems 3 and 4), we additionally learn that there cannot be more printers than rooms (because every type of room contains at most one printer).

Assuming that the physical real estate of the company is fixed, we incorporate in the last row the user input that there are exactly ten buildings, ten print rooms and 30 offices. Thereby, all bounds are tightened even further.

4.2 Integrating the presented approach in automatic configurators

We have integrated the techniques described so far in S'UPREME, a domain-independent framework for product configurators that is developed and used internally within Siemens [16]. S'UPREME is a (semi-)automatic configurator, i.e. it can be used by alternating manual and automated step sequences.

In automatic mode, classes of the configuration model are instantiated when an association contains too few instances, i. e. when the implicit cardinality constraint attached to an association is violated.

During manual configuration, a user can feed additional knowledge about the structure of the problem to the solver. This can be used to further constrain the cardinalities in the knowledge base as shown in the example at the end of Table 1.

Other solvers can benefit as well by integrating the presented techniques for computing cardinality bounds during knowledge base design. Most of the information required to construct the linear (in)equalities presented above is contained in the class diagram itself: It lies in the associations' cardinality constraints. Most models, however, still admit infinite instantiations. This is the reason why we need upper bounds on the cardinalities of at least one class to be able to compute upper bounds for other classes from the knowledge base alone. In configuration problems, there often exists a root class whose cardinality is exactly one, which can already be used as a means to this end.

Computing cardinalities already at knowledge base design time has two advantages: First, they can be used to statically check the consistency of the knowledge base (i.e. the domain model, consisting of the class diagram and the ad-

Table 2. Number of failures encountered until solution was found (maximum number of objects = 50); CC=with cardinality constraints, D=Device, P=Printer, PC=PC, B=Building, R=Room, O=Office, PR=PrintRoom

Fixed	D		P		PC		B		R		O		PR	
objects		CC		CC		CC		CC		CC		CC		CC
1	*62	*1	48	2	4	1	49	2	*171	*1	5	0	6	0
2	2	1	2	2	4	0	719,982	2	6	2	7	2	8	0
3	4	1	4	0	4	0	-	3	25	6	78	7	21	0
4	10	0	10	0	4	0	-	4	404	3	147	20	46	0
5	34	0	34	0	4	0	-	5	8,788	4	179	94	52	0
6	189	0	-	0	4	0	-	6	1,432,315	4	11,746	700	1,422	0
7	188	0	-	0	4	0	-	7	8,976,565	5	63,183	4,730	5,653	0
8	187	0	-	0	4	0	-	8	-	6	641,246	44,236	34,285	0
9	186	0	-	0	22	0	-	9	-	7	7,555,163	452,634	260,133	0
10	185	0	-	0	19	0	-	*1	-	8	-	5,323,734	2,311,113	0

ditional domain-specific constraints). Second, the computed cardinalities can later be utilized by a solver without needing any additional computational efforts.

4.3 Integrating the presented approach in constraint solvers

An encoding of a configuration problem in a constraint language like MiniZinc[16] [19] can also be seen as an automatic configurator. Such an encoding needs to contain variables for each object that may potentially be created during solving, and it must distinguish between used and unused instances.

There are two ways for such encodings to profit from our approach: Either, the cardinality bounds are pre-computed as described in Section 4.1 and included in the encodings as constants; or the generated (in)equations are directly included in the encodings as constraints.

5 EVALUATION

For this article, we chose to feature the evaluation of the constraint solving approach presented in Section 4.3. The configuration problems were encoded in MiniZinc, using a generic object-oriented encoding[17]. To evaluate the effects of the additional cardinality constraints on MiniZinc models of a configuration problem, we generated a set of random test cases for several domains. Here, the results for our running example are presented. Each test case was executed twice: Once with the additional cardinality constraints, once without.

Table 2 shows the number of failures for the different test cases encountered by MiniZinc (with Gecode solver) until a solution is found. The maximum number of objects is 50 in all test cases, while in each test case the cardinality of a single class is fixed. This class and its cardinality are given in the first row resp. column. For the cases where no solution can be found within the time limit of 15 minutes, "-" is given instead of the number of backtracks. Unsatisfiable cases are marked with an asterisk. For example, finding a solution with exactly 5 printers (column PR, row 5) requires 0 backtracks when the additional cardinality constraints (CC) are present, but 52 when they are not.

[16] `http://www.minizinc.org`

[17] The MiniZinc models and data can be found at `https://github.com/siemens/OOCSP`.

Without using the generated inequalities, the test cases can only be solved with a substantial number of failures (backtracks) or cannot be solved at all within the given time frame (15 minutes). Running the same test cases with the inequalities, almost all test cases can be solved with very few failures within one minute.

Although these results were confirmed in further experiments with different domains, occasionally adding the inequalities deteriorates the solver performance. In these cases the additional inequalities lead the solver into a part of the search tree where no solution can be found.

6 RELATED WORK

Using linear inequalities to decide the satisfiability of object-oriented data models has a long history in software engineering and artificial intelligence [17]. Our approach for generating inequalities extends the work of [4, 5, 11, 14, 18].

In [7], the correctness of UML/OCL models is verified using constraint programming. An approach combining UML, DL (description logics) and constraint programming is described in [8].

A Logic for Configuration Problems, called LoCo, is introduced in [2]. The authors discuss a logic-based formalism to describe configuration problems. For this formalism, they introduce inequalities that are similar to those proposed by [11]. Additionally, they introduce a concept called "one-to-many axioms" which leads to similar inequalities as those presented in Theorem 3, but which do not support generalization hierarchies.

The authors of [13] describe a hybrid description logic [3] reasoner combining an Abox tableau calculus and integer linear programming. They achieved improvements, especially in cases where large numbers occur in number restrictions. As far as our experience goes, few DL reasoners are capable of efficiently dealing with number restrictions, which correspond to association multiplicities in UML. Like in the case of UML object models, the reason seems to be that few real-world ontologies use specific number restrictions and closed-world reasoning.

7 CONCLUSION

We have shown how linear inequalities can be extracted from UML class diagrams to derive lower and upper bounds for

class cardinalities. Compared to the inequalities known from earlier work, additional knowledge inherent in association specialization is exploited in our approach. This leads to the derivation of tighter cardinality bounds in domains with complex class hierarchies. Such constellations often occur in configuration problems, an example of which is the placement of hardware modules of various widths into hardware racks of various types [12].

The additionally derived information can be used by automatic configurators to increase both efficiency and user-friendliness. In this way, more configuration problems can be solved automatically, the number of necessary user decisions is reduced (by removing alternatives which do not lead to an acceptable solution), and the final configuration result is achieved faster. This holds especially for the configuration problems we have encoded in MiniZinc in our evaluation.

7.1 Future work

Even though we have successfully applied the approach described in this article, there are still some open questions. In particular, we will include approaches where cardinality bounds are precomputed (cf. Section 4) in our in-depth evaluation. Also, it will be interesting to thoroughly evaluate our approach on real-world examples we encounter in practice.

ACKNOWLEDGEMENTS

This work was funded by the Vienna Business Agency (Austria), in the programme ZIT13 plus, within the project COSIMO (Collaborative Configuration Systems Integration and Modeling) under grant number 967327; and by the Austrian Research Promotion Agency within the project HINT (Heuristic Intelligence) under grant number 840242. The authors thank Gernot Salzer and Alois Haselböck for their comments on previous versions of this article.

REFERENCES

[1] Tobias Achterberg, 'SCIP: Solving Constraint Integer Programs', *Mathematical Programming Computation*, **1**(1), 1–41, (2009).

[2] Markus Aschinger, Conrad Drescher, and Heribert Vollmer, 'LoCo - A Logic for Configuration Problems', in *ECAI 2012*, ed., Luc de Raedt, volume 242 of *Frontiers in artificial intelligence and applications, 0922-6389*, pp. 73–78, Amsterdam, (2012). IOS Press.

[3] Franz Baader, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[4] Mira Balaban and Azzam Maraee, 'Consistency of UML Class Diagrams with Hierarchy Constraints', in *Next Generation Information Technologies and Systems*, eds., Opher Etzion, Tsvi Kuflik, and Amihai Motro, volume 4032 of *Lecture Notes in Computer Science*, pp. 71–82, Berlin, Heidelberg, (2006). Springer.

[5] Mira Balaban and Azzam Maraee, 'Finite Satisfiability of UML Class Diagrams with Constrained Class Hierarchy', *ACM Transactions on Software Engineering and Methodology*, **22**(3), 24:1–24:42, (2013).

[6] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés, 'Automated Analysis of Feature Models 20 Years Later: A Literature Review', *Information Systems*, **35**(6), 615–636, (2010).

[7] Jordi Cabot, Robert Clarisó, and Daniel Riera, 'UMLtoCSP: a Tool for the Formal Verification of UML/OCL Models Using Constraint Programming', in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pp. 547–548, (2007).

[8] Marco Cadoli, Giuseppe de Giacomo, Toni Mancini, and Diego Calvanese, 'Finite Model Reasoning on UML Class Diagrams via Constraint Programming', in *AI*IA 2007 Artificial Intelligence and Human-Oriented Computing*, eds., Roberto Basili and Maria Teresa Pazienza, volume 4733 of *Lecture Notes in Computer Science*, pp. 36–47, Berlin, Heidelberg, (2007). Springer.

[9] Diego Calvanese and Maurizio Lenzerini, 'On the Interaction Between ISA and Cardinality Constraints', in *Tenth International Conference on Data Engineering*, pp. 204–213. IEEE Computer Society, (1994).

[10] Dolors Costal, Cristina Gómez, and Giancarlo Guizzardi, 'Formal Semantics and Ontological Analysis for Understanding Subsetting, Specialization and Redefinition of Associations in UML', in *Conceptual Modeling – ER 2011*, eds., Manfred Jeusfeld, Lois Delcambre, and Tok-Wang Ling, volume 6998 of *Lecture Notes in Computer Science*, pp. 189–203, Berlin, Heidelberg, (2011). Springer.

[11] Andreas A. Falkner, Ingo Feinerer, Gernot Salzer, and Gottfried Schenner, 'Computing product configurations via UML and integer linear programming', *International Journal of Mass Customisation*, **3**(4), 351–367, (2010).

[12] Andreas A. Falkner and Herwig Schreiner, 'Siemens: Configuration and Reconfiguration in Industry', in *Knowledge-based Configuration*, eds., Alexander Felfernig, Lothar Hotz, Claire Bagley, and Juha Tiihonen, 199–210, Morgan Kaufmann, Amsterdam, (2014).

[13] Nasim Farsiniamarj and Volker Haarslev, 'Practical Reasoning with Qualified Number Restrictions: A Hybrid Abox Calculus for the Description Logic H', *AI Communications*, **23**(2-3), 205–240, (2010).

[14] Ingo Feinerer and Gernot Salzer, 'Numeric semantics of class diagrams with multiplicity and uniqueness constraints', *Software & Systems Modeling*, **13**(3), 1167–1187, (2014).

[15] Alexander Felfernig, Gerhard E. Friedrich, and Dietmar Jannach, 'UML as Domain Specific Language for the Construction of Knowledge-based Configuration systems', *International Journal of Software Engineering and Knowledge Engineering*, **10**(04), 449–469, (2000).

[16] Alois Haselböck and Gottfried Schenner, 'S'UPREME', in *Knowledge-based Configuration*, eds., Alexander Felfernig, Lothar Hotz, Claire Bagley, and Juha Tiihonen, 263–269, Morgan Kaufmann, Amsterdam, (2014).

[17] Maurizio Lenzerini and Paolo Nobili, 'On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata', *Information Systems*, **15**(4), 453–461, (1990).

[18] Azzam Maraee and Mira Balaban, 'Efficient Reasoning About Finite Satisfiability of UML Class Diagrams with Constrained Generalization Sets', in *Model Driven Architecture – Foundations and Applications*, eds., David H. Akehurst, Régis Vogel, and Richard F. Paige, volume 4530 of *Lecture Notes in Computer Science*, pp. 17–31, Berlin, Heidelberg, (2007). Springer.

[19] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack, 'MiniZinc: Towards a Standard CP Modelling Language', in *Principles and Practice of Constraint Programming – CP 2007*, ed., Christian Bessiere, volume 4741 of *Lecture Notes in Computer Science*, pp. 23–27, Berlin, Heidelberg, (2007). Springer.

[20] Gerhard Niederbrucker, *A Numeric Semantics for UML Class Diagrams: Methods and Tools*, Diplomarbeit, Technische Universität Wien, Wien, 2010.

[21] Object Management Group. OMG Unified Modeling Language Version 2.5, March 2015.

[22] Markus Stumptner, Gerhard E. Friedrich, and Alois Haselböck, 'Generative Constraint-Based Configuration of Large Technical Systems', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **12**(4), 1–27, (1998).

Automatic Configuration of Hybrid Mathematical Models

Michael Barry, René Schumann
Smart Infrastructure Laboratory
HES-SO Valais / Wallis ,
Rue de Technopôle 3, 3960 Sierre, Switzerland
michael.barry@hevs.ch, rene.schumann@hevs.ch

Abstract.

In this paper we address the problems faced for automatic configuration of flexible hybrid models. We approach this problem with concepts from the computer science field including object orientated design and dependency injection to create a new Hybrid Model that combines traditional modelling methods with a flexible design. This type of model is able to drastically change its functional behaviour, allowing it to simulate a larger variety of scenarios of varying complexity. We then use AI methods to automatically configure the model to reduce its complexity to the minimum while having minimal impact on the models accuracy. A small example is demonstrated where this method is used to configure the market environment for a hydro-power plant model, allowing us to determine which set of markets are most profitable for any given plant configuration . Furthermore, the use of flexible hybrid models opens up the possibility for further AI methods to be used in conjunction with mathematical models.

1 Introduction

The operational planning of a hydro-power station is a complex, but well studied task. As hydro-power is a technology that has been available for a long time, the methods for using this technology are well refined. Operation research methods (2, 6) dominate this problem and are heavily used in industry. Such methods describe the world in a mathematical model, including a set of constraints and an objective function, and is then solved by a solver such as IBM's CPLEX (1). As these methods are so established in industry, alternative AI based methods struggle to have an impact.

However, we have identified an interesting field for AI methods in conjunction with existing mathematical methods. Therefore, we investigate the problem that arise in such models from a computer science point of view and consider these models to be a domain for our research.

We first focus on the problems that are prominent in mathematical methods. In many ways such models relate to legacy software or software that was developed when computer science was in its infancy. They are static, problem specific and extremely difficult to maintain. Although these problems are extremely prominent in industry, there is no easy fix. The root problem is inherent with the process that Operations Research uses to derive their solutions, which is shown in Fig. 1.

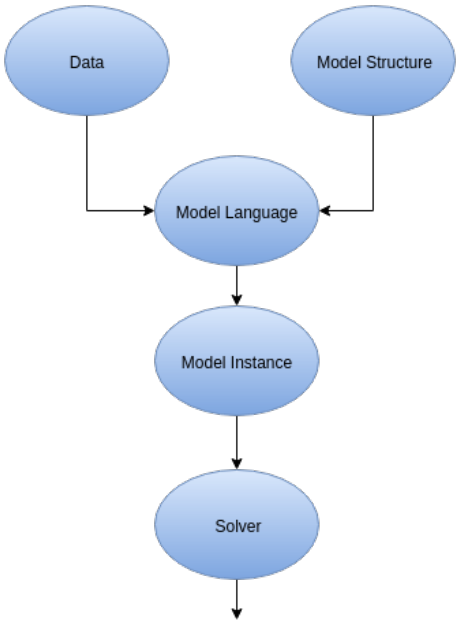


Figure 1. Graphical representation of the process used to create and solve mathematical models. It should be noted that all parts of the process is considered to be flexible except for the model structure.

The input and model structure is implemented using a model language, creating a model instance. This model instance can then me solved using a solver. This process does allow for flexibility, as the model structure is independent from the data and the solver. As a result, the data and solver can be changed. However, the model structure is considered to be static, never changing between model instances or during the development life cycle. This assumption limits the models functional flexibility.

Functional flexibility is the ability to change the behaviour of a module by physically altering the models functions themselves rather than only the input variables. Some degree of flexibility is achieved in current models using *IF* statements to physically modify a function based on a condition. Although widely used and acceptable on a small scale, it becomes chaotic for large models. Furthermore, flexible aspect must be implemented manually in every function that is

affected. This limitation is felt in the modelling world in two ways:

- Managing the life cycle of such models becomes difficult, often resulting in redevelopment of models rather than continues development.
- Scenarios requiring functional flexibility are either implemented with difficulty or are not investigated.

These problems are visible in the hydro-power industry due to the energy transition from fossil and nuclear power to renewable power. As the energy industry is in transition, models must adapt. However, this is a painful transition in the modelling world, as models in the industry struggle to adapt to new functionality and struggle to consider new scenarios which must be investigated. In many cases, models are redeveloped rather than updated, resulting in the loss of any still relevant knowledge or functionality retained within the outdated models.

In particular, we consider the configuration of the model to be a problem. Many models are unable to be configured to manage a large variety of scenarios. Previously, different scenarios were a matter of using different data, such as from another time period.

However, during the energy transition, investments in Hydropower are crucial. To accurately calculate the impact on the operation and profitability of a plant of an investment, models must be able to simulate a scenario where one aspect of a plant has been updated. This update can be in the form of a change to the topology, such as enlarging a water reservoir or building another conduit, to using a new technology, such as a new turbine with a different efficiency curve. However, current models are difficult to modify and therefore these scenarios become difficult to simulate. Each investment opportunity is extremely expensive to investigate, making combinations of investments a near impossible task.

In this paper we approach this configuration task by using concepts from the computer science community to introduce flexibility into mathematical models and use a configuration method to simulate a variety of scenarios. We describe in Section 1 the background and related literature, in Section 2 the configuration problem that we wish to solve, in Section 3 the method we use to solve the configuration problem, in Section 4 our results in a small example, in Section 5 our conclusions and in Section 6 the future research direction we will follow.

2 Background

Problems in the model development relating to maintainability and flexibility are well known. Such problem existed in the earlier stages of computer science, where software was developed in an extremely case sensitive and hardware dependant way. Similarly, modelling was originally extremely solver dependant. Software such as the General Algebraic Modeling System (GAMS) (4) allowed the solver to be separated from the modelling language, allowing a model to be tested with different solvers.

However, models are still extremely case sensitive and inflexible. Computer science overcame such obstacles with concepts such as object-orientated design and agile methodologies. Drawing of the success in computer science, these concept are being introduced to the modelling world. Many modern languages do incorporate an object orientated design (5). However, these languages are still at an infancy and are not widely used in industry. Tools and models that are based on more traditional modelling are more accepted within the modelling community.

Currently, there is a large emphasis on the development of hybrid models (3) within the operations research community. Hybrid models combine a top down model and a bottom up model. A top down model dissect a problem into several sub-problems, while a bottom up model works the other way around. Hybrid models open up the possibility of fusing a top down model design using traditional modelling languages with a bottom up approach that manages the object orientated design aspect. It allows the object orientated design to be abstracted from the model.

3 Problem

In this section we describe the problem we address in this paper by first describing the model used and then describing the configuration problem. We consider a model that optimises the operation of a hydro-power plant in terms of profitability with objective function Z and a set of constraints C as shown in Equation 1 to 2 . In addition, we have a set of scenarios S. Each scenario describes the environment that the model must simulate, such as what markets to trade on or weather to empty the reservoirs at the end of the time interval as shown in 3

$$\max. \sum_{i,m} P_{i,m} Q_{i,m} \tag{1}$$

$$C \begin{Bmatrix} Q_{i,m} = R_{i,m}\alpha \\ S_i = S_{i-1} + I_i - \sum_m R_{i,m} \\ S_i \leq S_{\max} \\ S_i \geq S_{\min} \\ \sum_m Q_{i,m} \leq Q_{max} \\ \dots \end{Bmatrix} \tag{2}$$

$$S \begin{Bmatrix} spot\_market = true, storage\_end = true... \\ spot\_market = false, storage\_end = true... \\ spot\_market = true, storage\_end = false... \\ \dots \end{Bmatrix} \tag{3}$$

Above, $P_{i,m}$ is the price at time interval i for market m, $Q_{i,m}$ is the produced energy for time interval i and market m, $R_{i,m}$ is the water released from the reservoir at time interval i for market m, α is the efficiency of the turbine (the amount of energy produced per water used), S_i is the storage level at time interval i, I_i is the inflow of water into the reservoir at time interval i, $S_{\max}$ is the maximum storage level of the reservoir, $S_{\min}$ is the minimum storage level of the reservoir and Q_{max} is the maximum production level.

Traditionally, constraints often contain conditionals such as in Equation 4 to modify their behaviour based on the scenarios requirements. Here, the conditional $water_loss$ is used to modify the equation defining the storage capacity to include a loss of water due to leaks or condensation.

$$S_i = S_{i-1} + I_i - \sum_m R_{i,m} - \$(water\_loss) L_i \tag{4}$$

This can also be used to enable / disable a constraint if it is required by the scenario as shown in Equation 5. Here, the boolean $storage_end$ is used to set a minimum water level at the final time interval.

$$\$(storage\_end) S_{i_{max}} = S_{end} \tag{5}$$

We reformulate the above to allow us to separate the models functionality from the conditionals by considering C to contain all forms of the constraints. For example, instead C containing a constraint in the form described in Equation 4, it exists as shown in Equation 6 and 7.

$$S_i = S_{i-1} + I_i - \sum_m R_{i,m} - L_i \tag{6}$$

$$S_i = S_{i-1} + I_i - \sum_m R_{i,m} \tag{7}$$

A model instance configured to match the functionality required by the scenario is then simply a subset of C.

However, for the model instance to be valid it must be able to compile into a meaningful model. Therefore, restrictions based on conflicting constraints and constraint interdependence from the top-down model must be considered.

In the traditional form, these restrictions are considered in the conditional statements. However, we separate these restrictions from the model functionality and describes them as a set of restrictions R that determine the validity of the model.

Therefore we define the problem as creating a model instance by selecting a subset C that contains the desired functionality of any scenario in S and can be combined into a valid model instance by respecting the restriction defined in R.

Next we describe a method that allows the conditional logic required by the scenarios to be separated from the mathematical model. Traditionally, the interaction of the mathematical model and the conditional logic is handled as shown in the two previous examples in Equation 6 and 7.

Much of the problems created by using such methods on a large model, which includes many functions, comes from having to define the conditional statement in each function that it affects. This can be avoided by grouping constraints that are limited by the same conditions into one module. Therefore, for example, constraints that describe a pump storage technology can be grouped together in a module and can be controlled with one conditional statement. In principle, this approach is similar to object-orientated design, that groups related functionality into a class. Similarly, different versions of a module may exist, each containing a different implementation of the contained constraints.

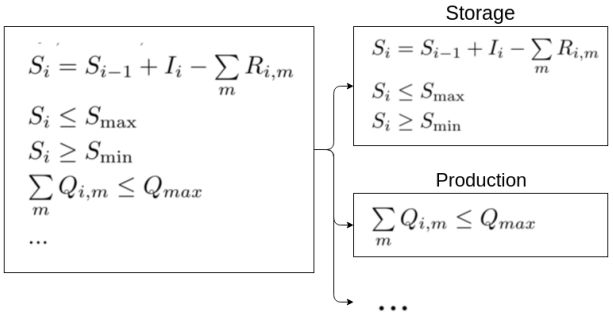


Figure 2. Graphical representation of how sets of related constraints are grouped together to form modules.

Therefore, the task can be simplified to selecting a set of modules to create a model instance. In essence, this method can be described as a form of dependency injection. A configuration file that describes a scenario can then simply contain import statements, that define which implementations to include in the model. Therefore we can create different configurations by selecting which modules to include for that configuration.

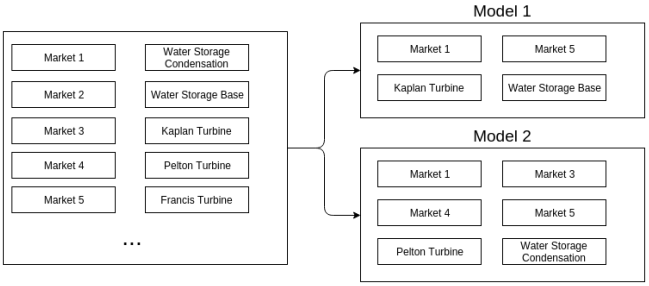


Figure 3. Graphical representation of how a set of modules is selected to form a configuration.

We then must consider the interdependency and conflicts that exist between modules. First, we focus on the interdependency of the modules. Since constraints relate to each other through shared variables, so do the modules. However, as modules relate to physical concepts such as a market or a type of turbine, it becomes intuitive. For example, it is impossible to create a module instance without a module describing the behaviour of the turbine or a market to sell the produced energy.

Therefore, we can create a dependency graph that shows how such modules relate. A small excerpt of such a graph is shown in Figure 4, showing the interdependency of the market modules and some of the storage modules.

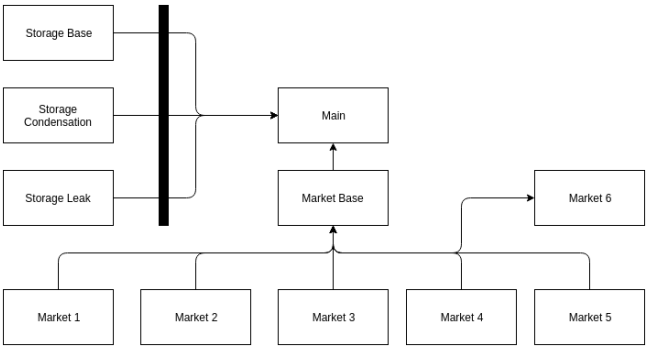


Figure 4. Graphical representation of how modules are dependent on each other. One and only one storage module can be selected at a time to avoid conflicts. Therefore the main module is dependent only one of the set of storage modules. However, for the market modules, one or all modules can be selected.

Modules can be dependent on a single module or on one of a set of modules as shown in 4.

We can then describe these dependencies with predicate logic. We use an XOR statement, relating to the XOR logic gate, to represent this relation as shown in Equation 8.

$$\oplus(Storage\_Base, \oplus(Storage\_Cond, Storage\_Leak)) \quad (8)$$

For simplicity, we extend the XOR relation to allow for multiple inputs as such a relation is synonymous to a conjunction of several classical XOR logic gates. Therefore, we can redefine Equation 8 as Equation 9.

$$\oplus(Storage\_Base, Storage\_Cond, Storage\_Leak) \quad (9)$$

We can then create scenarios by selecting which modules to be used in a configuration file and then check the modules validity with the logic described above. However, as scenarios also include data such as inflows, the topology of the hydro-power station and market prices, we must be able to incorporate such data into our module based approach.

For this purpose, we utilise the concept of instantiation. For example, a scenario may include a multi-site case, where several reservoirs, turbines, conduits, galleries, tailraces and penstalks exist. Reservoirs are used for storage, turbines generate the electricity, conduits deliver water into the hydro-power system, galleries are used to transport water from one component to another within the system, tailraces conduct the water coming from the turbines and penstocks conduct the water into the turbines. An example topology is given in Figure 5.

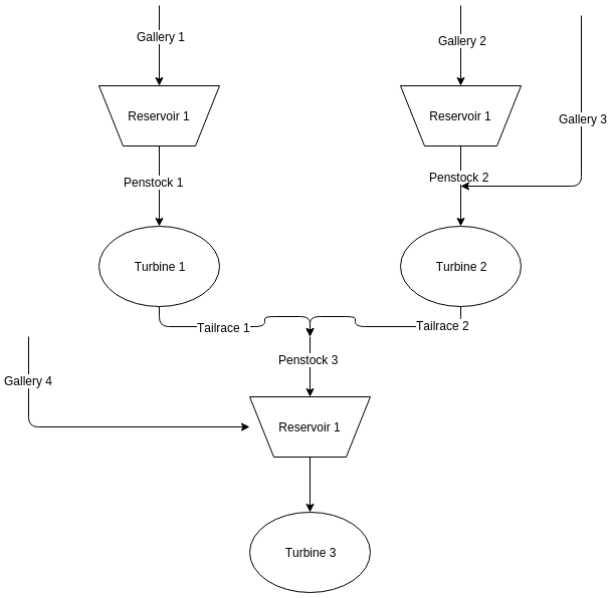


Figure 5. Graphical representation of the layout (topology) of an example Hydro-Power plant

To model such a scenario, we implement modules for each of these topology items and create multiple instances of each. These instances then have the specific parameter settings to represent a particular element of the system.

Although the model language does not support object orientated design or the concept of initialisation, it can still be easily implemented. In essence, an instance is simply a set of parameters and several instances can be simply described in a table. Therefore, the instantiation can be handled by the object-orientated part of the hybrid model and then communicated to the mathematical model as a set of input parameters.

The advantage of this approach is that further validation can be implemented by design. For example, a penstock can only connect to a turbine and a turbine can only connect to a tailrace. Furthermore, this modular approach also allows easy integration with other systems, such as an interface for creating a topology, a system for running systematic experiments on distributed systems or a system for visualising the results.

The resulting hybrid model is a highly flexible and can be validated easily. Configuration of small models is simple through interactive configuration as we have demonstrated in our previous work However, in large models, configuration becomes a complicated task and opens the possibility of using AI methods for automatic configuration. Some modules can be obsolete either as the underlying constraint does not restrict the search space or the module represents a functionality that is never used. Furthermore, some modules can be replaced by others that have a less complex implementation and therefore are easier to solve.

For example, in the hydro-power example, the model can be configured to simulate any given hydro-power plant. The model also includes the functionality to access multiple markets. However, not all hydro-power plants can access all markets due to technical limitations or simply because it is not profitable. Therefore, it makes sense to only include in the model instance the modules that contain the functionality for the appropriate markets and remove any modules representing a market that is never or rarely used. We use this example to demonstrate our method in Section 4.

Therefore, AI methods could be used for the configuration of flexible models to remove unnecessary constraints and reduce the time a solver requires to find a solution to the optimisation problem.

4 Method

In this section we describe a proof of concept method to solve the configuration problem. To be able to use a heuristic, we must first define two objectives:

- Minimise the model complexity
- Minimise deviation from Z

The first objective concerns the complexity that a solver requires to solve a configuration. Our previous work shows that there is, as one would expect, a direct relationship between the complexity of the model and its runtime. As a approximation of the models complexity, we simply use the number of modules.

The second objective takes into consideration the concept that mathematical models are considered to be inherently *wrong*, but some are accurate enough to be *useful*. It is based on the assumption that the more detailed the optimisation problem is described mathematically, the more *useful* is is. Therefore, we can assume that a model configuration that utilises the maximum amount and most detailed modules is the most useful answer to the Z function the model can produce. We wish to deviate from this result as little as possible to ensure that the model still produces useful answers. We use this answer as a reference point. The percentage of deviation as shown in Equation 10 is then calculated and used as an objective.

$$(Z_{ref} - Z_i)/Z_{ref} * 100 \quad (10)$$

We then use the Strength Pareto Evolutionary Algorithm (SPEA2) to solve for the above defined objectives. The SPEA2 algorithm is a

multi objective evolutionary algorithm that uses the concept of Pareto optimality and a clustering technique to iteratively evolve solutions towards the pareto front and then spread along it. Each configuration can be easily represented by a string of binary values, representing which modules are switched on or off. To initialise the algorithm, we use the configuration with the maximum amount of modules and its direct relatives.

The SPEA2 algorithm was chosen for several reasons. First, this is a computationally expensive process, as assessing the runtime and the deviation from Z involves actually solving the model instance. SPEA2 can be easily implemented for a distributed system, allowing each model instance to be solved on a separate machine and therefore speeding up the process considerably. Second, SPEA2 excels at finding surprising solutions which are either not intuitive or simply difficult to arrive to. For example, switching one module alone may not bring a benefit, but in conjunction with others might yield a surprisingly optimal configuration. Such configurations are difficult to derive, but evolutionary algorithms have generally been successful in such situations due to their random nature.

The algorithm produces the Pareto front between the complexity of the model configuration and the deviation from the most accurate result for Z. From this Pareto front it is then possible to select the configuration with an acceptable deviation and the lowest complexity. Although this process is computationally expensive, it only needs to be used once before deployment to configure the model and reduce the solvers runtime in the future.

The method described above was implemented using a combination of GAMS and Java. The model was implemented using standard methods in GAMS and solved using IBM's CPLEX solver. Scenarios were described in Java configuration files as shown in below.

market 1 = true
market 2 = false
market 3 = true
market 4 = true
market 5 = false
market 6 = false

Such a configuration file can then used to configure a GAMS file with all the appropriate import statements as shown below. A market is then simply removed by commenting out an import statement.

import market1.gms
*import market2.gms
import market3.gms
import market4.gms
*import market5.gms
*import market6.gms

The SPEA2 algorithm was implemented in Java, which would create the module instances and then use the CPLEX solver to solve them.

5 Results

In this section we demonstrate the feasibility of our method on a small example. We apply our method to the market configuration for a specific hydro-power plant. As previously stated, the energy produced by the hydro-power plant can be sold on different markets, as shown below with their abbreviations:

- market 1: day-ahead market
- market 2: intra-day market
- market 3: primary reserve market (PRL)
- market 4: secondary reserve market (SRL)
- market 5: positive tertiary reserve market (TRL+)
- market 6: negative tertiary reserve market (TRL-)

However, some markets are rarely used and therefore the models complexity can be reduced by identifying configurations that remove the unnecessary markets. Therefore we apply our method and produce the results shown in Figures 6 and 7.

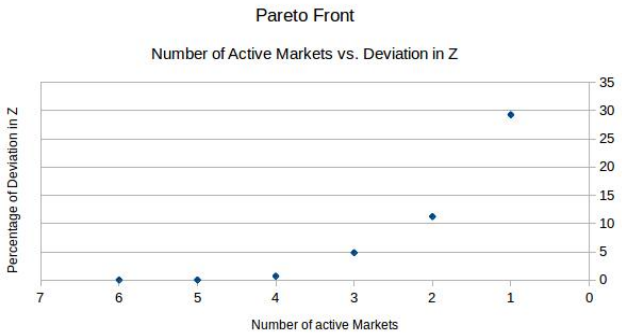


Figure 6. Graph showing the results of using the SPEA2 algorithm configure different sets of active markets. The X axis represents how many markets (or market modules) are activated in a configuration and the Y axis represents how much percent the results to the models objective function deviates from the most accurate answer (in this case when all six markets are activated). The data points represent the Pareto optimal configurations, meaning there is no configuration with a smaller deviation for the given number of modules.

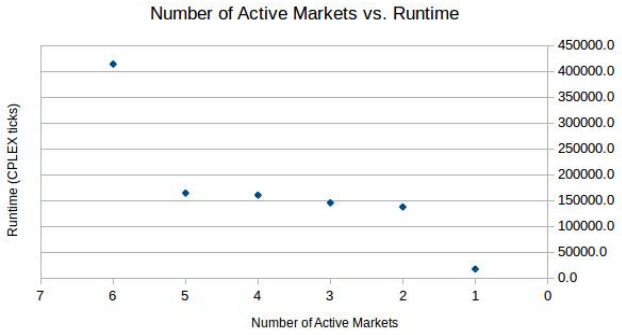


Figure 7. Graph showing the runtime for the configurations shown in Figure 6. The runtime is measured in CPLEX ticks, which is a platform independent measure for how much *work* Cplex had to do to solve the model.

Notable in the results is that market configurations exist that only use a subset of the available markets, but generate near identical results. The reason for this is that some markets simply prove to be

less profitable than others and therefore are used only in extremely rare circumstances. In the results shown, we have several interesting configurations that stand out and are shown in the table below:

#	Active Markets	Runtime	% Deviation
1	1,2,3,4,5,6	413722.0	0.0000
2	1,3,4,5,6	164166.0	0.0062
3	1,4,5,6	160181.0	0.6736
4	1,3,4,5	145352.3	4.8373

Table 1. Table showing CPLEX performance for various configurations. Notable is the the high runtime in cofiguration 1 depsite similar results (similar deviation) to cofiguration 2 and 3. Also notable is the increase in deviation for cofiguration 4.

6 Conclusion

Configuration 1 in the table above is the reference point. Interestingly, configuration 2 and 3 only deviate from the reference point by less than 1% and configuration 4 by less that 5%. Configuration 2,3 and 4 also show a more than 50% faster solve time. Therefore we should consider to configure the model for this topology to ignore markets 2 (intra-day market),3 (primary reserve market) and 6 (negative tertiary reserve market) to achieve a faster solver time. It should also be noted that the configuration 2,3 and 4 are closely related to each other, hinting towards favourable conditions for evolutionary algorithms as they can derive one good configuration from another easily through mutation and crossover operations.

To summarise, we describe an AI problem that exists for the configuration of mathematical models. We apply a method for identifying the Pareto front based on two objectives, one an approximation of the models complexity and the other a measurement of how much the models answer, and therefore its behaviour, changes. The method was applied on a small example to identify which markets are the least profitable and may be exclude from the model to provide a faster runtime. Although the example is only small, it provides a proof of concept for our method and shows how it can be applied.

7 Future Work

This work serves as a basis for our future work as we well indulge further into the implications of flexible behaviour in a mathematical model. In particular we will focus on the runtime prediction of such models, as the runtime becomes more difficult to predict with high functional flexibility. We will also focus on the turning of the mathematical solver. For traditional models, it is possible to select optimal parameters for all possible model instances. However, with a more flexible model, the solver requires different tuning parameters for various configurations to be optimal. Therefore, our research will focus on using knowledge from the model structure to determine ideal model tuning parameters for each possible model configuration.

8 Acknowledgements

This work has been done in the context of the SNSF funded project *Hydro Power Operation and Economic Performance in a Changing Market Environment*. The project is part of the National Research Programme *Energy Transition* (NRP70).

REFERENCES

[1] *GAMS/CPLEX 10 Solver Manual*, GAMS Development Corporation.

[2] Lorenzo Alfieri, Paolo Perona, and Paolo Burlando, 'Optimal water allocation for an alpine hydropower system under changing scenarios', *Water resources management*, **20**(5), 761–778, (2006).

[3] Christoph Böhringer, 'The synthesis of bottom-up and top-down in energy policy modeling', *Energy economics*, **20**(3), 233–248, (1998).

[4] GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 24.2.1. Washington, DC, USA, 2013.

[5] Emmanuel Fragniere and Jacek Gondzio, 'Optimization modeling languages', *Handbook of Applied Optimization*, 993–1007, (2002).

[6] Shenglian Guo, Jionghong Chen, Yu Li, Pan Liu, and Tianyuan Li, 'Joint operation of the multi-reservoir system of the three gorges and the qingjiang cascade reservoirs', *Energies*, **4**(7), 1036–1050, (2011).

Solving the Partner Units Configuration Problem with Heuristic Constraint Answer Set Programming

Erich C. Teppan[1]

Abstract. The partner units problem (PUP) is an acknowledged hard benchmark problem for the logic programming community with various industrial application fields. The state-of-the-art heuristic for the PUP is the QuickPup heuristic. Unfortunately, complex domain-dependent heuristics like QuickPup could not be realized within a declarative solving framework like constraint or answer set programming. A new hybrid technique called constraint answer set programming (CASP) offers the possibility to realize declarative frameworks in which it is possible to also express complex heuristics like QuickPup. In this paper we present the CASP solver ASCASS (A Simple Constraint Answer Set Solver) which provides novel methods for defining and exploiting problem-dependent search heuristics. Beyond the possibility of using already built-in problem-independent heuristics, ASCASS allows on the ASP level the definition of problem-dependent variable selection, value selection and pruning strategies which guide the search of the CP solver. Due to the new possibilities for representing and exploiting complex domain heuristics in ASCASS, we show how to encode the PUP and realize QuickPup in ASCASS. An evaluation reveals that due to the QuickPup heuristic, which is not expressible in any other ASP or CASP approach, ASCASS outperforms state-of-the-art ASP and CASP solvers on the tested PUP instances.

1 Introduction

The partner units problem (PUP) [1] is a perfect representative of a configuration problem in the classical sense, i.e. where certain components have to be connected so that predefined user requirements and technical constraints are respected [14]. Because of its generic nature it posses many real world application domains like railway safety, surveillance or electrical engineering [2, 17]. The PUP is $\mathcal{NP}$-complete in the general case and also for most industrially important subclasses. Furthermore, it is one of the hardest benchmark problems participating in the ASP competitions[2][19].

The PUP originates in the domain of railway safety systems. One of the problems in this domain is to make sure that certain rail tracks are not occupied by a train/wagon before another train enters this track. The signals for the corresponding occupancy indicators are calculated by special processing units based on the input of several observing sensors. Because of fail-safety and realtime requirements the number of sensors respectively indicators which can be connected to the same unit is limited (called unit capacity, UCAP). Also one sensor/indicator device can only be directly connected to one unit. However, a unit can be connected to a limited number (called inter unit capacity, IUCAP) of other units. These units are called the partner units of the unit. Devices (i.e. sensors and indicators) can only communicate with devices connected to the same unit and with devices connected to one of the partner units. Given the IUCAP, UCAP and a bipartite input graph represented by edges specifying which sensor data is needed in order to calculate the correct signal of an occupancy indicator, the problem consists in connecting sensors/indicators with units and units with other units such that all communication requirements are fulfilled and IUCAP and UCAP are not violated. For minimizing hardware costs, a common further objective is the minimization of the number of used units.

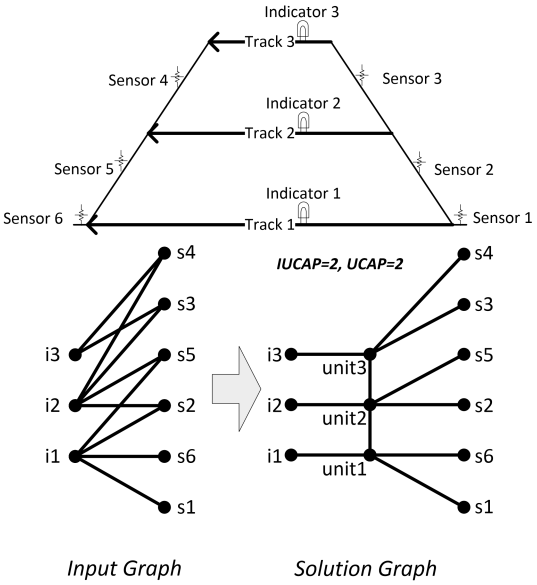


Figure 1. Railway track layout, PUP input and solution

Figure 1 shows a simple example for a railway track layout, the corresponding bipartite input graph and a possible solution for IUCAP=2 and UCAP=2. In order to calculate the correct signal for Indicator 3 only data from Sensor 3 and Sensor 4 is needed. If the number of outgoing wheels counted by Sensor 4 is equal to the incoming wheel counts of Sensor 3 then Track 3 is empty. In order to calculate the correct signal for Indicator 2 it is not sufficient to only incorporate data from Sensor 2 and Sensor 5 as it is not clear whether a wheel has headed to or is coming from Track 3. Therefore, additional data from Sensor 3 and Sensor 4 is needed.

[1] Universität Klagenfurt, Austria, email: firstname.lastname@aau.at
[2] Further information can be found at www.mat.unical.it/aspcomp2014/

Several general problem solving methods have been tested on the PUP such as Integer Programming, Constraint Programming (CP), SAT Solving, and Answer Set Programming (ASP), among of which ASP performed best (Aschinger et al. 2011a). However, the real breakthrough for solving real-world instances of the PUP was achieved by the development of the domain-dependent QuickPup heuristic [18].

QuickPup is based on three major techniques:

1. Based on the input graph and a distinguished root indicator, QuickPup produces a topological ordering of the devices, which is basically the minimum distances from the root indicator to all other devices. The distance to itself is zero, the distance to the direct neighbors is one, the distance to the neighbors of the neighbors is two and so forth. This reflects the (partial) ordering in which the devices should be processed.
2. For each device, first try to place it on the next empty unit and if this is unsuccessful try the already used units in descending order.
3. Try different root indicators, and consequently different topological orderings, and limit search for each trial. The intuition behind that is that not all root indicators are equally good to start search from.

Unfortunately, also state-of-the-art general problem solvers do not allow the formulation of complex domain-dependent heuristics like QuickPup without external non-declarative procedures. For example, for realizing QuickPup in CP it would afford a custom (procedurally programmed) propagator respectively global constraint. CP languages like MiniZinc[3] do not allow complex calculations like the computation of a topological order in a graph. In ASP heuristics like QuickPup are not possible as ASP is stateless in nature such that things like 'the next empty unit' are not expressible.

However, the new hybrid approach of Constraint Answer Set Programming (CASP) [13] combining ASP and CP seems to be perfect to overcome these issues. On the one hand, ASP provides language constructs which even go beyond first-order logic. On the other hand, CP is not stateless. Furthermore, a heuristic in CP is well contoured. Any heuristic in CP basically consists of three components:

1. a problem-dependent variable selection strategy
2. a problem-dependent value selection strategy
3. a problem-dependent pruning strategy

For certain classes of problems like industrial-sized scheduling CASP was already successfully applied [4]. Especially search problems with large variable domains often profit from the CASP representation due to the alleviation of the grounding bottleneck [12].

In this paper we present ASCASS, a novel CASP solver which uses Clingo for answer set solving and the Java framework Jacop for CP solving. ASCASS combines and extends the heuristic possibilities of state-of-the-art CASP solvers and makes them completely available on the problem encoding level. Beyond the usage of built-in strategies, ASCASS provides powerful constructs for the formulation and exploitation of problem-dependent heuristics consisting of variable selection, value selection and pruning strategies.

Subsequently, it is discussed in detail how to represent the Partner Units Problem (PUP) and the QuickPup heuristic in ASCASS. In a first proof-of-concept evaluation it is shown that due to this heuristic, which, to the best of our knowledge, cannot be expressed within any other ASP or CASP approach, ASCASS outperforms state-of-the-art ASP and CASP solvers.

[3] www.minizinc.org

2 Background

In this section we introduce the basic concepts of answer set and constraint answer set programming as it is needed for the purposes of this article. In particular, we ignore disjunctive logic rules and classical negation in ASP for readability reasons. For information about ASP and CASP please refer to [8], [7], [13], [15] and [3].

2.1 Syntax of ASP

in ASP, a *term* refers either to a *variable* or a *constant*. Strings starting with upper case letters denote variables. Constants are represented by strings starting with lower case letters, by quoted strings or by integers. An *atom* is either a *classical atom*, a *cardinality atom* or an *aggregate atom*. A classical atom is an expression $p(t_1, \ldots, t_n)$ where p is an n-ary predicate and $t_1, \ldots, t_n$ are terms. A *negation as failure (NAF) literal* is either a classical atom λ or its negation $not\ \lambda$. A cardinality literal is either a cardinality atom ψ or its negation $not\ \psi$. A *cardinality atom* is of the form

$$l \prec_l \{a_1 : l_{1_1}, \ldots, l_{1_m}; \ldots; a_n : l_{n_1}, \ldots, l_{n_o}\} \prec_u u$$

where

- $a_i : l_{i_1}, \ldots, l_{i_j}$ represent *conditional literals* in which a_i (the heads of the cardinality atom) constitute classical atoms and l_{i_j} are NAF literals
- l and u are terms (i.e. variables or constants) representing non-negative integers. If not specified, the defaults are 0 respectively ∞.
- $\prec_l$ and $\prec_u$ are comparison operators. If not specified, the default is $\leq$.

An aggregate literal is either an aggregate atom φ or its negation $not\ \varphi$. An *aggregate atom* is of the form

$$l \prec_l \#op\{t_{1_1}, \ldots, t_{1_m} : l_{1_1}, \ldots, l_{1_n}; \ldots; t_{o_1}, \ldots, t_{o_p} : l_{o_1}, \ldots, l_{o_q}\} \prec_u u$$

Most syntactical parts of aggregate literals are the same as for cardinality atoms, except that

- a head of a conditional literal is a tuple of terms $t_{i_1}, \ldots, t_{i_j}$ and
- $\#op$ is an aggregate function in $\{\#min, \#max, \#count, \#sum\}$.

Generally, a *rule* is of the form

$$h \leftarrow b_1, \ldots, b_m, not\ b_{m+1}, \ldots, not\ b_n.$$

where

- $h, b_1, \ldots, b_m$ are atoms (i.e. positive literals),
- $not\ b_{m+1}, \ldots, not\ b_n$ are negative literals,
- $H(r) = \{h\}$ is called the *head* of the rule,
- $B(r) = \{b_1, \ldots, b_m, \ldots, not\ b_{m+1}, \ldots, not\ b_n\}$ is called the body of the rule,
- $B^+(r) = \{b_1, \ldots, b_m\}$ is called the positive body of the rule and
- $B^-(r) = \{not\ b_{m+1}, \ldots, not\ b_n\}$ is called the negative body of the rule.

A rule r with $H(r)$ including a cardinality atom is called *choice rule*. A rule r where $B(r) = \{\}$, e.g. '$a \leftarrow$' is called *fact*. For facts, typically '$\leftarrow$' is omitted. A rule r where $H(r) = \{\}$, e.g. '$\leftarrow b$', is called *integrity constraint*, or simply *constraint*.

Furthermore, we allow the typically built-in arithmetic functions ($+, -, *, /$) and comparison predicates ($=, \neq, <, >, \leq, \geq$).

2.2 Semantics of ASP

The semantics of a non-ground ASP program is defined w.r.t. its *grounding*. A program's grounding can be defined in terms of its Herbrand universe and base. The *Herbrand universe* HU_P of a program P is the set of all constants appearing in P.

The grounding for a rule r without cardinality atoms and aggregates is the set of rules obtained by applying all possible substitutions of variables in r with constants in HU_P. The grounding of a rule which contains cardinality or aggregate literals is defined by the two-step instantiation described in [16]: first produce a set of partially grounded rules by substitution of variables occurring outside the cardinality/aggregate literal and then, within each partially grounded rule, substitute each conditional literal by a set of ground conditional literals by substituting the remaining variables inside the cardinality or aggregate literal.

The *grounding* P_G of a program P is the union of all rule groundings. The *Herbrand base* HB_P w.r.t P is the set of all positive NAF literals (i.e. classical atoms) that occur in P_G.

An interpretation I satisfies a (ground) positive NAF literal λ (written as $I \vDash \lambda$) iff $\lambda \in I$. A positive cardinality literal is satisfied by I iff the number of satisfied head literals in the cardinality atom satisfies the lower and upper bounds l and u w.r.t. the order relations $\prec_l$ and $\prec_u$. Both, bounds and comparison symbols are optional. By default, $0 \leq$ is used for the lower and $\leq \infty$ for the upper bound. A positive aggregate literal is satisfied iff the value returned by the aggregate function $\#op$ applied on the set of term tuples fulfilling its conditions does not violate the lower and upper bounds. Here, $\#count$ counts the number of distinct term tuples fulfilling the related conditions, and $\#min, \#max$ and $\#sum$ are calculating the minimum, maximum or sum of the first terms in the distinct term tuples fulfilling the related conditions. A negative literal $not\ \omega$ is satisfied (written as $I \vDash not\ \omega$) iff ω is not satisfied.

A ground rule r is satisfied by I (written as $I \vDash r$) iff the head is satisfied or the body is not. The body of a rule is satisfied by I iff all literals in the body are satisfied. The head of a rule is satisfied iff the literal in it is satisfied. In particular, an empty body is always satisfied and integrity constraints are satisfied iff the body is not satisfied, i.e. the constraint is not violated. A program P is satisfied by an interpretation I iff all rules in its grounding P_G are satisfied.

An answer set for a program can be defined on the basis of the program's *reduct* [9, 16]. The reduct P^I of a ground program P relative to an interpretation $I \subseteq HB_P$ is defined as $P^I := \{H(r) \leftarrow B(r)^+ : r \in P, B(r)^- \cap I = \emptyset\}$.

An interpretation $I \subseteq HB_P$ (which may be empty) is an *answer set* for a program P not containing choice rules iff

- I satisfies all rules r in P^I, i.e. $\forall r \in P^I$: $I \vDash r$ and
- I is subset-minimal, i.e. there is no $I' \subset I$ so that I' satisfies all rules in $P^{I'}$.

Choice rules can produce answer sets that are not subset-minimal, which leads to a slight change of semantics when such rules are present. For example, the program consisting only of the choice rule $\{a\}.$ possesses the two answer sets $\{\}$ and $\{a\}$. In order to be in line with the original semantics and thus restore subset-minimality an equivalent program can be produced by extending the program as follows:

For every head a_i within a cardinality atom of a choice rule, add a new atom a_i' (which is not occurring elsewhere in the program) and a constraint $\leftarrow a_i, a_i'$. Informally, a_i' expresses that a_i is not in the interpretation. This way, the choice rule $\{a\}.$ equivalently produces the two answer sets $\{a'\}$ and $\{a\}$. For convenience, we can imagine the new atoms a_i' and the constraints $\leftarrow a_i, a_i'$ to be invisible. For details, consult [7].

An ASP program is *unsatisfiable* iff it has no answer sets and *satisfiable* otherwise.

2.3 Constraint Answer Set Programming

A constraint satisfaction problems (CSP) can be defined as a three-tuple $\langle V, D = \{dom(v) : v \in V\}, C\rangle$ whereby V is a set of variables, D is the set of domains of the variables in V and C is a set of constraints on variables in V. A solution to a CSP is an assignment $\forall v \in V, v := d \in dom(v)$ such that all constraints $c \in C$ are fulfilled. A CSP comprising only finite domains is called finite. If all domains are defined over discrete values (most commonly integers), the CSP is called discrete.

For integrating CP into ASP there are basically two approaches. First, solvers like Clingcon [15] are based on the extension of the ASP input language in order to support the definitions of constraints. The Ezcsp solver [3] is based on a different approach where ASP and CP are not integrated into one language. ASP rather acts as a specification language for Constraint Satisfaction Problems (CSPs). The main idea is that answer sets constitute CSP encodings which are used as input for a CP solver. The above example can be expressed in Ezcsp as:

```
num(N):-N=1..3.
cspdomain(fd).
cspvar(var(N),1,6):-num(N).
required(var(X) + var(Y) + var(Z) == 6):-
  num(X), num(Y), num(Z), X!=Y, Y!=Z, X!=Z.
required(var(1) > 1).
required(all_distinct([var/1]).
```

After some pre-processing, an answer set is calculated which includes $cspdomain$-, $cspvar$- and $required$ facts. $cspdomain(fd)$ denotes that the CSP is finite and discrete. Ezcsp is also able to handle real domains. CSP variables are explicitly defined by $cspvar$ facts also defining lower and upper bounds of the variable domains. Constraints are represented as $required$ facts. For expressing global constraints, and thus refer to sets of CSP variables, Ezcsp allows the usage of functional symbols. E.g. $[var/1]$ refers to all variables formed by the unary function var. Once an answer set has been produced, the CSP encoded within the $cspdomain$-, $cspvar$- and $required$ facts is passed to the CP solver. As answer set production and CSP solution search are two separated processes, different CP solvers can be used in Ezcsp. Currently, Sicstus- and B-Prolog are supported.

The semantics of a program builds on the notion of *extended answer sets* [3]: A pair $\langle A, S\rangle$ is an extended answer set of program Π iff A is an answer set of Π and S is a solution to the CSP defined by A. We further define that the empty CSP (i.e. without any CSP variables) possesses the empty solution.

For CSP solution search, Ezcsp provides different search strategies impacting the underlying CP solver. In case of Sicstus Prolog as a CP solver, the built-in value selection strategies *step* (min domain value, when ascending order is used, max domain value when descending order is used) and *bisect* (bisection of the domain in the middle) are available. Similarly in case of B-Prolog, the bisection strategies *split* and *reverse_split* are supported. The supported variable selection strategies are *leftmost* (leftmost variable), *min* (leftmost variable with minimal lower bound), *max* (leftmost variable with maximal upper bound), and *ff* (first-fail). By the special *label_order/2* predicate it

is also possible to define problem-dependent CSP variable orderings for the CP solver. However, there are no constructs for expressing problem dependent value or pruning strategies.

3 A Simple Constraint Answer Set Solver

ASCASS[4] is a novel finite discrete CASP solver following the approach of Ezcsp, i.e. the input language is pure ASP and the answer sets encode CSPs. The major difference to to state-of-the-art CASP systems are the sophisticated features for expressing domain-dependent heuristics consisting of custom variable selection, value selection and pruning strategies.

Figure 2 shows the overall architecture of ASCASS. Answer set production (grounding and solving) is done by Clingo[5], which is currently one of the most powerful ASP systems. The input language is the ASP standard ASP-Core-2[6].

After answer set solving, a produced answer set is handed over to a parsing module that extracts the facts which encode the CSP and search directives. This information is used to instantiate a corresponding CSP in the CP solver and perform search conforming to the given search directives. Currently, Jacop[7] is used within ASCASS as a CP solver. In case that the CSP could not be solved by the CP solver or a timeout occurred (defined by the special predicate $csptimeout(\Delta)$), the process continues with the next answer set, until a solution is found, or there are no more answer sets. The empty CSP (i.e. when there is not a single CSP variable) is always satisfiable and possesses the empty CSP solution.

3.1 Encoding of CSPs

ASCASS focuses on finite discrete Constraint satisfaction problems (CSPs). In order to encode a CSP within ASCASS there can be used a number of specific predicates. Of course, in the input these predicates can contain variables. The following explanations refer to their grounded form.

The predicates $cspvar(\alpha, \lambda, \upsilon)$ and $cspvar(\alpha, \lambda, \upsilon, \eta)$ are responsible for encoding CSP variables. Hereby, α represents the variable name and λ and υ represent respectively the numerical lower and upper bound of the variable's domain. For example $cspvar(x, 1, 10)$ stands for a CSP variable v with the domain $[1..10]$. The numerical priority η is used to define a custom variable selection ordering. When using the variable selection strategy $priority$ (see below), the CP solver selects the variable with the highest priority first.

The predicate $cspconstr(\alpha, \rho, \tau)$ encodes a relational constraint (i.e. $=, <>, <, <=, >, >=$) over a variable α. ρ denotes the type of relation and must be a constant out of $\{eq, neq, lt, lteq, gt, gteq\}$. τ denotes another CSP variable or a numerical constant. For example, $cspconstr(x, lt, 5)$ expresses that variable x must be lower than 5.

The predicate $csparith(\alpha, \pi, \beta, \rho, \gamma)$ encodes arithmetic constraints. α, β and γ are CSP variable names. Like for $cspconstr$, the constant ρ denotes the type of relation. π is a constant representing an arithmetic operation. Currently, ASCASS supports addition ($plus$), subtraction ($minus$), multiplication ($mult$), division (div) and exponent (exp). For example, $csparith(xa, plus, xb, eq, xc)$ states that the sum of the values of xa and xb must be equal the value of xc.

For expressing logical constraints predicates of the form $cspif(\Xi_1, and, \Xi_2, and, \ldots, and, \Xi_m, then, \Xi_{m+1}, or, \Xi_n)$ can be used. Each Ξ consists of a variable α, a relational symbol ρ and another variable or numerical constant τ. For example, $cspif(x, lt, 5, and, y, gt, 10, then, z, gteq, 0)$ is to be read as 'if x is lower than 5 and y is greater than 10 then z must be non-negative'.

Global constraints are constraints over arrays of variables. In ASCASS global constraints are defined by predicates of the form $cspglobal(\sigma_1, \ldots, \sigma_m, \kappa)$ and $cspglobal(\sigma_1, \ldots, \sigma_m, \kappa, \tau_1, \ldots, \tau_n)$. κ is a constant denoting the type of global constraint. $\sigma_1, \ldots, \sigma_m$ represent arrays of variables. $\tau_1, \ldots, \tau_n$ represent single CSP variables or integers. The selection of global constraints currently supported by ASCASS has been determined by the needs of our application areas and will be further expanded. ASCASS currently supports the following global constraints[8]:

- min: $cspglobal(\sigma, min, \tau)$, the minimum value of the variables σ is equal to τ
- max: $cspglobal(\sigma, max, \tau)$, the maximum value of the variables σ is equal to τ
- sum: $cspglobal(\sigma, sum, \tau)$, the sum of values of the variables σ is equal to τ
- count: $cspglobal(\sigma, count, \tau_1, \tau_2)$, τ_1 is equal to the counted number of variables in σ with value τ_2
- global cardinality: $cspglobal(\sigma_1, \sigma_2, gcc)$, a more general counting constraint where the occurring values in σ_1 are counted in the corresponding counter variables in σ_2
- all different: $cspglobal(\sigma, alldiff)$, all variables in σ are mutually unequal
- element: $cspglobal(\sigma, element, \tau_1, \tau_2)$, the value of the τ_1-th variable in σ is equal to τ_2
- cumulative: $cspglobal(\sigma_1, \sigma_2, \sigma_3, cumulative, \tau)$, σ_1 represents the starting times of $|\sigma_1|$ many jobs, σ_2 represents the durations of the jobs, σ_3 represents the amounts of needed resources of the jobs and τ represents the allowed accumulated amount of resources at any time point
- bin packing: $cspglobal(\sigma_1, \sigma_2, \sigma_3, binpacking)$, σ_1 represents bin assigments for $|\sigma_1|$ many items, σ_2 represents the bin sizes of the $|\sigma_2|$ many bins and σ_3 represents the item sizes

In order to address arrays of CSP variables, ASCASS not only allows simple constants but also n-ary functional terms for variable names of the form $\phi(\iota_1, \ldots, \iota_n)$ with $\iota_1, \ldots, \iota_n$ representing string or integer arguments (see Figure 3). The special functional argument *all* acts as a placeholder and can be used for addressing arrays of variables. For example, take the four variable definitions $cspvar(v(1,1), 1, 10)$, $cspvar(v(1,2), 1, 10)$, $cspvar(v(2,1), 1, 10)$ and $cspvar(v(2,2), 1, 10)$. A natural interpretation of the arguments is *row* and *column* of a two-dimensional variable array. Consequently, $cspglobal(v(all, 2), alldiff)$ expresses that the values of all second column's variables, in our case $v(1,2)$ and $v(2,2)$, must be different to each other. *v(all,all)* stands for all variables in the two-dimensional array, i.e. all variables formed by the functional symbol v with arity 2.

[4] http://isbi.aau.at/hint/ascass
[5] sourceforge.net/projects/potassco/files/clingo
[6] www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03b.pdf
[7] jacop.osolpro.com

[8] More information about global constraints can be found at http://jacop.osolpro.com/guideJaCoP.pdf and http://sofdem.github.io/gccat/

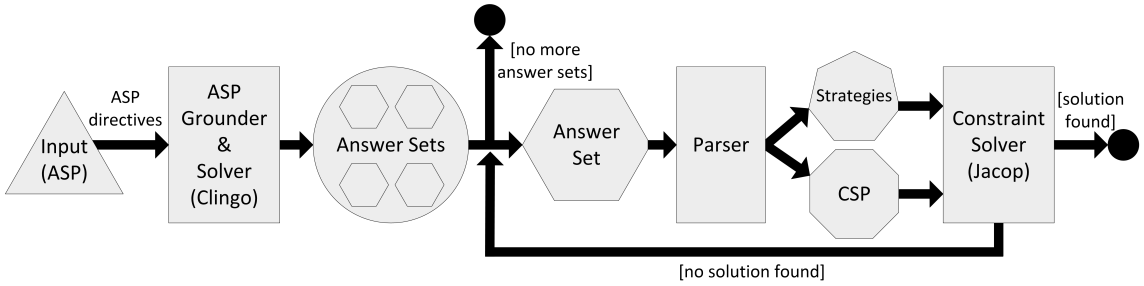


Figure 2. Architecture of ASCASS

v(1,1)	v(1,2)	...	v(1,c)	v(1,all)
v(2,1)	v(2,2)	...	v(2,c)	v(2,all)
...	...	...	...	...
v(r,1)	v(r,2)	...	v(r,c)	v(r,all)
v(all,1)	v(all,2)	...	v(all,c)	v(all,all)

Figure 3. Concept of variable arrays in ASCASS

3.2 Encoding of variable selection strategies

Apart from the predicates for defining a CSP, ASCASS provides predicates for steering the search of the CP solver. The predicates $cspvarsel(\epsilon)$ and $cspvarsel(\epsilon, \theta)$ define the variable selection strategy to be used. Hereby, ϵ is the primary selection strategy and, if defined, θ acts as a secondary, tiebreaking strategy. For variable selection, ASCASS currently supports the problem-independent built-in strategies *smallestDomain*, *mostConstrainedStatic*, *mostConstrainedDynamic*, *smallestMin*, *largestDomain*, *largestMin*, *smallestMax*, *maxRegret*, *weightedDegree* and the problem-dependent strategy *priority*.

When using the *priority*-strategy, ASCASS builds an ordering of the CSP variables based on the provided priorities η in $cspvar(\alpha, \lambda, \upsilon, \eta)$. Variables with high priorities are selected first. Variables for which there is no η defined are selected as the last ones. Hence, the $priority$ strategy in combination with the variable priorities is similar to the *label_order* predicate in Balduccini's EZCSP.

3.3 Encoding of value selection strategies

For value selection ASCASS provides the predicates $cspvalsel(\phi)$ and $cspvalsel(\phi, \varphi)$ where ϕ and φ are constants denoting the strategy. As it is often important to have different value selection strategies for different sets of variables, ASCASS provides also the predicates $cspvalsel(\sigma, \phi)$ and $cspvalsel(\sigma, \phi, \varphi)$ where σ represents an array of variables like in global constraints. ASCASS supports the already built-in strategies *indomainMin*, *indomainMiddle*, *indomainMax* and *indomainRandom*. For expressing problem-dependent value selection strategies, the novel strategy *indomainPreferred* can be used.

When using *indomainPreferred*, the CP solver first tries to use specified values before changing to the built-in strategy φ (*minDomain* if not stated otherwise). For specifying preferred values, ASCASS provides the special predicate $csppprefer(\alpha, \rho, \tau)$ and $csppprefer(\alpha, \rho, \tau, \eta)$. Like for relational constraints, α represents a CSP variable, ρ represents a relational symbol and τ stands for a further variable or a numerical constant. For example, $csppprefer(v, eq, 5)$ states that for the CSP variable v a preferred value is 5. In order to specify an ordering of the specified values, it is possible to make use of a numerical priority η. Higher priority statements are taken into account first by ASCASS. For example, if there is given $csppprefer(v, eq, 5, 1)$ and $csppprefer(v, eq, 20, 2)$, ASCASS tries to first label v with 20 and only after that with 5. Of course, only preferred values are taken into account which are still in the variable's domain. In case that τ denotes another variable, the minimum value in the current domain of τ is used as a preferred value, i.e. τ does not need to be singleton for specifying a preferred value of α. This in combination with global constraints is a highly dynamic and powerful mechanism.

As with the relational constant eq in combination with the priorities η every ordering of preferred values can be expressed, the usage of lt, $lteq$, gt and $gteq$ can be clearly seen as syntactic sugar. By using lt, $lteq$, gt and $gteq$ sets of preferred values can be expressed:

- $lteq\ \tau : \{\tau, \tau - 1, \ldots, -\infty\}$
- $lt\ \tau : \{\tau - 1, \ldots, -\infty\}$
- $gteq\ \tau : \{\tau, \tau + 1, \ldots, \infty\}$
- $gt\ \tau : \{\tau + 1, \ldots, \infty\}$

Note that all preferred values of such a set P have the same priority (possibly given explicitly by η). For defining an order relation over P, i.e. fix the order in which ASCASS considers the preferred values in P, the following holds: For lt and $lteq$ decreasing order is used, i.e. $\tau, \tau - 1, \ldots, -\infty$ and for gt and $gteq$ increasing order is used, i.e. $\tau, \tau + 1, \ldots, \infty$. For example having the variable definition $cspvar(v, 1, 10)$ and the value selection strategy $cspvalsel(indomainPreferred, indomainMin)$, $csppprefer(v, lt, 5)$ would effect that ASCASS considers the domain values in the following order: $4, 3, 2, 1, 5, 6, 7, 8, 9, 10$. The reason why for lt and $lteq$ descending order and for gt or $gteq$ ascending order is used is simply the following: Would it be the other way round, the behavior with lt and $lteq$ would conform to $indomainMin$ and with gt and $gteq$ to $indomainMax$.

3.4 Encoding of pruning strategies

The third component of many problem-dependent heuristics is the pruning strategy. For specifying how a search tree is pruned, ASCASS provides the special predicate $cspsearch(\omega, \mu)$. Hereby, ω specifies the pruning type and μ specifies a numerical limit that, when reached, triggers backtracking. Again it could be beneficial

having different limits for different groups of variables or even having no limit on certain variables whilst search on others is limited. To this, ASCASS provides the predicate $cspsearch(\sigma, \omega, \mu)$ with σ denoting an array of variables like for global constraints.

Currently, ASCASS provides two pruning types. $cspsearch(limited, \mu)$ limits the number of wrong decisions for variables. If the number μ of wrong choices for a variable is reached, backtracking is triggered and the counter for the variable is reset. For example, $cspsearch(limited, 3)$ specifies that for every variable v there must not be more than three labeling trials for v within a search branch. The second pruning type is based on limited discrepancy search [10] and operates on the level of search paths. When specifying $cspsearch(lds, \mu)$ only a certain number of wrong decisions (called discrepancies) along the whole search path is allowed. If this number reaches μ, backtracking is triggered.

Furthermore, it is possible to limit search time of the CP solver by $csptimeout(\Delta)$ where Δ is the number of seconds when the timeout is triggered. The timeout concerns only the search of Jacop so that search might start over based on the next answer set if such exists.

3.5 Directives for answer set production

In order to specify which heuristic is to be used by Clingo, the special ASCASS predicate $aspheuristic(\nu)$ can be used. Hereby, ν is a constant denoting the heuristic which is passed to Clingo as a command line option $--heur = \nu$. As this happens before the actual answer set solving, $aspheuristic(\nu)$ must only be used as a single fact within program source code. Common heuristics to be used are $VSIDS$ or $Berkmin$ [11]. When using $aspheuristic(domain)$, Clingo uses a user-defined heuristic defined via the special predicate $\_heuristic$ [6]. For limiting the number of produced answer sets, the special predicate $aspnumas(\Delta)$ can be used. Δ is a non-negative integer and is passed to Clingo as a command line option. The default is '1' and '0' effects the production of all answer sets. Like $aspheuristic$, also $aspnumas$ must only be used as a single fact within the problem source code. Similarly, $asptimeout(\Delta)$ specifies a timeout for answer set solving.

4 Solving the PUP with ASCASS

The input comprises of a set of $egde(i, s)$ facts where i takes the numerical identifier of an indicator and s takes the identifier of a sensor. Additionally the input includes a fact $ucap(x)$ with $x > 0$ that defines the unit capacity (UCAP) and a fact $iucap(y)$ with $y > 0$ that defines the inter-unit capacity (IUCAP).

For the code snippets given in the remainder of this section we use the standard notation of logic programming. In particular, left-implication $\leftarrow$ is represented as ':-'.

In order to produce explicit indicator and sensor information the following lines of code are used:

```
sensor(S):-edge(I,S).
indicator(I):-edge(I,S).
numIndicators(N):-N=#count{I:indicator(I)}.
numSensors(N):-N=#count{S:sensor(S)}.
```

The number of indicators ($numIndicators$) respectively sensors ($numSensors$) are calculated by means of the $\#count$ aggregate literal provided by Clingo.

We restrict the number of units ($numUnits$) available for a solution to the theoretical lower bound, i.e. $numUnits = \left\lceil \frac{max(numIndicators, numSensors)}{UCAP} \right\rceil$:

```
max(M):-numIndicators(E),numSensors(F),
                                    M=#max(E;F).
numUnits(N):-max(M),ucap(C),N=(M+1)/C.
unit(Z):-numUnits(N),1<=Z,Z<=N.
```

For each indicator i there is a CSP variable $device(i, 1)$ and for each sensor s there is a CSP variable $device(s, 2)$. This way it is also possible to refer to the array of all CSP device variables as $device(all, all)$, to only the indicator variables as $device(all, 1)$ and to the sensor variables as $device(all, 2)$ which will be useful later. The value range for these CSP variables is $[1..numUnits]$. Furthermore, the variables get a priority defining the topological order in which they are labeled by ASCASS:

```
cspvar(device(I,1),1,N,P):-
                  numUnits(N),iPriority(I,P).
cspvar(device(S,2),1,N,P):-
                  numUnits(N),sPriority(S,P).
```

The calculation of the priorities is explained in detail below.

In order to assure UCAP, for each unit u there are two counting variables $ci(u)$ and $cs(u)$. These variables can take values in the range $[0..UCAP]$. Furthermore, for each unit u there are two $count$ global constraints counting the number of indicator respectively sensor variables taking the value u:

```
cspvar(ci(U),0,C):-ucap(C),unit(U).
cspvar(cs(U),0,C):-ucap(C),unit(U).
cspglobal(device(all,1),count,ci(U),U):-
                                    unit(U).
cspglobal(device(all,2),count,cs(U),U):-
                                    unit(U).
```

In order to capture which unit $u1$ is connected to which unit $u2$ there are $numUnits \times numUnits$ many CSP variables (i.e. $conn(U1, U2)$). The variables can take values in the range [0..1] if $u1 <> u2$. Otherwise, the variables' ranges consists of only a single value, i.e. [1..1]. This is because in our model each unit u is always connected to itself. Furthermore, there is a constraint assuring symmetry, i.e. if $u1$ is connected to $u2$ also $u2$ is connected to $u1$:

```
cspvar(conn(U1,U2),0,1):-unit(U1),unit(U2),
                                     U1<>U2.
cspvar(conn(U,U),1,1):-unit(U).
cspconstr(conn(U1,U2),eq,conn(U2,U1)):-
                    unit(U1),unit(U2),U1<U2.
```

For summing up how many units are connected to a unit u we make use of the global sum constraint. The used summing variables can hereby take values in the range $[1..IUCAP + 1]$ as every unit is also connected to itself:

```
cspvar(sumconns(U),1,K+1):-iucap(K),unit(U).
cspglobal(conn(U,all),sum,sumconns(U)):-
                                    unit(U).
```

In order to make the summing variables and constraints take effect, it must be assured that any connection variable $conn(u1, u2)$ is set to one whenever there is an $edge(i, s)$ in the input so that $device(i, 1) = u1$ and $device(s, 2) = u2$. Following the approach of [5], this is implemented by means of the global $element$ constraint. Given an array of CSP variables arr, an index i and a value v, an $element$ constraint assures that the i^{th} variable in arr is equal to v. In our case, for each $edge(i, s)$ in the input there is such a global constraint setting the appropriate connection variable within $conn(all, all)$ to one:

```
cspglobal(conn(all,all),element,index(I,S),1)
                                    :-edge(I,S).
```

As the $element$ constraint cannot directly handle multi-dimensional arrays, the respective index is calculated as $index(i,s) = (device(i,1)-1) \times numUnits + device(s,2)$. The formulation with constraints is straightforward.

The priorities for the device variables (i.e. $device(i,1)$ and $device(s,2)$ are based on a topological ordering of the devices. Given the layer of a sensor or indicator whereby the root of the topological graph is at layer zero, the priority is higher the lower the layer is:

```
iPriority(I,P):-indicatorLayer(I,L),P=9999-L.
sPriority(S,P):-sensorLayer(S,L),P=9999-L.
```

The effect is that, given a root indicator, ASCASS first tries to label the root indicator, then the neighbors of the root indicator, then the neighbors of the neighbors, and so on. In our implementation a choice rule is used to express that there is exactly one distinguished indicator that acts as root. This indicator is always placed at the first unit:

```
1{root(I):indicator(I)}1.
cspconstr(device(I,1),eq,1):-root(I).
```

The choice rule $1\{root(I) : indicator(I)\}1$ produces one answer set for each root indicator and asserts a $root(i)$ fact.

For calculating the actual layers, we first calculate the minimum distances to the root whereas root indicator has a zero distance to itself[9]:

```
indicatorDist(I0,0):-root(I0).
sensorDist(S,D+1):-indicatorDist(I,D),
                   edge(I,S),numDevices(M),D<M.
indicatorDist(I,D+1):-sensorDist(S,D),
                   edge(I,S),numDevices(M),D<M.
numDevices(N):-numIndicators(E),numSensors(F),
                                        N=E+F.
```

The layers are calculated by using the $\#min$ aggregate literal from Clingo:

```
indicatorLayer(I,Dmin):-indicator(I),
              Dmin=#min{D:indicatorDist(I,D)}.
sensorLayer(S,Dmin):- sensor(S),
               Dmin = #min{D:sensorDist(S,D)}.
```

First to try to place devices on unused units and, only if not successful, on used units in descending order can be expressed in ASCASS by means of preferred values:

```
cspprefer(device(I,1),lteq,nextUnit):-
                                  indicator(I).
cspprefer(device(S,2),lteq,nextUnit):-
                                     sensor(S).
```

The CSP variable $nextUnit$ points to the next unused unit, which is the current unit plus one[10]:

```
cspvar(curUnit,1,N):-numUnits(N).
cspvar(nextUnit,1,N+1):-numUnits(N).
csparith(curUnit,plus,one,eq,nextUnit).
```

For the calculation of the current unit, i.e. the highest number taken by some $device(i,1)$ or $device(s,2)$ variable, the global max constraint is used:

```
cspglobal(device(all,all),max,curUnit).
```

As ASCASS uses the lower bound of variables for calculating the preferred values, each device variable is first tried to be bound to values lower than or equal to the lower bound of $nextUnit = curUnit + 1$ in descending order.

In order to control how many units are maximally tried per device variable, the search is pruned such that only the next unit and a limited number of already used units can be tried before backtracking is triggered. In our implementation we use the following statement for only trying the next, the current and the last unit:

```
cspsearch(limited,3).
```

We furthermore restrict the maximum CSP search time for each call of the CP solver in order to try different start indicators:

```
csptimeout(300).
```

For making ASCASS respect the problem-dependent selection strategies, $cspvarsel(priority)$ and $cspvalsel(device(all,all), indomainPreferred)$ must be included. Thus, the concepts of QuickPup can be fully expressed in a declarative way by ASCASS. To the best of our knowledge, this is not possible within any other ASP or CASP approach.

4.1 Evaluation

We tested the ASP solver Clingo 4 and the CASP solvers ASCASS, Clingcon and Ezcsp on the PUP benchmark suite used in [2][11]. Clingo was tested using the PUP encoding proposed in [2][12]. The tests were run on a 3.2 Ghz machine with 64 GByte of RAM, assuring that the grounding bottleneck does not play a role for the tested instances[13] and performance can be attributed to the search phase.

In the Clingcon model, problem-dependent CSP variable selection, value selection or pruning strategies cannot be exploited. For Ezcsp, it is possible to express the topological variable orderings similar to ASCASS. However, there are no means for pruning search or problem-dependent value strategies.

Table 1 depicts how many instances of each type in the benchmark suite could be solved by the different approaches within a 1000 seconds time frame. Clingo using VSIDS heuristic peformed very well on the benchmark suite showing once again that the conflict-driven search techniques employed by Clingo are quite powerful. Also Ezcsp was able to solve some instances. Using other built-in heuristics did not result in better performance. Clingcon was not able to solve a single instance. In the contrary, ASCASS was able to solve all but one instances within time limits. We want to point out that only optimal solutions (i.e. minimum number of units) were allowed for easing the grounding bottleneck of conventional ASP. Increasing

[9] In order to make grounding safe, we have to limit the maximum possible distance which is equal to the total number of devices.

[10] Within the constraint, the helping variable $cspvar(one,1,1)$ is used as arithmetic constraints only accept variables in ASCASS.

[11] Encodings and benchmark instances can be found at http://isbi.aau.at/hint/ascass

[12] The 'new' encoding provided by the ASP competition 2014 was found to be inconsistent as it also produces answer sets for unsatisfiable instances.

[13] The biggest grounding in the ASP model was $\sim$ 12 GByte.

	#	Clingo	ASCASS	Clingcon	Ezcsp
double(IUCAP=2)	10	2	10	0	2
doublev(IUCAP=2)	6	3	6	0	0
triple(IUCAP=2)	3	2	3	0	2
triple(IUCAP=4)	7	6	6	0	3
grid(IUCAP=4)	10	10	10	0	0
total	36	23	35	0	7

Table 1. Solved instances whithin 600 seconds

the number of allowed units in a solution would increase grounding size for ASP significantly. In the cases of ASCASS and Ezcsp, increasing the number of allowed units would not affect the grounding size as the number of allowed units is captured by the upper bounds of the CSP variables.

Furthermore, we want to make clear that the superior performance of ASCASS can be attributed to the inclusion of the QuickPup strategies. This was crosschecked by removing the heuristic parts from the ASCASS problem encodings. It is to be noted that QuickPup originally was designed for producing only near-optimal solutions. However, the concepts of QuickPup obviously also work well for finding optimal solutions.

5 Conclusions

Solving configuration problems are among the biggest challenges and also major success stories of artificial intelligence. In this context, the Partner Units Problem (PUP) has gained more and more attention in the knowledge-based configuration community. The PUP is a classical configuration problem with many application fields like railway safety, CCTV surveillance or electrical engineering. Moreover, the PUP constitutes one of the hardest benchmark problems of the ASP competitions. Many general solving frameworks like SAT solving, integer, constraint or answer set programming have already been applied on the PUP. However, the domain-specific QuickPup heuristic has proven to outperform general problem solvers on real-world problem instances. Up to now, it was not possible to express complex domain heuristics like QuickPup within a declarative framework. The new hybrid technique of constraint answer set programming (CASP) introduces the chance to realize declarative frameworks with which it is possible to express and exploit also more complex domain heuristics like QuickPup.

In this paper we presented ASCASS, a novel CASP solver which allows the declarative formulation of domain-specific heuristics. In particular, ASCASS exploits ASP to generate problem-specific variable and value selection as well as pruning strategies for usage by the constraint solver. It was discussed in detail how the PUP and the QuickPup heuristic can be implemented in ASCASS. A first evaluation carried out on a well-established PUP benchmark clearly proofs the concept. Due to the sophisticated QuickPup heuristic, which can be expressed in ASCASS quite naturally but cannot be expressed in any other ASP or CASP system, ASCASS clearly outperforms state-of-the-art ASP or CASP systems on the tested instances.

ACKNOWLEDGEMENTS

Work has been funded by the Austrian Research Fund (FFG) in the context of project Heuristic Intelligence (*HINT*, FFG-PNr.: 840242) in cooperation with Siemens AG Österreich.

REFERENCES

[1] Falkner A., A. Haselboeck, G. Schenner, and H. Schreiner, 'Modeling and solving technical product configuration problems', *AI EDAM*, 115–129, (2011).

[2] M. Aschinger, C. Drescher, G. Friedrich, G. Gottlob, P. Jeavons, A. Ryabokon, and E. Thorstensen, 'Optimization methods for the partner units problem', in *CPAIOR'11*, pp. 4–19, Berlin, Heidelberg, (2011). Springer-Verlag.

[3] M. Balduccini, 'Representing constraint satisfaction problems in answer set programming', in *ICLP09 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'09)*, (2009).

[4] Marcello Balduccini, 'Industrial-size scheduling with asp+cp', in *Proceedings of the 11th Int. Conf. on Logic Programming and Nonmonotonic Reasoning*, LPNMR'11, pp. 284–296, Berlin, Heidelberg, (2011). Springer-Verlag.

[5] Conrad Drescher, 'The partner units problem: A constraint programming case study', in *ICTAI'12*, (2012).

[6] M. Gebser, B. Kaufmann, J. Romero, R. Otero, T. Schaub, and P. Wanko, 'Domain-specific heuristics in answer set programming', in *27th AAAI Conf. (AAAI'13)*, pp. 350–356, (2013).

[7] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub, *Answer Set Solving in Practice*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012.

[8] Michael Gelfond and Vladimir Lifschitz, 'The stable model semantics for logic programming', in *Proceedings of the Fifth Int. Conf. and Symp. of Logic Progr. (ICLP'88)*, eds., R. Kowalski and K. Bowen, pp. 1070 – 1080. MIT Press, (1988).

[9] Michael Gelfond and Vladimir Lifschitz, 'The stable model semantics for logic programming', in *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, eds., R. Kowalski and K. Bowen, pp. 1070 – 1080. MIT Press, (1988).

[10] William D. Harvey and Matthew L. Ginsberg, 'Limited discrepancy search', in *Proceedings of the 13th International Joint Conf. on Artificial Intelligence*, pp. 607–613. Morgan Kaufmann, (1995).

[11] Matthew D. T. Lewis, Tobias Schubert, and Bernd W. Becker, 'Speedup techniques utilized in modern sat solvers', in *Proceedings of the 8th Int. Conf. on Theory and Applications of Satisfiability Testing*, SAT'05, pp. 437–443, Berlin, Heidelberg, (2005). Springer-Verlag.

[12] Yuliya Lierler, Shaden Smith, Miroslaw Truszczynski, and Alex Westlund, 'Weighted-sequence problem: Asp vs casp and declarative vs problem-oriented solving', in *Proceedings of the 14th Int. Conf. on Practical Aspects of Declarative Languages*, PADL'12, pp. 63–77, Berlin, Heidelberg, (2012). Springer-Verlag.

[13] Veena S. Mellarkod, Michael Gelfond, and Yuanlin Zhang, 'Integrating answer set programming and constraint logic programming', *Annals of Mathematics and Artificial Intelligence*, **53**(1-4), 251–287, (August 2008).

[14] Sanjay Mittal and Felix Frayman, 'Towards a generic model of configuraton tasks', in *11th International Joint Conference on AI - Vol. 2*, IJCAI'89, pp. 1395–1401, San Francisco, CA, USA, (1989). Morgan Kaufmann Publishers Inc.

[15] Max Ostrowski and Torsten Schaub, 'Asp modulo csp: The clingcon system', *Theory Pract. Log. Program.*, **12**(4-5), 485–503, (September 2012).

[16] Tommi Syrjänen, 'Cardinality constraint programs', in *JELIA 2004*, volume 3229 of *Lecture Notes in Computer Science*, pp. 187–199. Springer, (2004).

[17] Erich Christian Teppan, 'Re-/configuring legacy instances of the partner units problem', in *International Conference on Tools with Artificial Intelligence (ICTAI'12)*, pp. 154–161. IEEE, (2012).

[18] Erich Christian Teppan, Gerhard Friedrich, and Andreas Falkner, 'Quickpup: A heuristic backtracking algorithm for the partner units configuration problem.', in *International Conference on Innovative Applications of AI (IAAI'12)*, pp. 2329–2334. AAAI, (2012).

[19] Erich Christian Teppan, Gerhard Friedrich, and Georg Gottlob, 'Tractability frontiers of the partner units configuration problem', *Journal of Computer and System Sciences*, **http://dx.doi.org/10.1016/j.jcss.2015.12.004**, (2016).

Towards Group-Based Configuration

Alexander Felfernig[1] and **Müslüm Atas**[1] and **Thi Ngoc Trang Tran**[1] and **Martin Stettinger**[1]

Abstract. Group-based configuration is a new configuration approach that supports scenarios in which a group of users is in charge of configuring a product or service. In this paper, we introduce a definition of a group-based configuration task and a corresponding solution. Furthermore, we show how inconsistent situations in group-based configuration can be resolved to achieve consensus within the group. We introduce these concepts on the basis of a working example from the domain of (group-based) software release planning.

1 Introduction

Configuration [1, 16] is considered as one of the most successful applications of Artificial Intelligence technologies. It is applied in many domains such as financial services [2], telecommunication [5], and the furniture industry [7]. Configuration environments are typically single-user oriented, i.e., the underlying assumption is that a specific user is in charge of completing the configuration task. However, considering configuration as a single user task can lead to suboptimal decisions [4]. For example, release planning is a task that typically requires the engagement of a group of stakeholders where the knowledge and preferences of all stakeholders should be taken into account in order to be able to achieve high-quality decisions [4, 6].

There are various scenarios where configuration decisions are not taken by a single person but by a group of users [3]. As mentioned, *Software Release Planning* [4] is a requirements engineering related task, where groups of users (stakeholders) are deciding about the ordering in which requirements should be implemented. In this scenario, stakeholders have different preferences and knowledge regarding the implementation alternatives. Consequently, requirements-related knowledge should be exchanged as much as possible and existing contradictions in preferences and evaluations have to be resolved. *Holiday Planning* [8] is another scenario where a group is in charge of identifying a configuration that is accepted by all group members – examples of related decisions are region to visit, hotel, and activities during the stay. *Product Line Scoping* [14] is related to the task of determining boundaries in a product line. This task is a specific type of requirements engineering task and related decisions are crucial for the success of a whole product line effort. *Investment Decisions* (e.g., project funding) [3] are often taken by a group of users who have to take into account constraints with regard to the overall amount of money that can be invested and the topics projects should deal with. The overall configuration task in this context is to identify a bundle of project proposals that takes into account the financial limits and includes high-quality proposals.

Existing configuration environments do not take into account the aspect of group configuration [3]. In contrast, for non-configurable items such as movies, restaurants, personnel decisions, and music, there already exist proposals how to support related group decision processes [11, 12, 15]. In this context, group recommendation heuristics [12] are applied to support groups in their decision making activities. In order to achieve consensus, different decision heuristics are applied which propose decisions acceptable for a group as a whole. For example, the *least misery* heuristic proposes alternatives which do not represent an absolute no-go for at least one of the group members. Besides decision heuristics, standard recommendation approaches [9] such as matrix factorization can be applied to predict recommendations acceptable for a group as whole. These approaches rely on existing group recommendations. Based on such information about group selection behavior, corresponding recommendations can be determined for similar groups.

In this paper, we focus on introducing a formal definition of a group configuration problem and show how inconsistencies in the preferences of group members can be resolved.[2] The remainder of this paper is organized as follows. In Section 2 we introduce a basic definition of a group-based configuration task and introduce a corresponding example configuration knowledge base. In Section 3 we discuss approaches that can help to resolve inconsistencies in the preferences of individual group members. In Section 4 we discuss further issues for future work. With Section 5 we conclude the paper.

2 Group-Based Configuration

In the following, we introduce definitions of a group configuration task and a corresponding solution. These definitions are based on a Constraint Satisfaction Problem (CSP) [17] which is frequently used for the definition of (single user) configuration tasks. The major characteristic of group-based configuration compared to other types of group decision tasks is that the alternatives are defined in terms of a knowledge base, i.e., the alternatives are not pre-specified. This requires new approaches to configuration and diagnosis search, and to represent the configuration task in a corresponding user interface.

Definition 1: Group-based Configuration Task. A group-based configuration task can be defined as a CSP (V,D,C) where V is a set of variables, D represents the corresponding domain definitions, and $C = PREF \cup CKB$ represents a set of constraints. In this context, $PREF = \bigcup PREF_i$ is the union of customer preferences $PREF_i$ and CKB represents a configuration knowledge base.[3]

Definition 2: Group-based Configuration. A group-based configuration (solution) for a group-based configuration task is a complete set of assignments $CONF = \bigcup a_i : v_i = v_{ai}$ to the variables $v_i \in V$ such that $CONF \cup PREF \cup CKB$ is consistent.

[1] Graz University of Technology, Austria, email: {alexander.felfernig,muatas,ttrang,mstettinger}@ist.tugraz.at

[2] The work presented in this paper has been developed within the scope of the WEWANT project (Enabling Technologies for Group-based Configuration) which is funded by the Austrian Research Promotion Agency (850702).

[3] We denote *customer requirements* as *preferences* ($PREFS$) in order to distinguish these from *software requirements* in the working example.

Example 1: Group-based Configuration Task. For demonstration purposes, we introduce a simplified group-based configuration task from the domain of *software release planning*. The goal of software release planning is to assign to each software requirement a corresponding release. In this example, 9 requirements are represented in terms of variables $V = req_1, req_2, .., req_9$ and releases are represented as variable domains. If we assume that *three releases* have been planned for completing the whole software (i.e., implementing each individual requirement), each variable has a corresponding domain [1 .. 3], e.g., dom(r_1) = [1 .. 3]. For the purpose of this example, we assume the existence of *three* stakeholders who are in charge of release planning – $PREF_i$ represents the preferences of stakeholder i.

The following is a complete specification of a group-based configuration task. In this task, the individual user requirements $PREF_i$ are consistent, i.e., a corresponding solution (software release plan) can be identified.[4] The configuration knowledge base CKB includes additional constraints that describe dependencies between different software requirements req_i, for example, $req_1 > req_5$ denotes the fact that requirement req_1 must be implemented for req_5, i.e., there is a dependency between these requirements. Furthermore, the requirements req_3 and req_4 must not be implemented in the same release (e.g., due to resource constraints).

- V = $\{req_1, .., req_9\}$
- D = {dom(req_1) = [1 .. 3], .., dom(req_9 = [1 .. 3])}
- $PREF_1$ = $\{pref_{11} : req_1 = 1, pref_{12} : req_2 = 1, pref_{13} : req_3 = 1, pref_{14} : req_5 = 2, pref_{15} : req_8 = 3\}$
- $PREF_2$ = $\{pref_{21} : req_3 = 1, pref_{22} : req_4 = 2, pref_{23} : req_6 = 3, pref_{24} : req_7 = 3\}$
- $PREF_3$ = $\{pref_{31} : req_5 = 2, pref_{32} : req_6 = 3, pref_{33} : req_8 = 3, pref_{34} : req_9 = 2\}$
- CKB = $\{c_1 : req_1 < req_5, c_2 : req_2 < req_8, c_3 : req_3 < req_6, c_4 : req_3 \neq req_4\}$

Example 2: Group-based Configuration. On the basis of the example group-based configuration task, a constraint solver could determine the following solution: $CONF$ = $\{a_1 : req_1 = 1, a_2 : req_2 = 1, a_3 : req_3 = 1, a_4 : req_4 = 2, a_5 : req_5 = 2, a_6 : req_6 = 3, a_7 : req_7 = 3, a_8 : req_8 = 3, a_9 : req_9 = 2\}$. For each requirement, the constraint solver proposes a corresponding release in the context of which the requirement should be implemented.

3 Resolving Inconsistencies in Group Preferences

In the example introduced in Section 2, the basic assumption is that the preferences of individual group members are *consistent*. However, in group-based configuration scenarios it happens quite often that the preferences of individual users differ. In the context of release planning scenarios, it is often the case that stakeholders have different preferences regarding the implementation of specific requirements. One requirement could be favored due to the fact that the corresponding functionalities are needed by the stakeholder. Another reason could be that a stakeholder has no preferences or simply does not understand the requirement in detail. Inconsistencies between preferences can be manually resolved by showing inconsistent preferences to stakeholders and let them decide which changes should be performed. In such scenarios, minimal conflict sets are determined [10] and conflict resolution is performed by users in a manual fashion.

[4] In Section 3 we discuss approaches to deal with inconsistencies.

Alternatively, conflicts between requirements can be resolved automatically by calculating minimal diagnoses (Definition 4) for minimal conflict sets (Definition 3).

Definition 3: Conflict Set. A conflict set $CS \subseteq \bigcup REQ_i$ is a minimal set of requirements such that inconsistent(CS). CS is minimal if there does not exist a conflict set CS' with CS' is a conflict set and $CS' \subset CS$.

Minimal conflict sets can be exploited for determining the corresponding diagnoses [13]. Assuming that $\bigcup PREF_i \cup CKB$ is inconsistent, a minimal diagnosis (Definition 4) represents a minimal set of requirements that have to be deleted from $\bigcup PREF_i$ such that a solution can be found for the remaining constraints (see Definition 4).

Definition 4: Group-based Configuration Diagnosis Task. A group-based configuration diagnosis task is defined by a group-based configuration task ($V, D, C = PREF \cup CKB$) where $PREF \cup CKB$ is inconsistent.

Definition 5: Group-based Configuration Diagnosis. A diagnosis for a given group-based configuration task ($V, D, C = PREF \cup CKB$) is a set Δ such that $CKB \cup PREF - \Delta$ is consistent. Δ is minimal if $\neg\exists\ \Delta'$: $\Delta' \subseteq \Delta$.

Example 3: Group-based Configuration Diagnosis Task. An example group-based configuration task that includes inconsistencies between different user requirements is the following.

- V = $\{req_1, .., req_9\}$
- D = {dom(req_1) = [1 .. 3], .., dom(req_9) = [1 .. 3]}
- $PREF_1$ = $\{pref_{11} : req_1 = 2, pref_{12} : req_2 = 1, pref_{13} : req_3 = 1, pref_{14} : req_5 = 2, pref_{15} : req_8 = 3\}$
- $PREF_2$ = $\{pref_{21} : req_3 = 2, pref_{22} : req_4 = 3, pref_{23} : req_6 = 3, pref_{24} : req_7 = 3\}$
- $PREF_3$ = $\{pref_{31} : req_5 = 2, pref_{32} : req_6 = 3, pref_{33} : req_8 = 3, pref_{34} : req_9 = 2\}$
- CKB = $\{c_1 : req_2 > req_1, c_2 : req_2 < req_8, c_3 : req_3 < req_6, c_4 : req_3 \neq req_4\}$

In this example, the requirements of the first stakeholder are inconsistent since the combination $req_1 = 2$ and $req_2 = 1$ is inconsistent with the underlying knowledge base ($req_2 > req_1$). Furthermore, there exists an inconsistency between the requirements $req_3 = 1$ (stakeholder 1) and $req_3 = 2$ (stakeholder 2).

The minimal conflict sets that can be derived from our working example are the following: $CS_1 = \{pref_{11}, pref_{12}\}$ and $CS_2 = \{pref_{13}, pref_{21}\}$. The corresponding set of alternative diagnoses (hitting sets) is the following: $\Delta_1 = \{pref_{11}, pref_{13}\}$, $\Delta_2 = \{pref_{11}, pref_{21}\}$, $\Delta_3 = \{pref_{12}, pref_{13}\}$, and $\Delta_4 = \{pref_{12}, pref_{21}\}$. A diagnosis is a minimal set of requirements from $\bigcup PREF_i$ such that $CKB \cup PREF - \Delta$ is consistent.

Diagnoses represent a set of consistency-preserving delete operations that can be applied to the set $\bigcup PREF_i$ in the case that $PREF \cup CKB$ is inconsistent. In many cases, there exist different diagnoses that can be recommended for preserving the consistency between user requirements and the configuration knowledge base (CKB). A ranking of alternative diagnoses in the context of group configuration scenarios can be achieved, for example, by determining a candidate set of minimal diagnoses that is then ranked on the basis of different types of group decision heuristics [12].

An example of the application of such group decision heuristics will be discussed in the following. Table 1 depicts a situation where individual user requirements are inconsistent. In order to resolve this inconsistency, the alternative diagnoses $\Delta_1, \Delta_2, \Delta_3$, and Δ_4 can be applied. An open question in this context is which of the alternative

stake-holder	req_1	req_2	req_3	req_4	req_5	req_6	req_7	req_8	req_9
1	$pref_{11}: req_1 = 2$	$pref_{12}: req_2 = 1$	$pref_{13}: req_3 = 1$		$pref_{14}: req_5 = 2$			$pref_{15}: req_8 = 3$	
2			$pref_{21}: req_3 = 2$	$pref_{22}: req_4 = 3$		$pref_{23}: req_6 = 3$	$pref_{24}: req_7 = 3$		
3					$pref_{31}: req_5 = 2$	$pref_{32}: req_6 = 3$		$pref_{33}: req_8 = 3$	$pref_{34}: req_9 = 2$

Table 1. Tabular representation of constraints in an example group-based configuration task. Conflict set $CS_1 = \{pref_{11}, pref_{12}\}$ reflects inconsistent preferences of stakeholder 1 (the preferences are inconsistent with the configuration knowledge base) and conflict set $CS_2 = \{pref_{13}, pref_{21}\}$ reflects a conflict between the preferences of stakeholders 1 and 2.

diagnoses should be recommended first to the group of users – Table 2 summarizes the impact of the different diagnoses on the current preferences of stakeholders (users). For this purpose, different group decision heuristics can be applied that help to figure out alternatives acceptable for the whole group.

In the following, we exemplify three basic heuristics and show how these can influence the selection of a diagnosis. First, the *least misery* heuristic prefers alternatives (in our case diagnoses) that minimize the misery of individual users (see Formula 1 – $pref_\delta(s, \Delta)$ denotes the number of preferences that have to be changed by user s in the context of diagnosis Δ). In our scenario, least misery for a whole group would reflect the minimum of the maximum number of preferences part of a diagnosis, i.e., the lower the least misery value the better the corresponding diagnosis. For example, if diagnosis Δ_2 is recommended, user 1 would have to adapt two of his/her requirements and user 2 would have to adapt zero. Diagnosis Δ_2 has a lower misery value since the maximum number of requirements to be adapted is 1. Obviously, user 3 is in the situation of not being affected by any of the diagnosis candidates. Second, the *average* heuristic prefers alternatives with the lowest average deviation from the original preferences (see Formula 2). Finally, the *most pleasure* heuristic prefers alternatives with the best outcome for one user (see Formula 3). For example, in Table 1 the most pleasure value of all diagnoses Δ_i is 0.0 since for user 3 there does not exist a need to adapt his/her preferences in all of the diagnoses. For a detailed discussion of group decision heuristics we refer to [12].

$$leastmisery(\Delta) = argmax_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d \quad (1)$$

$$average(\Delta) = \frac{\Sigma_{s \in users} pref_\delta(s, \Delta)}{\#users} \quad (2)$$

$$mostpleasure(\Delta) = argmin_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d \quad (3)$$

stakeholder	$\Delta_1 = \{r_{11}, r_{13}\}$	$\Delta_2 = \{r_{11}, r_{21}\}$	$\Delta_3 = \{r_{12}, r_{13}\}$	$\Delta_4 = \{r_{12}, r_{21}\}$
1	2	1	2	1
2	0	1	0	1
3	0	0	0	0

Table 2. Overview of the impact of the different diagnoses Δ_i on the current preferences of stakeholders, for example, stakeholder 1 has to change two of his/her requirements if diagnosis Δ_1 gets selected.

heuristic	$\Delta_1 = \{r_{11}, r_{13}\}$	$\Delta_2 = \{r_{11}, r_{21}\}$	$\Delta_3 = \{r_{12}, r_{13}\}$	$\Delta_4 = \{r_{12}, r_{21}\}$
least misery	2.0	1.0	2.0	1.0
average	0.67	0.67	0.67	0.67
most pleasure	0.0	0.0	0.0	0.0

Table 3. Evaluation of the different diagnoses using the *least misery*, *average*, and the *most pleasure* heuristic. In all three heuristics the ranking criteria for the diagnoses is *less is better*.

4 Future Work

The major goal of this paper is to present our initial ideas related to the implementation of group-based configuration technologies. There are a couple of issues to be solved within the scope of future work - these issues will be discussed in the following paragraphs.

Consensus in Group Decision Making. Presenting diagnoses in situations where user preferences are inconsistent with the underlying configuration knowledge base and/or the preferences of other users is a basic means to trigger discussions and achieve consensus [4]. However, further aspects have to be taken into account in order to be able to accelerate the achievement of consensus in group decision making. Approaches that are promising in this context are, for example, the following. User interfaces have to be enriched in order to allow basic negotiation mechanisms between users. An example thereof is the following: stakeholder A is interested in having implemented requirement req_a as soon as possible. Furthermore, stakeholder B is interested in having implemented requirement req_b as soon as possible. Stakeholder A would accept an earlier implementation of req_b if stakeholder B accepts an earlier implementation of requirement req_a. In this context, visualization concepts for the representation of the current decision situation will play a major role – alternative ways to represent decision situations are a focus of future work.

Fairness in Group Decision Making. An important issue in group decision making is fairness with regard to group members. Fairness is especially a topic within the scope of repeated decision processes where the same or similar groups are taking a decision. A related example is holiday decisions where a group of friends decides about a new travel destination and related activities. The preferences of users who were discriminated to some extent in previous year's travel arrangements should have a higher emphasis in the new holiday decision. Fairness also includes visualization aspects since the visualization of the current state of the decision process could help to increase fairness in group decision making, for example, by increasingly taking into account the preferences of other group members.

Predictive Search. Based on the information about already completed group decision processes, diagnosis and repair could be im-

proved by better predicting alternatives acceptable for the whole group. In this context, different types of personalization approaches should be included that help to take into account the preferences of the whole group when determining diagnoses and corresponding repair actions. Diagnosis prediction approaches for single users are already discussed in related work [1], however, in group decision scenarios further related aspects have to be taken into account. The prediction of a relevant diagnosis does not only have to take into account the selection behavior of users but also how users interacted with each other within the scope of a group decision process. Furthermore, the search for alternative configurations has to take into account group preferences, i.e., search heuristics must be learned on the basis of past group interactions.

Negotiation Mechanisms. The main challenge of negotiation mechanisms is to include these in a way that is easy to understand for users. Complex negotiation mechanisms will not be accepted by end-users, i.e., the major challenge is to propose decision and negotiation mechanisms that help to achieve high-quality decisions and consensus as soon as possible and to trigger inconsistency management only in situations where real disagreements exists. For example, if one stakeholder evaluates the risk level of a requirement with 7 (on a scale [1..10]) and the other stakeholder evaluates the same requirement with 8, there seems to be no real disagreement and the system may not have to point out an existing inconsistency.

Intelligent User Interfaces. Since group-based configuration tasks are solved in a distributed and asynchronous fashion, user interfaces should be able to take into account this situation. Figure 1 includes a screenshot of the CHOICLA group decision support environment [15].[5] In its current version, the system supports group decisions related to non-configurable products and services (e.g., party locations and type of dinner), i.e., decisions are taken with regard to a collected assortment of alternatives but are not taken with regard to certain attributes (variables) which are basic elements of a configuration task. In the current version of CHOICLA, the only possibility of taking decisions regarding configurable products is to enumerate a representative set of alternatives (e.g. new family car). In future versions of CHOICLA, we will support the integration of complete configuration tasks into decision processes. Variables will then be represented as alternatives and user preferences and inconsistencies will be represented on a corresponding graphical level.

5 Conclusions

In this paper, we introduced the concept of *group-based configuration*. We introduced a basic definition of a group-based configuration task (represented as a constraint satisfaction problem) and showed how to deal with inconsistent preferences of group members on the basis of the concepts of model-based diagnosis. In this context, we showed how to integrate different types of decision heuristics into diagnosis selection processes. Finally, we discussed different challenges for future work we want to tackle.

REFERENCES

[1] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann Publishers, 1st edn., 2014.

[2] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, 'The VITA Financial Services Sales Support Environment', in *AAAI/IAAI 2007*, pp. 1692–1699, Vancouver, Canada, (2007).

[5] www.choicla.com.

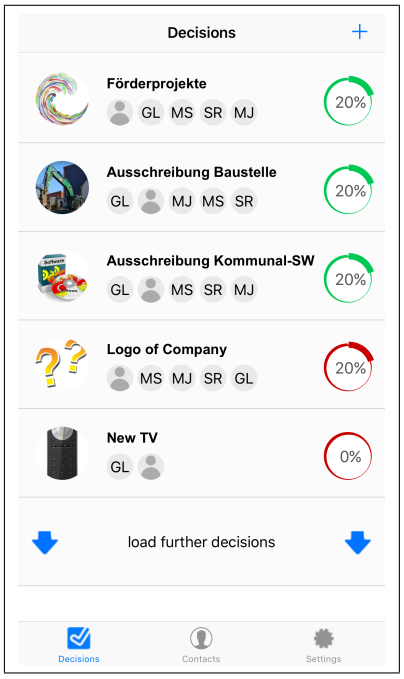


Figure 1. CHOICLA group decision support environment. Each entry represents a group decision task – the corresponding percentages indicate the share of users who already articulated their requirements. A red circle indicates the fact that the current user did not articulate his/her preferences.

[3] A. Felfernig, M. Stettinger, G. Ninaus, M. Jeran, S. Reiterer, A. Falkner, G. Leitner, and J. Tiihonen, 'Towards open configuration', in *16th Intl Workshop on Configuration*, pp. 89–94, Novi Sad, Serbia, (2014).

[4] A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maaleij, D. Pagano, L. Weninger, and F. Reinfrank, 'Group Decision Support for Requirements Negotiation', in *Advances in User Modeling, Springer Verlag*, volume 7138 of *LNCS*, pp. 105–116, (2012).

[5] Gerhard Fleischanderl, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner, 'Configuring large systems using generative constraint satisfaction', *IEEE Intelligent Systems*, **13**(4), 59–68, (1998).

[6] T. Greitemeyer and S. Schulz-Hardt, 'Preference-consistent evaluation of information in the hidden profile paradigm: Beyond group-level explanations for the dominance of shared information in group decisions.', *Jrnl of Personality & Soc Psychology 84(2)*, 332–339, (2003).

[7] A. Haag, 'Sales Configuration in Business Processes', *IEEE Intelligent Systems*, **13**(4), 78–85, (1998).

[8] A. Jameson, S. Baldes, and T. Kleinbauer, 'Two methods for enhancing mutual awareness in a group recommender system', in *ACM Intl. Working Conf. on Advanced Vis. Interf.*, pp. 48–54, Gallipoli, Italy, (2004).

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems – An Introduction*, Cambridge University Press, 2010.

[10] U. Junker, 'QuickXPlain: Preferred Explanations and Relaxations for Over-Constrained Problems', in *19th National Conference on AI (AAAI04)*, pp. 167–172, San Jose, CA, (2004).

[11] J. Masthoff, 'Group modeling: Selecting a sequence of television items to suit a group of viewers', *UMUAI*, **14**(1), 37–85, (2004).

[12] J. Masthoff, 'Group recommender systems', *Recommender Systems Handbook*, 677–702, (2011).

[13] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).

[14] K. Schmid, 'Scoping software product lines', in *Software Product Lines – Experience and Research Directions*, pp. 513–532, (2000).

[15] M. Stettinger, 'Choicla: Towards domain-independent decision support for groups of users', in *8th ACM Conference on Recommender Systems*, pp. 425–428, (2014).

[16] M. Stumptner, 'An overview of knowledge-based configuration', *AICOM*, **10**(2), 111–125, (1997).

[17] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.

Towards Configuration Technologies for IoT Gateways

Alexander Felfernig[1] and **Seda Polat Erdeniz**[1] and **Paolo Azzoni**[2] and **Michael Jeran**[1] and **Arda Akcay**[1] and **Charalampos Doukas**[3]

Abstract. The AGILE project aims to create Internet of Things (IoT) gateway technologies that support many devices, protocols, and corresponding administration and software development activities. In this context, there are scenarios that require the support of configuration technologies. The major goal of this short paper is to provide an overview of application scenarios and related configuration technologies that will be developed within the scope of the AGILE project.

1 Introduction

Configuration is a process in which agents (users or external systems) can specify requirements and the configuration system (often denoted as the *configurator*) provides feedback in terms of solutions and/or explanations [6, 17, 18]. Configuration can be interpreted as a type of design activity where a product is composed (configured) from a set of instances corresponding to predefined component types such that the resulting configuration (solution) is consistent with a given set of constraints [17, 18]. Requirements can be regarded as specifications of intended properties of the product (i.e., specific constraints), for example, a specific *application* or *mail server version* should be included in the *system configuration*. In such contexts, system feedback for a user is provided in terms of configurations, reconfigurations, and explanations for situations in which no solution could be found. There is a multitude of application examples of knowledge-based configuration, for example, in the automotive domain, railway interlocking systems, financial services, operating systems, and software product lines [6].

Configuration services for the Internet of Things (IoT) domain [2] is a new application area. The IoT is an emerging paradigm that envisions a networked infrastructure enabling different devices (things) to be interconnected at anyplace and anytime. In this paper we discuss two basic scenarios that will be supported by software components to be developed in the AGILE research project.[4] First, *ramp-up configuration* services will be developed that help to determine an initial configuration for the whole IoT gateway infrastructure. One of the major tasks of such gateways[5] is to bridge devices to corresponding applications on the basis of different communication protocols such as Hue and Zigbee. For example, in the smarthome domain, a configuration would determine the set of sensors, connection protocols, and apps needed to make a gateway operable for the user. Second, we will develop technologies that help to *optimize configuration and reconfiguration* of communication protocols in such a way that user requirements (e.g., performance requirements) and side conditions (e.g., available bandwith) can be taken into account.

The contributions of this paper are the following. First, we introduce *Internet of Things* (IoT) as a new application domain of knowledge-based configuration technologies. Second, we provide an overview of example IoT application scenarios that are in the need of configuration support. Finally, we summarize research objectives and technological approaches that will be followed in the AGILE project.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of basic configuration functionalities that will be provided in the context of AGILE ramp-up configuration scenarios. In Section 3, we focus on a runtime scenario in which we want to optimize the selection of communication protocols with regard to optimization criteria. In Section 4 we provide an overview of related work. With Section 5 we conclude the paper.

2 "Ramp-Up Configuration" in AGILE

AGILE gateways will be deployed in different domains such as *health monitoring*, *animal monitoring in wildlife areas*, *air quality and pollution monitoring*, *enhanced retail services*, *smart homes*, and *port area monitoring*. Each application scenario requires a preconfiguration which estimates the needed hardware and software components / devices to be deployed in the ramp-up phase of the system. We denote this type of configuration *ramp-up configuration* since each scenario requires a specific set of hardware components and software components (including apps) to "ramp-up" the system.

AGILE Air Pollution Monitoring. Environmental pollution has become an issue of serious international concern and is increasingly stimulating the development and adoption of solutions to monitor and reduce the effects of pollution. This is an interesting and challenging market, with both potential economical outcomes and a strong societal impact. The convergence of hardware integration, reduction of sensor costs, IoT and M2M technologies introduces a new panorama where it is really possible to deliver low cost, high quality monitoring systems with a capillary coverage of the territory. This convergence leads to a new era of solution for environmental pollution monitoring. Air quality and pollution monitoring stations are complex systems that, depending on the application context, deliver added value pollution monitoring services based on a delicate equilibrium between the adoption of the most appropriate sensors, their correlation, the selection of the correct algorithms and the configuration of their hardware and software parameters. A wrong or imprecise selection and configuration of these elements leads to misleading, wrong, and completely useless results and services. *Air Pollution Monitoring* is in the need of configuration support since the measuring equipment has to be pre-selected and parametrized in the

[1] Graz University of Technology, Austria, email: {alexander.felfernig, spolater, mjeran, aakcay}@ist.tugraz.at
[2] Eurotech Group, Italy, email: paolo.azzoni@eurotech.com
[3] Create-Net, Italy, email: cdoukas@create-net.org
[4] AGILE (An Adaptive & Modular Gateway for the IoT) is an EU-funded H2020 project 2016–2018 – see agileiot.eu.
[5] *Raspberry Pi* is one of the hardware platforms used in AGILE – see www.raspberrypi.org.

line of the environmental conditions that exist, for example, in a city.

Further AGILE Scenarios. The basic task of a configurator in *health monitoring* is to figure out which measuring devices are needed (including their parametrization) to be able to monitor and analyze specific body functions. In the *animal monitoring* scenario it is important to figure out which infrastructure can be used to complete predefined data collection tasks. In such scenarios, *reachability* of animals (and corresponding sensors) plays a major role in order to be able to complete data collection. Reachability depends on the selected drone types but also on the selected communication protocols which have different degrees of power consumption. *Enhanced retail services* that allow a personalized shopping experience in physical stores are in the need of configuration functionalities that indicate the amount and positioning of sensors (e.g., for indoor position detection) and displays that are needed to successfully support customers in their shopping experiences. In the *port area monitoring* scenario, configuration technologies are needed that help to select relevant sensors (e.g., gas, radioactivity, and water quality sensors) that are able to provide the needed data. In *smarthome* scenarios, the task of the configurator is to identify sensors, communication components, and protocols that are needed to provide the smarthome functionalities required by a customer. In this context, examples of customer requirements are rooms in the house and their type, maximum accepted price, and needed functionalities (e.g., presence monitoring and simulation, and video surveillance).

Knowledge Acquisition & Representation. When configuring, for example, *smart homes*, the configuration model includes information about the relationships between building properties and corresponding sensors (e.g., *if a room is a kitchen and includes an oven, then a corresponding temperature sensor has to be included for the room*) or between user preferences and the corresponding technical infrastructure (e.g., *if a user wants to save money, wireless communication is preferred*). A configuration for a given configuration task includes information about which components, devices, and drivers are part of the initial gateway installation.

In AGILE, we will evaluate the applicability of different types of configuration knowledge representations such as answer set programs (ASP) [13] and constraint-based representations [6, 9, 20]. Our aim is to *identify a knowledge representation language* that can be applied for each of the different application scenarios in order to provide a basic technology for supporting IoT ramp-up configuration tasks. The applicability of these languages will be primarily evaluated with regard to expressiveness and reasoning efficiency. Especially, ASP-based configuration approaches will be evaluated with regard to their applicability in typical gateway ramp-up scenarios.

Consistency Management of Knowledge Bases. Configuration knowledge bases can become inconsistent, i.e., the defined component types and constraints lead to the problem that no solution can be identified. Such a situation can occur in the context of regression testing [4] but also in situations where the conflict is induced by the configuration knowledge base itself. In such scenarios, configuration technologies in combination with model-based diagnosis [16] can be exploited to automatically identify the sources (e.g., constraints) of a given inconsistency [4]. Such functionalities will be included in a development environment for IoT configuration knowledge bases.

In the context of AGILE, we focus on the development of techniques that help to improve the efficiency of configuration knowledge engineering processes. Although automated debugging [4] is a useful means to reduce time efforts related knowledge base development and maintenance, the development and maintenance of related test cases (also denotes as examples [4]) is still costly. We will analyze the applicability of different testing approaches from software engineering and will especially focus on the the development of *mutation testing* approaches for knowledge bases [10]. In this context, a mutation will serve as a basis for generating tests that are, for example, accepted by the original knowledge base but should not.

Consistency Management of User Requirements. Consistency management not only plays a role in the context of knowledge base development and maintenance but also within the scope of a configuration process. A user of an AGILE configurator could articulate a set of requirements in such a way that no solution can be identified. Also in such a situation, model-based diagnosis approaches can be exploited to indicate sets of user requirements that have to be adapted such that at least one solution can be identified [4, 7, 12, 21]. A similar situation occurs in the context of reconfiguration, i.e., in a situation where hardware and software components of an IoT gateway have to be adapted. In this context, minimal changes have to be proposed that indicate how the existing configuration has to be adapted such that a consistent configuration can be determined that takes into account all reconfiguration requirements [8].

In AGILE, we focus on the development of personalization techniques that help to improve the diagnosis prediction quality, i.e., to identify those diagnoses that will be accepted by the user. Such personalized diagnoses will be determined on the basis of an analysis of the interaction behavior of users of similar gateway installations (available in gateway profile repositories). In this context we will develop learning-based approaches that help to calibrate search heuristics in order to improve efficiency and prediction quality of configuration and reconfiguration.

A simple example of our envisioned approach is the following. Let us assume the existence of a *configuration log* as the one shown in Table 1. The parameters req_i indicate user requirements and x_i indicate technical product parameter settings (consistent with the user requirements) accepted by the user u_i. The overall goal is to optimize the configurator search heuristics (e.g., variable and domain orderings) in such a way, that the prediction quality for the technical parameter settings is maximized. More precisely, we want to identify search heuristics that guide to solutions (configurations) that will be accepted by the *current* user. User interactions (see, e.g., Table 1) serve as a basis for learning. Prediction quality can be measured, for example, in terms of the user acceptance degree of parameter settings (configurations) proposed by the configurator. In this context we will evaluate different clustering techniques, i.e., to learn heuristics not on a global level, but depending on a specific cluster derived, for example, from the user requirements.

$user$	req_1	req_2	x_1	x_2	x_3	x_4
u_1	1	2	3	4	4	2
u_2	2	2	8	3	4	2
u_3	1	2	3	4	5	2
$current$	1	1	?	?	?	?

Table 1. Example configuration log as a basis for optimizing the prediction quality of configuration parameters (req_i represent user requirements and x_i represent technical product parameter settings accepted by previous users).

3 Runtime Configuration in AGILE

Modern embedded systems included in IoT scenarios support a rich set of connectivity solutions (e.g., 3G, LTE, TD-LTE, FDD-LTD, WIMAX, and Lora). In this context, configuration technologies play

an important role in terms of suggesting optimal connectivity configurations. Such configurations include a collection of connectivity solutions that are needed to support a set of active applications (apps). Criteria that have to be taken into account are, for example, location information, available connectivity, performance and reliability requirements, contractual aspects, and costs.

In AGILE, runtime configuration must be performed on the gateway – in contrast, ramp-up configuration can also take place in the cloud. On the one hand we will evaluate different types of reasoning engines, for example, the CHOCO constraint solver[6] and the Sat4j boolean satisfaction library[7]. We will also take into account the application of rule engines[8], optimization libraries, and knowledge compression techniques [1] to assure efficiency of problem solving on the gateway level.

In AGILE, gateway configurations can be manually defined by users but also be determined on the basis of a configurator that is in charge of keeping the overall system installations consistent. A configurator (e.g., a constraint solver) can determine alternative configurations which have to be ranked. In order to determine a ranking for alternative configurations, a MAUT[9] approach can be used [22]. Examples of evaluation *dimensions* (dim) used in MAUT could be *performance*, *reliability*, and *costs*. Depending on the current gateway configuration and the usage context, a configurator can determine alternative (re-)configurations and rank them accordingly.

A simplified example of the application of a utility-based approach is the following. Table 2 includes an evaluation of connectivity protocol configurations $conf$ ($conf_a$ and $conf_b$) to be used on the gateway, for example, for different types of data exchange. The three evaluation dimensions used in this example are performance, reliability, and costs. Furthermore, Table 3 includes the personal preferences of two different gateway users (u_1 and u_2).

In order to determine the configuration that should be chosen for a specific user, we can apply a utility function (see, e.g., Formula 1).

$$utility(conf, u) = \Sigma_{d \in dim} interest(u, d) \times value(conf, d) \quad (1)$$

In this context, *utility(conf,u)* denotes the utility of the configuration $conf$ for the user u, *interest(u,d)* denotes the interest of user u in evaluation dimension d, and *value(conf,u)* denotes the contribution of configuration $conf$ to the interest dimension d. In the example, configuration $conf_a$ has a higher utility for user u_1 (107.0) whereas configuration $conf_b$ has a higher utility for u_2 (111.0). Note that for simplicity we omitted to sketch the determination of the evaluations depicted in Table 2 – for details see [5]. In order to increase the efficiency of runtime configuration, we will evaluate knowledge compression techniques that help to reduce search efforts as much as possible. For example, we will apply decision diagram techniques [1] to pre-calculate possible configurations and re-configurations.

Table 4 provides a summary of the configuration-related research objectives in AGILE. Within the context of ramp-up configuration scenarios we will identify knowledge representation mechanisms that allow an easy representation of the AGILE IoT domains introduced in Section 2. Furthermore, we will develop test case generation techniques that will help to make the development and management of test cases more efficient. For AGILE scenarios, we will develop concepts that support the learning of search heuristics to optimize configuration and reconfiguration processes. Furthermore, we will work on knowledge compression techniques [1] that help to make solution search on the gateway level as efficient as possible.

[6] choco-solver.org.
[7] sat4j.org.
[8] java-source.net/open-source/rule-engines.
[9] Multi-attribute utility theory.

4 Related Work

Although different from basic IoT scenarios [2], there exist applications that support the configuration of systems including hardware and software components. Falkner and Schreiner [3] introduce approaches to the configuration of railway interlocking systems as examples of complex industrial systems designed on the basis of constraint-based configuration technologies. Krebs et al. [11] show the application of configuration technologies in the area of car periphery supervision that includes detection of the car environment, the recognition of hazardous situations, and the handling of difficult traffic situations. Related applications are pre-crash detection, the detection of obstacles, and parking assistance. Related car configuration processes have to take into account existing hardware components and to combine these with the corresponding software units. Finally, Perera et al. [15] introduce an approach to the end-user-oriented configuration of IoT middleware components.

The afore mentioned approaches are in the line of the mentioned "ramp-up" scenario, i.e., infrastructures are configured before the system is operable. In contrast to the developments in [3, 11, 15], the "ramp-up" configuration approach that is currently developed in AGILE focuses on advanced testing methods for supporting configuration knowledge engineering and also on approaches to improve configurator usability by the inclusion of different types of personalized consistency restoration methods. Initial approaches to include recommendation methods into configuration problem solving are documented, for example, in [19]. These approaches do not take into account the issue of consistency management in a satisfactory fashion which will be a major focus of our work in the AGILE project. Finally, for an overview of different IoT smart solutions available on the market we refer to [14].

5 Conclusions and Future Work

In this paper we provide a short introduction to basic configuration scenarios of the AGILE project. We discussed the two scenarios of "ramp-up configuration" and "runtime optimization". Major challenges for our future work will be approaches to automated test case generation for configuration knowledge bases, efficient techniques to solve the "no solution can be found" problem in interactive configuration settings, and the personalization of related repair approaches.

REFERENCES

[1] H. Andersen, 'An introduction to binary decision diagrams', in *Lecture Notes for Efficient Algorithms and Programs*, pp. 1–35, (1999).
[2] L. Atzori, A. Iera, and G. Morabito, 'The Internet of Things: A survey', *Computer Networks*, **54**(15), 2787–2805, (2010).
[3] A. Falkner and H. Schreiner, 'SIEMENS: Configuration and Reconfiguration in Industry', in *Knowledge-based Configuration – From Research to Business Cases*, eds., A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, pp. 199–210. Morgan Kaufmann, (2014).
[4] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based diagnosis of configuration knowledge bases', *Artificial Intelligence*, **152**(2), 213–234, (2004).
[5] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker, 'An environment for the development of knowledge-based recommender applications', *International Journal of Electronic Commerce (IJEC)*, **11**(2), 11–34, (2006).
[6] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann Publishers, 1st edn., 2014.

configuration	performance	reliability	costs
$conf_a$	9	5	2
$conf_b$	5	8	3

Table 2. Utility table: evaluation of configurations with regard to the interest dimensions *performance*, *reliability*, and *costs*.

user	performance	reliability	costs
u_1	10	3	1
u_2	5	7	10

Table 3. Example user preferences w.r.t. interest dimensions *performance*, *reliability*, and *costs*.

configuration topic	research objective
appropriate knowledge representations	knowledge representations for easy modeling and efficient configuration search
efficiency of knowledge base development and maintenance	automated test case generation and mutation testing
personalized consistency management	personalized configuration based on learning search heuristics and knowledge compression techniques

Table 4. Overview of AGILE research objectives.

[7] A. Felfernig, M. Schubert, and C. Zehentner, 'An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets', *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, **25**(2), 175–184, (2011).

[8] A. Felfernig, R. Walter, and S. Reiterer, 'FlexDiag: AnyTime Diagnosis for Reconfiguration', in *16th International Workshop on Configuration*, pp. 105–110, Vienna, Austria, (2015).

[9] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner, 'Configuring large systems using generative constraint satisfaction', *IEEE Intelligent Systems*, **13**(4), 59–68, (1998).

[10] Y. Jia and M. Harman, 'An Analysis and Survey of the Development of Mutation Testing', *IEEE Transactions on Software Engineering*, **37**(5), 649–678, (2011).

[11] T. Krebs, L. Hotz, and A. Günter, 'Knowledge-based Configuration for Configuring Combined Hardware/Software Systems', in *Proceedings of PuK'2002*, pp. 1–6, Freiburg, Germany, (2002).

[12] J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov, 'On computing minimal correction subsets', in *IJCAI*, pp. 615–622, (2013).

[13] V. Myllärniemi, J. Tiihonen, M. Raatikainen, and A. Felfernig, 'Using answer set programming for feature model representation and configuration', in *Workshop on Configuration*, pp. 1–8, Novi Sad, (2014).

[14] C. Perera, C. Liu, and S. Jayawardena, 'The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey', *IEEE Transactions on Emerging Topics in Computing*, **3**(4), 585–598, (2015).

[15] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos, 'Semantic-driven configuration of internet of things middleware', in *9th International Conference on Semantics, Knowledge & Grids (SKG)*, pp. 66–73, Beijing, China, (2013).

[16] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).

[17] Daniel Sabin and Rainer Weigel, 'Product configuration frameworks - a survey', *IEEE Intelligent Systems*, **13**(4), 42–49, (1998).

[18] M. Stumptner, 'An overview of knowledge-based configuration', *AICOM*, **10**(2), 111–125, (1997).

[19] J. Tiihonen and A. Felfernig, 'Towards Recommending Configurable Offerings', *International Journal of Mass Customization*, **3**(4), 389–406, (2010).

[20] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.

[21] R. Walter, A. Felfernig, and W. Küchlin, 'Constraint-based and sat-based diagnosis of automotive configuration problems', *Journal of Intelligent Information Systems*, 1–32, (2016).

[22] D. Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*, Cambridge University Press, 1986.

Towards Modularization and Configuration of Services – Current Challenges and Difficulties

Thorsten Krebs[1] and **Aleksander Lubarski**[2]

Abstract. The megatrend individualization forces companies to diversify and tailor their products up to lot size one. However, in order to stay competitive they need to increase the overall efficiency by standardizing their internal processes, which leads to a conflict of interest between a company and its customer. The principle of modularization gives a possible solution for wide market coverage with a sufficiently large number of product variants without major investments. Modularization and configuration are already widely used in manufacturing and software development, but the service domain still seems to be at the beginning of its development, even though service providers face similar challenges when offering individualized services. Additionally, traditional product manufacturers are more often investing into services for being able to cope with saturated and commoditized global environments, striving for attracting and retaining customers. This paper presents a problem description and as such it addresses the challenges and difficulties for the emerging service modularity. We elaborate on the commonalities and differences between modularization and configuration of tangible products and services and present first ideas towards modularization and configuration of services. The insights are based on a number of case studies with local companies that were carried out in a joint research project (http://www.bakerstreet.uni-bremen.de).

1 INTRODUCTION

In advanced economies, the service sector continuously gains importance. Not only service firms, whose core competencies are the provision of services, generate a high portion of the economic output, but also manufacturers recognize the importance of services for their business success [1]. The service-dominant logic (SDL) has become predominant in the modern service provision, underlying a close interaction between provider and customer, especially in business-to-business (B2B) services.

In the areas of consumer and capital goods we have seen how customers became more demanding regarding the individualization of their requests, which, in turn, forced product manufacturers to offer more and more variants of their products. In mass customization scenarios, product configuration based on a modular strategy is an enabler for this trend [2]. In general terms, modularity can be seen as a principle of building a complex system from smaller parts that can be designed and improved independently, yet function together as a whole [3]. Linked to "modularity" as the basic concept, "modularization" denotes the actual transformation process, and a "modular architecture" is the desired result of it [4]. The idea of loose coupling results in interchangeability of modules (separate service components, e.g., packaging of a machine before transport) and flexibility (substitution of one module for another, without affecting the main service, e.g. contract logistics), as long as the interfaces between separate modules are well-defined and standardized and a clear one-to-one matching of modules and functions exists [5].

Manufacturers are adapting their business models and information systems according to increasing customer demands. As a result, product diversifications leads to increased use of modular systems as a basis for product configuration [6]. While product configuration and modularization strategies are well understood and applied in the areas of consumer goods and capital goods [7], the discussion about service modularity remains mostly theoretical with little application, even though service providers face similar challenges when offering individualized services (Figure 1).

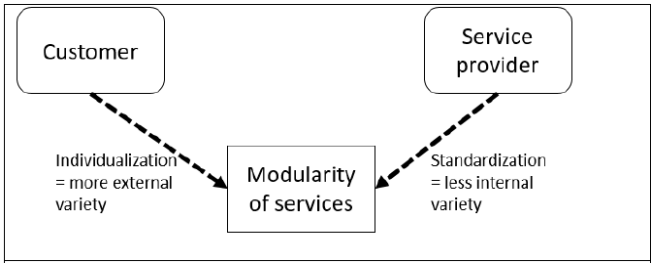


Figure 1. Need for service modularity

However, the concept of service modularity and its potential implementation has attracted academic attention of both service design researchers as well as information system society, thus becoming one of the central topics of international conferences and academic journals. Service design researchers like Dörbecker and Böhmann [8] emphasize the need of examining, whether the concept of modularity can be actually useful for practical purposes in the service sector. Similarly, it has been underlined that "so far the literature does not provide clear guidelines for how to accomplish modular service design and development" [9].

The purpose of this paper is therefore to start a discussion on the service modularity from the practical point of view and its possible implementation with the help of configurators. We elaborate on commonalities and differences between modularization and configuration of tangible products and services, concentrating on the resulting challenges and difficulties and we present first ideas towards modularization and configuration of services.

[1] encoway GmbH, Buschhöhe 2, 28357 Bremen, email: krebs@encoway.de
[2] Universität Bremen, Wilhelm-Herbst-Str. 5, 28359 Bremen, email: lubarski@is.uni-bremen.de

The remainder of this paper is organized as follows. Section 2 gives an overview of existing work in the field of service modularity. Section 3 describes the methodology of the case study research, followed by the conclusions based on the BakerStreet project in Section 4. The findings are compared to what is known from the modularization and configuration of tangible products. Finally, Section 5 summarizes the paper.

2 RELATED WORK

In general, it seems that the concept of service modularity still remains in its infancy both in terms of academic discussion and practical application [10]. Previous studies have discussed the concept of modularity in general [11]–[13], conducted literature reviews [9] or assessed modularity potentials for specific application scenarios [14], [15]. A good overview on existing work on service modularity, regarding its relevant definitions as well as discussion of positive and negative effects, can be found in the literature review of [8]. Similarly, in their conceptual paper, Müller and Lubarski [16] developed seven dimensions on how relevant literature on service modularity can be structured and introduce four schools of thought, to show how the concept is evolving in terms of SDL and business model transformation.

Service modularity has been also recognized as a tool for strategic positioning of the firm. For instance, Carlborg and Kindström [10] apply the idea of the SDL (i.e. intensity of customer participation in the service provision) combined with different types of service processes and recommend different modular strategies depending on the types (Figure 2).

		ROLE OF CUSTOMER	
		PASSIVE	**ACTIVE**
SERVICE PROCESS	**RIGID**	**Type 1** - Key issue: Internal efficiency through a high degree of standardized modules - Example: automatic gas supply - High level of process formalization - Primarily provider-driven resources - Low level of technical skills - Modular strategy: Bundled	**Type 2** - Key issue: Modularizing the customer's own experience - Example: online tool - High level of process formalization - Customer-driven resources - Low level of technical skills - Modular strategy: Pre-defined bundled offerings
	FLUID	**Type 3** - Key issue: Interaction of modules - Example: Remote monitoring - Provider-driven resources - High level of technical skills - Modular strategy: Flexible bundling	**Type 4** - Key issue: Interaction of modules and supporting resources - Example: Operator-driven reliability - Provider- and customer-driven resources - High level of technical skills - Modular strategy: Unbundled modules

Figure 2. Service types and modular strategies [10]

They present their typology as a matrix of two dimensions. The first dimension distinguishes service processes as being either rigid or fluid. Rigid service processes are characterized by a high level of formalization, standardization and low task variety, and a relatively low level of information exchange between service provider and customer. On the other hand, fluid service processes require a high level of technical skill and information exchange and exhibit a high task variety. While the activities of rigid service processes are mainly directed to the customer's possessions (e.g., equipment or material), the activities of fluid service processes are mainly directed towards the customer processes. The second dimension distinguishes between the modes of customer participation in the service process, which can vary from passive (i.e., employees of the service firm produce the service) or active (i.e., customer action is required). The four cells of the matrix (Figure 2) represent four different service types that are each characterized by different key issues in modularity, supporting resources and modular strategies [10]. While such a typology marks a first step in introducing service modularity to the service providers, it still gives no practical guidance on its implementation or evaluation methods, in particular in terms of the sales process.

Finally, additional attention has been directed towards modularization methods, which are either adapted from product modularity [17] or designed specifically for services [18]. However, even though several individual modularization methods have been introduced and applied in case studies [19], they were not able to cover the whole modularization process, but instead concentrated on single phases only (e.g., decomposition of monolithic service offerings, module creation). In addition, most of the existing methods make simplified assumptions, which are not necessarily valid for the real service providers (e.g., existence of an already well-defined and clearly decomposed service portfolio, or a comprehensive transparency over service delivery processes). The first classification of the existing methods for service modularization based on their specific characteristics was given by Lubarski and Pöppelbuß [4], who used two dimensions for structuring (Figure 3).

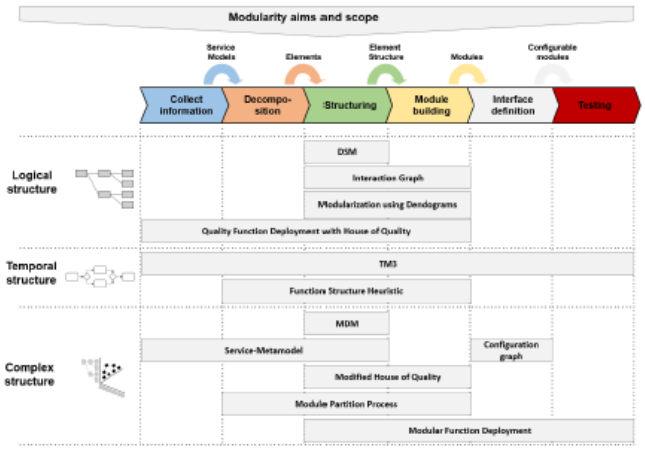


Figure 3. Process model for service modularization [4]

The first dimension presents different stages of the modularization process, which are to be completed within a restructuring initiative, beginning with the information capturing about the existing service portfolio of the service provider up to the test phase of the final modularized construction kit, including modules and rules for their configuration. The result of this process is a modular architecture comprised of configurable modules, which can be operated by users (e.g., sales personnel or customers) using appropriate configuration tools. The second dimension analyzes how modules are structured, differentiating between (1) logical structures (widely used in the manufacturing industry, where the composition of a product from its static components is described), (2) temporal structures (more suitable for the service domain due to the process nature) and (3) complex structures (arbitrary combination of logical and temporal structures).

Several case studies have confirmed that there is a current need for a modular service portfolio amongst practitioners, especially in the healthcare [19], [20], remote monitoring systems [10] or logistics sector [15], [21]. Therefore, the idea of an IT-supported configurator that builds on a modular architecture seems very promising. Such approaches would allow the customer to get an overview on possible variants and their corresponding prices, which would be novel particularly in the B2B environment that is characterized by a high level of complexity of services, information asymmetry regarding the service portfolio and price calculation, and the effortful tender preparation processes.

3 CASE STUDIES

The decomposition of services and their subsequent aggregation into modules are likely to cause organizational and process changes, which are best observed in practical application. In order to gather useful insights, methods of qualitative empirical research shall be employed. Qualitative research is especially suitable when the research area is still emerging and not controllable by the investigators [22], which is the case with service modularization. To achieve a comprehensive solution for each of the research objectives, the service providers in the case studies should be as heterogeneous as possible. Therefore, this research concentrated on potential partners from key industries in Bremen – logistics and wind energy services. These industries are different in the nature of the services offered and service provision, but they face similar challenges in terms of developing offers that are both flexible and customizable to fit the specific requirements of the customer.

3.1 Preparation and interviews

Therefore, as part of the research project BakerStreet we have carried out 18 interviews with service providers; 9 companies from the area of wind energy and 7 companies from the area of logistics. The companies are varying in size such that they range between 50 and 8000 employees and between an annual revenue between 2 million and 5.5 billion Euros. The guiding questions addressed the sales process of the firms as well as the characteristics of the actual service processes. In line with the subject area of this research, interviewees from organizations providing these services were selected. The interviews were conducted in September and October 2015 and lasted from 20 to 45 minutes each. All interview partners requested to remain anonymous. We used MAXQDA 12 for analyzing the data. With four of the interviewees we made an in-depth case study, carried out as a half day workshop with a number of relevant persons from that companies, such as sales reps and product managers.

Since we investigated companies from different business areas the results are only partially comparable. But the areas of logistics and wind energy also have a lot in common. In addition, we also made a case study with a company selling construction and engineering services in the automotive industry and gained insights from discussions with two companies from the areas of printing machines and marking systems.

3.2 Results

What all these service providers have in common is that the complexity behind configuration of services from a customer point of view is rather a minor problem. The major difficulty in setting up modular service portfolios is the influence of the customer on the individual shaping of how the service needs to be carried out. This, in addition, leads to rather complex pricing strategies.

The impact of individual customer situations depends on the type of business. For logistics main variant drivers are source and target location and size, weight and type of goods to be transported. For wind energy the plant location and turbine manufacturers are important. For companies selling combined product service systems, the type of product has impacts on the available services as well as individual shaping a service. Services carried out based on products already in use require additional information about the product's age, how it was used and serviced in the past.

In addition, service providers often need to manage knowledge about past quotes and orders. Often every customer inquiry is executed individually, independent of the question whether the same or at least a similar inquiry was completed before. Hence, service providers would largely benefit from a system supporting the retrieval of past cases based on the customer input.

4 LESSONS LEARNED

The high impact of customers and their situation on the shaping of how services need to be carried out shows us that a manageable outer variety of services needs to be set up based on a large inner variety of tasks to handle specific situations. At first sight, this seems contrary to what we know from variant management: managing a small inner variety of modules to offer a large outer variety of products. But variant management activities are still valuable support for service providers in setting up their portfolio.

Treating each customer inquiry individually is obviously no good way to deal with diversity. A smart move may be grouping customers who are in similar situations into clusters and using this knowledge to identify modules that in combination fulfill the overall services. This means that, in contrast to modularization of traditional products, additional sources of knowledge are required.

Let us look at how a car insurance vendor does this: customers are grouped based on a number of factors: the location, the individual driving skills and, of course, the type of car that will be insured. The location and the type of car both give insight on how an average driver behaves in the specific area and with that specific type of car. Both factors are based on statistics learned from the past. The individual driving skills, for example, are simply measured by the number of years without a crash. All three factors will influence the individual customer's price for the insurance.

What we can learn from the car insurance vendor is that abstracting the individual customer situation to as much information as necessary and as little information as possible makes the task of setting up a modularization strategy manageable. The resulting price is an average that might not be equally fair for all clients but it will be somewhat close to the truth.

4.1 What are the differences?

In this section we compare modularization and configuration of services and products and we point out the major differences.

Products can be manufactured customer-neutral or -specific, depending on the business scenario. Pick-to-order (PTO) products are typically pre-produced and put on stock. When an order comes in, the product is packaged and shipped. Assemble-to-order (ATO)

and engineer-to-order (ETO) products include creating a customer-specific compilation of constituents. Service fulfillment is always carried out customer-specific and after the order comes in.

Furthermore, we can distinguish point-of-sale (POS) and contract services. A POS service stands for its own or is sold as part of a combined product service system (PSS), in which case it is dedicated to a new product. Contracts can also be signed for servicing products that are already in use, which might even include services for products from a different manufacturer.

The manufacturing process for a product is always independent from the customer. Even though the customer can choose the characteristics of his specific product, the manufacturing is done independently of him. The machines that are used for manufacturing are the same for all customers. Service fulfillment is different in the sense that it always includes the customer himself or his products that are serviced, goods to be transported, and so on. Obviously, this means that services cannot be fulfilled prior to an order. This is different from product manufacturing and leads to a situation in which the actual price of the service fulfillment is hard to predict: specific situations might not be foreseeable. The pricing strategy will thus be an important future topic for service firms that tend to modularize their portfolio.

As already mentioned in the previous section, pricing is based on different factors. While for product manufacturing the production costs can be calculated independently from the customer, for service fulfillment this is not possible. This leads to different pricing strategies. For product manufacturing typical pricing strategies are cost-plus and list prices. Recently, also value-based pricing gained a lot of attention. Cost-plus is driven by the idea to know the production costs and to be sure there still is a margin. List prices are driven by the idea that there is a fixed price which can be shown to customer as a criterion for his product selection process. Value-based pricing is different in the sense that the main question is: how much is the customer willing to pay? Of course, all three pricing strategies are also possible for service fulfillment. But generating a price beforehand, i.e. independently from the specific customer situation requires additional information based on statistics over relevant criteria from the past.

5 SUMMARY

In this paper we introduce the problem description of modularizing and configuring services. We describe how the input was gained from case studies that were carried out based on the joint research project BakerStreet. We present first results from the case studies are draw respective conclusions that give valuable input for service firms of any application domain. Furthermore, we elaborate on the differences between modularization and configuration of traditional products and service fulfillment.

REFERENCES

[1] S. L. Vargo and R. F. Lusch, “Evolving To A New Dominant Logic for Marketing,” *J. Mark.*, 2004.

[2] F. T. Piller, “Mass Customization: Reflections on the State of the Concept,” *Int. J. Flex. Manuf. Syst.*, vol. 16, no. 4, pp. 313–334, Oct. 2004.

[3] Baldwin and Clarc, “Baldwin, Clarc (1997) Design Rules Volume 1, The Power of Modularity.pdf,” 1997.

[4] A. Lubarski and J. Pöppelbuß, “Methods for service modularization - a systematization framework,” in *Proceedings of the Pacific Asia Conference on Information Systems*, Chiayi, Taiwan, 2016.

[5] E. D. Arnheiter and H. Harren, “Quality management in a modular world,” *TQM Mag.*, vol. 18, no. 1, pp. 87–96, Jan. 2006.

[6] A. Smirnov, A. Kashevnik, N. Shilov, A. Oroszi, S. Mario, and T. Krebs, *Changing Business Information Systems for Innovative Configuration Processes*. 2015.

[7] D. Sabin and R. Weigel, “Product configuration frameworks-a survey,” *IEEE Intell. Syst.*, vol. 13, no. 4, pp. 42–49, 1998.

[8] R. Dörbecker and T. Böhmann, “The Concept and Effects of Service Modularity -- A Literature Review,” 2013, pp. 1357–1366.

[9] T. Tuunanen, A. Bask, and H. Merisalo-Rantanen, “Typology for Modular Service Design: Review of Literature,” *Int. J. Serv. Sci. Manag. Eng. Technol.*, vol. 3, no. 3, pp. 99–112, 33 2012.

[10] P. Carlborg and D. Kindström, “Service process modularization and modular strategies,” *J. Bus. Ind. Mark.*, vol. 29, no. 4, pp. 313–323, Apr. 2014.

[11] A. Bask, M. Lipponen, M. Rajahonka, and M. Tinnilä, “The concept of modularity: diffusion from manufacturing to service production,” *J. Manuf. Technol. Manag.*, vol. 21, no. 3, pp. 355–375, Mar. 2010.

[12] Y. Lin, J. Luo, and L. Zhou, “Modular logistics service platform,” in *Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on*, 2010, pp. 200–204.

[13] S. Pekkarinen and P. Ulkuniemi, “Modularity in developing business services by platform approach,” *Int. J. Logist. Manag.*, vol. 19, no. 1, pp. 84–103, May 2008.

[14] T. Bohmann and K. Loser, “Towards a service agility assessment-Modeling the composition and coupling of modular business services,” in *E-Commerce Technology Workshops, 2005. Seventh IEEE International Conference on*, 2005, pp. 140–148.

[15] A. Bask, M. Lipponen, M. Rajahonka, and M. Tinnilä, “Modularity in logistics services: a business model and process view,” *Int. J. Serv. Oper. Manag.*, vol. 10, no. 4, pp. 379–399, 2011.

[16] F. Müller and A. Lubarski, “School of thought in service modularity,” in *Proceedings of the European Conference on Information Systems (ECIS)*, Istanbul, Turkey, 2016.

[17] R. Dörbecker, O. Tokar, D. Heddaeus, and T. Böhmann, “Evaluation der Multiple Domain Matrix Methode zur Modularisierung von Dienstleistungen am Beispiel eines Versorgungsnetzwerks für psychische Erkrankungen,” *Multikonferenz Wirtsch. Annahme Zur Publ.*, 2014.

[18] Y. Lin and S. Pekkarinen, “QFD-based modular logistics service design,” *J. Bus. Ind. Mark.*, vol. 26, no. 5, pp. 344–356, Jun. 2011.

[19] C. Peters and J. M. Leimeister, “TM3-A Modularization Method for Telemedical Services: Design and Evaluation,” in *Proceedings of 21st European Conference on Information Systems (ECIS)*, 2013.

[20] C. de Blok, K. Luijkx, B. Meijboom, and J. Schols, “Modular care and service packages for independently living elderly,” *Int. J. Oper. Prod. Manag.*, vol. 30, no. 1, pp. 75–97, Jan. 2010.

[21] A. Lubarski and J. Pöppelbuß, “Modularization of Logistics Services – An Investigation of the Status Quo,” presented at the 5th International Conference on Dynamics in Logistics, Bremen, 2016.

[22] R. K. Yin, *Case study research: Design and methods*. Sage publications, 2013.

Determining New Components for Open Configuration

Linda L. ZHANG[1] **and Xiaoyu CHEN**

Abstract. Traditional product configuration is based on the assumption that all configuration elements and their relationships are predefined. Because of this assumption, most configuration-related solutions have limitations in dealing with unforeseen customer requirements. In view of these limitations, open configuration is proposed to assist companies in configuring products that can meet both unforeseen and anticipated customer requirements. This study investigates the open configuration process and proposes an approach based on cloud computing. Based on a cloud, the open configuration process includes three sub-processes: new component determination, constraint processing, and component modification and refinement. This study describes the details of the new component determination sub-process. Laptop computer open configuration is used to demonstrate how new components are determined.

1 INTRODUCTION

Product configuration has been applied in a variety of industries, such as computer, telecommunication systems, transportation, industrial products, medical systems and services [1, 2, 3]. It brings companies a number of advantages in delivering customized products, including managing product variety [4], shortening delivery time [5], improving product quality [1], simplifying order acquisition and fulfilment activities [6, 7, 8]. Despite the diversities among them, the configuration tools and methodologies are developed based on a common assumption: the configuration elements, such as components or modules, attributes, functions and their relationships are predefined [9, 10]. Because of this assumption, the products that can be configured are predefined and known in principle. In other words, the available product configuration-related solutions consider only anticipated or predefined customer requirements and leave such customer requirements that call for new components (i.e., unforeseen customer requirements) unaddressed.

In view of the above limitation of traditional product configuration, the authors in [11] propose open configuration to help companies configure products that can meet both anticipated and unforeseen customer requirements. Open configuration deals with not only the determination of new components but also the necessary modifications of predefined ones. While for meeting the unforeseen customer requirements, open configuration determines new components from a pool of external resources, it does not involve the design of new components.

Built on top of the initial study in [11], this study investigates open configuration details with a focus on its process. In view of the advantage of cloud-based computing [12, 13], we develop the open configuration process based on a cloud. The cloud-based open configuration process not only determines new components for addressing the unforeseen customer requirements but also selects predefined components for dealing with the anticipated customer requirements. It includes three sub-processes: new component determination, constraint processing, and component modification and refinement. In this paper, we shed light on the details of the new component determination sub-process.

The rest of the paper is organized as follows. In Section 2, we provide the fundamentals of open configuration, including open configuration knowledge and process. The details of determining new components are discussed in Section 3. Laptop open configuration is used as an example in Section 4 to demonstrate how new components are determined. We conclude the paper in Section 5 by providing directions for future research.

2 FUNDAMENTALS OF OPEN CONFIGURATION

2.1 Open configuration knowledge

In open configuration, there are two types of components: predefined and new. While predefined components are used to meet anticipated customer requirements, new components contribute to fulfilling unforeseen ones. To ensure the compatibility, necessary modifications and refinements are applied to predefined and new components in each open configuration. The resulting modified or refined components become predefined components for the future open configurations. Each component has a component type, several attributes and ports.

Among predefined components, there are two types of relationships. *Interior* and *exterior* constraints are defined to model these two types of relationships. While *interior* constraints define the relationships among predefined components, *exterior* constraints specify these between predefined and new components. The relationships among new components are modeled as *connects with*, *requires*, and *compatibility*. The *connects with* constraints restrict the physical connections between two components via corresponding ports, whereas the *requires* constraints specify the dependency between two components, i.e., one component requires the presence of the other component with which there is no direct physical connection. For example, a component Fingerprint Reader is physically connected with Keyboard Panel in some laptops. Thus, the relationship between Fingerprint Reader and Keyboard Panel is *connects with*. The same component Fingerprint Reader requires the presence of a CPU though there is no physical contact between them. The relationship between Fingerprint Reader and CPU is *requires*. *Compatibility* constraints determine if two components can be used in a same product configuration. For example, a bigger

[1] Corresponding author. Email: l.zhang@ieseg.fr

motherboard is not compatible with a very small liquid crystal display, thus not being able to be used in a laptop configuration.

During each open configuration process, additional constraints are produced. These constraints specify the relationships between predefined and new components. They are *modification*, *refinement*, and *connects with* constraints. While m*odification* and *refinement* constraints determine if modifications and refinements are required for selected configuration components, *connects with* constraints define the possible connections among them.

2.2 Open configuration process

In product development, cloud computing is increasingly used for providing resources and decreasing development time [14,15]. In cloud computing, a cloud is rapidly provided and released with minimal management effort or service provider interaction [16, 17]. In view of this advantage, the open configuration process in this study is proposed based on a cloud (see the functions of the cloud below). More specifically, new components in an open configuration are specified based on the cloud. The cloud enables easy access to a pool of configurable computing resources (e.g., storage, services). One of the services in the cloud is the maintenance and delivery of product databases from different providers, e.g., companies, reliable online channels. (Note, in this study, we assume that same ontology is used in organizing data in the databases.) These databases contribute to the determination of new components. They consist of all product and component data from different sources. As there are diverse databases in the cloud, to ensure the efficient utilization, in relation to a product type, each database is assigned a weight, which represents its relevance to the product to be configured. A database with a higher weight is more relevant to the product to be configured. The weights can be assigned by domain experts. The databases are indexed based on their weights and a database with a smaller index value has a higher weight.

As shown in Fig. 1, the open configuration process has three major sub-processes: new component determination, constraint processing, and component modification and refinement.

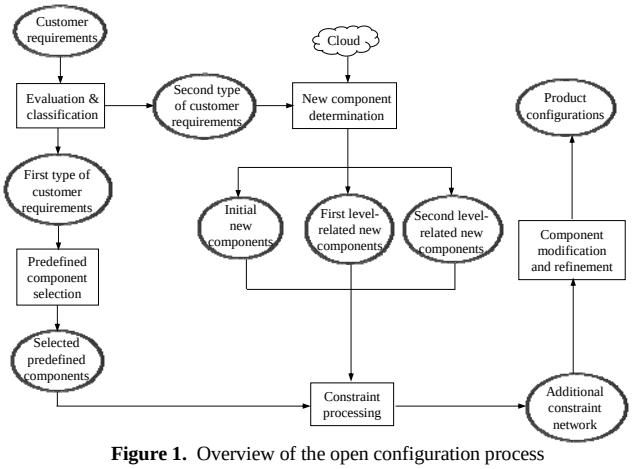


Figure 1. Overview of the open configuration process

Representing the desired product performance, customer requirements for a product to be configured are evaluated first based on the open configuration knowledge. This process rules out the unreasonable customer requirements. For example, 'a car that runs faster than an airplane' is an unreasonable requirement. Also ruled out are invalid, incomplete requirements. After the elimination, the process classifies the remaining requirements into two types. The first type is the requirements that can be met by predefined components, whereas the second type is the unforeseen customer requirements calling for the specification of new components. Corresponding to the first type, predefined components are evaluated and suitable ones are selected.

For fulfilling the second type of unforeseen customer requirements, the new component determination sub-process determines initial new components from the databases in the cloud. To specify the possible connections between the initial new components and selected predefined ones, first and second level-related new components are determined subsequently. (See details in the next section.) The first level-related new components are components that are connected to the initial new components in the same databases, whereas the second level-related are new components that are connected to the first level-related new components. In the second sub-process: constraint processing, the new components and constraints interact with these selected ones, resulting in additional constraints.

The last sub-process: component modification and refinement either modifies some of the selected predefined components or refines the newly determined ones. The modifications and refinements are based on the newly produced *modification* and *refinement* constraints. With the newly produced *connects with* constraints, the connections among all the final components are also established such that the final product configurations are determined.

3 NEW COMPONENT DETERMINATION

In response to the second type of customer requirements $CR_2 = \{cr_{21}, cr_{22}, ..., cr_{2N^{CR2}} | Type_{pd}\}$, where $Type_{pd}$ represents the type of final products to be configured, initial new components are determined from several databases, which contain the same type of final products to be configured, in the cloud, as shown in Figure 2. An initial new component is described by its type, attributes, ports, the type of final products to be configured, and the weight of the corresponding database.

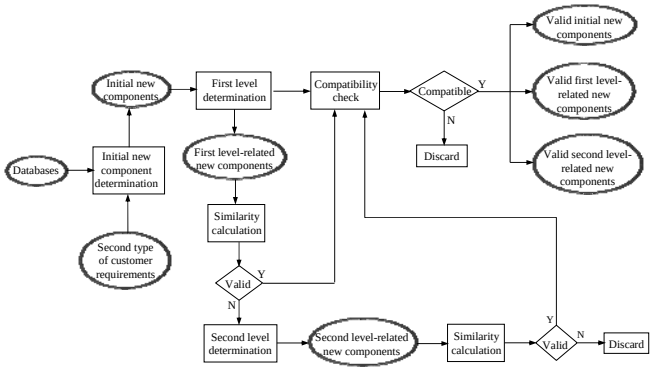


Figure 2. Determining new components

To specify the relationships between the initial new components and selected predefined components, the first level-related new components are determined. These components are connected to the initial new components with either *requires* or *connects with* constraints in the same databases. As not all of them can specify

the relationships between initial new components and predefined ones, these first level-related new components need to be validated. In validating them, similarity calculation between the first level-related and selected predefined components with the same types is carried out. The similarity value indicates how similar two components are and to what extent they can be used as an alternative replacing each other. The similarity value is zero for a first level-related component if no predefined component with the same type is found. The higher the similarity value is, the more information a first level-related component can provide for specifying the relationships between an initial new component and selected predefined ones. Attribute names and values, port types and connections, and weight values of the components are taken into account in calculating similarities.

The validity of a first level-related component is determined by comparing the corresponding similarity values with a threshold value specified by domain experts. A first level-related component is valid if at least one of its similarity values is equal to or greater than the threshold value; otherwise, it is invalid. A valid first level-related component is used as a potential reference for component modifications and refinements if it has *requires* constraints with initial new components. If a valid first level-related component has *connects with* constraints with initial new components, it is used as both a potential reference for modifications and refinements and an intermediate component for connecting an initial new component with selected predefined components.

For an invalid first level-related component, its second level-related components may be able to provide references for modifications and refinements or to connect initial new components with predefined ones. In this regard, in this study, the second level-related components of invalid first level-related ones are determined as well. These components are connected to the invalid first level-related components using either *requires* or *connects with* constraints in the databases. In the same way, similarity calculation between the second level-related components and predefined components is carried out; the validity of these second level-related components is determined based on the comparison of similarity values and given threshold values. (See the details of determining first/second level-related new components and similarity calculation in Subsections 3.2 and 3.3.) When a second level-related component is invalid, it is discarded. In summary, the valid first and second level-related components have two roles: i) establishing the connections between the initial new components and corresponding selected predefined components and ii) providing references for modifying and refining corresponding selected predefined components and valid initial new components.

At the end of the process (shown in Figure 2), compatibility checks are conducted on the newly determined components, be they valid initial, first or second level-related. The check is based on *compatibility* constraints of new components and *exterior* constraints of selected predefined components. The checks remove such initial, first, and second level-related new components that have conflicts with the above constraints. The components which respect the constraints are retained for further processing.

3.1 Determining initial new components

We first define a new component $\mathcal{N}$ as follows: $\mathcal{N} = \{T, A, P, C | w\}$, where T is the component type; A is a set of component attributes; P is a set of component ports; C is a set of constraints defining the relationships between $\mathcal{N}$ and other new components; w is the weight of $\mathcal{N}$, which is the same as that of the corresponding database.

The component attributes A are described by attribute names NA and attribute values VA. The component ports P are described by port types: TP and corresponding port connections: CP. The constraints C include (a) *connects with* constraints C^p defining the physical connections between $\mathcal{N}$ and other new components, (b) *requires* constraints C^r specifying the other new components which are required by $\mathcal{N}$, and (c) *compatibility* constraints C^c limiting the components that can be used together with $\mathcal{N}$ in a product configuration.

For any requirement of the second type : cr_{2i}, some initial new components are determined from the relevant databases. In determining the initial new components, a requirement is described in the way such that the required component type is specified and the component attributes are restrained (see Section 4 for requirement description). The new components with the required component type and attributes are determined as initial new components. An initial new component $\mathcal{N}_i$ from database *m* is denoted as { $\mathcal{N}_i^I$, w_m } with w_m representing the weight of database *m*.

3.2 Determining first and second level-related new components

In determining the first/second level-related new components, the *requires* and *connects with* constraints of the initial new components are utilized. More specifically, the components, which have either *requires* or *connects with* relationship with initial new components are determined as first level-related components; these having either *requires* or *connects with* relationship with a first level-related component are determined as second level-related new components. In the condition that a first level-related new component has only *requires* (or *connects with*) constraint with initial new components, the corresponding second level-related new components are determined based on *requires* (or *connects with*) constraints only.

For examples, as shown in Figure 3, based on a *connects with* constraint: $C^p_{1,1}$, $\mathcal{N}^{FL}_{11}$ is determined as a first level-related new component corresponding to an initial new component $\mathcal{N}^I_1$; with a *requires* constraint: $C^r_{1,1}$, $\mathcal{N}^{FL}_{12}$ is determined as another first level-related new component of $\mathcal{N}^I_1$; based on a *connects with* constraint: $C^p_{11,1}$, $\mathcal{N}^{SL}_{111}$ is determined as a second level-related new component corresponding to the first level-related new component: $\mathcal{N}^{FL}_{11}$; based on a *requires* constraint: $C^r_{12,1}$, $\mathcal{N}^{SL}_{121}$ is determined as a second level-related new component of the first level-related new component $\mathcal{N}^{FL}_{12}$.

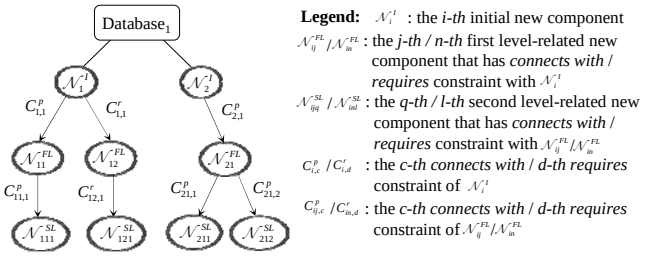


Figure 3. Determining first and second level-related new components

Determining first level-related new component using *connects with* constraint:
$\left(\exists \mathcal{N}_k \middle| \mathcal{N}_i^I.C_{i,c}^p = \mathcal{N}_k\right) \Rightarrow \left(\{\mathcal{N}_{ij}^{FL}, w_m\} := \{\mathcal{N}_k, w_m\}\right)$, where $\mathcal{N}_i^I$ is the *i-th* initial new component from the *m-th* database; $C_{i,c}^p$ is the *c-th connects with* constraint of $\mathcal{N}_i^I$ indicating that $\mathcal{N}_i^I$ is connected with $\mathcal{N}_k$; $\mathcal{N}_k$ is the *k-th* new component from the same database; $\mathcal{N}_{ij}^{FL}$ is the *j-th* first level-related new component of $\mathcal{N}_i^I$; w_m is the weight of the *m-th* database.

Determining first level-related new component using *requires* constraint:
$\left(\exists \mathcal{N}_k \middle| \mathcal{N}_i^I.C_{i,d}^r = \mathcal{N}_k\right) \Rightarrow \left(\{\mathcal{N}_{in}^{FL}, w_m\} := \{\mathcal{N}_k, w_m\}\right)$, where $\mathcal{N}_i^I$, $\mathcal{N}_k$, and w_m have the same meaning as above; $C_{i,d}^r$ is the *d-th requires* constraint of $\mathcal{N}_i^I$ indicating that $\mathcal{N}_i^I$ requires $\mathcal{N}_k$; $\mathcal{N}_{in}^{FL}$ is the *n-th* first level-related component of $\mathcal{N}_i^I$.

Determining second level-related new component using *connects with* constraint:
$\left(\exists \mathcal{N}_k \middle| \mathcal{N}_{ij}^{FL}.C_{ij,c}^p = \mathcal{N}_k\right) \Rightarrow \left(\{\mathcal{N}_{ijq}^{SL}, w_m\} := \{\mathcal{N}_k, w_m\}\right)$, where $\mathcal{N}_{ij}^{FL}$ is the *j-th* first level-related new component of $\mathcal{N}_i^I$ from the *m-th* database; $C_{ij,c}^p$ is the *c-th connects with* constraint of $\mathcal{N}_{ij}^{FL}$ indicating that $\mathcal{N}_{ij}^{FL}$ is connected to $\mathcal{N}_k$; $\mathcal{N}_{ijq}^{SL}$ is the *q-th* second level-related component corresponding to $\mathcal{N}_{ij}^{FL}$; $\mathcal{N}_k$ and w_m have the same meaning as above.

Determining second level-related new component using *requires* constraint:
$\left(\exists \mathcal{N}_k \middle| \mathcal{N}_{in}^{FL}.C_{in,d}^r = \mathcal{N}_k\right) \Rightarrow \left(\{\mathcal{N}_{inl}^{SL}, w_m\} := \{\mathcal{N}_k, w_m\}\right)$, where $\mathcal{N}_{in}^{FL}$, $\mathcal{N}_k$, and w_m have the same meaning as above; $C_{in,d}^r$ is the *d-th requires* constraint of $\mathcal{N}_{in}^{FL}$ indicating that $\mathcal{N}_{in}^{FL}$ requires $\mathcal{N}_k$; $\mathcal{N}_{inl}^{SL}$ is the *l-th* second level-related component corresponding to $\mathcal{N}_{in}^{FL}$.

3.3 Calculating similarities

In calculating the similarity between a first (or second) level-related new components and a selected predefined one, the similarity values of their attribute names NA, attribute values VA, port type TP, and port connections CP are calculated first. The similarity of the two components is the weighted sum of the similarity values of the above four elements multiplied by the corresponding weight of the first (or second) level-related new component. For a first level-related component $\mathcal{N}_{ij}^{FL}$ and a predefined component $\mathcal{P}_s$, their similarity values $S(\mathcal{N}_{ij}^{FL}, \mathcal{P}_s)$ is calculated based on Eq. (1).

$$S(\mathcal{N}_{ij}^{FL}, \mathcal{P}_s) = \begin{cases} w_m * \begin{pmatrix} \alpha * S(NA_{ij}, NA_s) + \beta * S(VA_{ij}, VA_s) + \\ \lambda * S(TP_{ij}, TP_s) + \eta * S(CP_{ij}, CP_s) \end{pmatrix}, & T_{ij} = T_s \\ 0, & otherwise \end{cases} . \quad (1)$$

In the above equation, w_m is the weight of $\mathcal{N}_{ij}^{FL}$; $S(NA_{ij}, NA_s)$, $S(VA_{ij}, VA_s)$, $S(TP_{ij}, TP_s)$, and $S(CP_{ij}, CP_s)$ are the similarity values of NA, VA, TP, and CP, respectively; α, β, λ, and η with $\alpha+\beta+\lambda+\eta=1$ are the weights of NA, VA, TP, and CP. Based on their own situations, companies can adopt different weight values. For illustrative simplicity, we assign 0.25 to each weight in this study. In calculating the similarity values of the four elements, a Boolean operator ε is introduced.

Calculating $S(NA_{ij}, NA_s)$: The similarity value of attribute names of $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$: $S(NA_{ij}, NA_s)$ is calculated based on the below equation.

$$S(NA_{ij}, NA_s) = \frac{2\sum_{a,b} \varepsilon(NA_{ij,a}, NA_{s,b})}{N^{NA_{ij}} + N^{NA_s}}, \quad (2)$$

where $N^{NA_{ij}}$ and N^{NA_s} represent the total number of attributes describing $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$ with a = 1, …, $N^{NA_{ij}}$ and b = 1, …, N^{NA_s};

$\varepsilon(NA_{ij,a}, NA_{s,b}) = \begin{cases} 0, & \neg Sim(NA_{ij,a}, NA_{s,b}) \\ 1, & Sim(NA_{ij,a}, NA_{s,b}) \end{cases}$ indicating the similarity status of the *a-th* attribute of $\mathcal{N}_{ij}^{FL}$: $NA_{ij,a}$ and the *b-th* attribute of $\mathcal{P}_s$: $NA_{s,b}$. $\varepsilon(NA_{ij,a}, NA_{s,b}) = 0$ if $NA_{ij,a}$ and $NA_{s,b}$ are different (i.e., $\neg Sim(NA_{ij,a}, NA_{s,b})$); otherwise (i.e., $Sim(NA_{ij,a}, NA_{s,b})$ is true), $\varepsilon(NA_{ij,a}, NA_{s,b}) = 1$.

Calculating $S(VA_{ij}, VA_s)$: The similarity value of attribute values of $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$: $S(VA_{ij}, VA_s)$ is calculated based on Eq. (3) below.

$$S(VA_{ij}, VA_s) = \begin{cases} 0, & \sum_{a,b} \varepsilon(NA_{ij,a}, NA_{s,b}) = 0 \\ \dfrac{\sum_{a,b} S(VA_{ij,a}, VA_{s,b})}{\sum_{a,b} \varepsilon(NA_{ij,a}, NA_{s,b})}, & otherwise \end{cases}, \quad (3)$$

where $S(VA_{ij,a}, VA_{s,b})$ is the similarity value of attribute values of $NA_{ij,a}$ and $NA_{s,b}$; $\varepsilon(NA_{ij,a}, NA_{s,b})$ denotes the similarity status of $NA_{ij,a}$ and $NA_{s,b}$ (see the definition in Eq. (2)). $S(VA_{ij,a}, VA_{s,b})$ is computed based on Eq. (4) below.

$$S(VA_{ij,a}, VA_{s,b}) = \begin{cases} 0, & \varepsilon(NA_{ij,a}, NA_{s,b}) = 0 \\ \varepsilon(VA_{ij,a}, VA_{s,b}), & \varepsilon(NA_{ij,a}, NA_{s,b}) = 1 \end{cases}. \quad (4)$$

In the above equation, $\varepsilon(VA_{ij,a}, VA_{s,b})$ indicating the similarity status of the *a-th* attribute value of $\mathcal{N}_{ij}^{FL}$: $VA_{ij,a}$ and the *b-th* attribute value of $\mathcal{P}_s$: $VA_{s,b}$. It has two values: 0 and 1, i.e.,

$\varepsilon(VA_{ij,a}, VA_{s,b}) = \begin{cases} 0, & \neg Sim(VA_{ij,a}, VA_{s,b}) \\ 1, & Sim(VA_{ij,a}, VA_{s,b}) \end{cases}$. $\varepsilon(VA_{ij,a}, VA_{s,b}) = 0$ if $VA_{ij,a}$ and $VA_{s,b}$ are different (i.e., $\neg Sim(VA_{ij,a}, VA_{s,b})$); otherwise (i.e., $Sim(VA_{ij,a}, VA_{s,b})$ is true), $\varepsilon(VA_{ij,a}, VA_{s,b}) = 1$.

Calculating $S(TP_{ij}, TP_s)$: The similarity value of port types of $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$: $S(TP_{ij}, TP_s)$ is calculated using Eq. (5).

$$S(TP_{ij}, TP_s) = \frac{2\sum_{e,f} \varepsilon(TP_{ij,e}, TP_{s,f})}{N^{TP_{ij}} + N^{TP_s}}, \quad (5)$$

where $N^{TP_{ij}}$ and N^{TP_s} represent the total number of ports of $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$, respectively with e = 1, …, $N^{TP_{ij}}$ and f = 1, …, N^{TP_s};

$\varepsilon(TP_{ij,e}, TP_{s,f}) = \begin{cases} 0, & \neg Sim(TP_{ij,e}, TP_{s,f}) \\ 1, & Sim(TP_{ij,e}, TP_{s,f}) \end{cases}$ indicating the similarity status of the *e-th* port type of $\mathcal{N}_{ij}^{FL}$: $TP_{ij,e}$ and the *f-th* port type of $\mathcal{P}_s$: $TP_{s,f}$. $\varepsilon(TP_{ij,e}, TP_{s,f}) = 0$ if $TP_{ij,e}$ and $TP_{s,f}$ are different (i.e., $\neg Sim(TP_{ij,e}, TP_{s,f})$); otherwise (i.e., $Sim(TP_{ij,e}, TP_{s,f})$ is true), $\varepsilon(TP_{ij,e}, TP_{s,f}) = 1$.

Calculating $S(CP_{ij}, CP_s)$: The similarity value of port connections of $\mathcal{N}_{ij}^{FL}$ and $\mathcal{P}_s$: $S(CP_{ij}, CP_s)$ is calculated based on Eq. (6) below.

$$S(CP_{ij}, CP_s) = \begin{cases} 0, & \sum_{e,f} \varepsilon(TP_{ij,e}, TP_{s,f}) = 0 \\ \dfrac{\sum_{e,f} S(CP_{ij,e}, CP_{s,f})}{\sum_{e,f} \varepsilon(TP_{ij,e}, TP_{s,f})}, & otherwise \end{cases}, \quad (6)$$

where $S(CP_{ij,e}, CP_{s,f})$ is the similarity value of port connections of port types $TP_{ij,e}$ and $TP_{s,f}$ and is computed based on Eq. (7); $\varepsilon(TP_{ij,e}, TP_{s,f})$ denotes the similarity status of $TP_{ij,e}$ and $TP_{s,f}$ (see the definition in Eq. (5)).

$$S(CP_{ij,e}, CP_{s,f}) = \begin{cases} 0, & \varepsilon(TP_{ij,e}, TP_{s,f}) = 0 \\ \dfrac{2\sum_{g,h} \varepsilon(CP_{ij,eg}, CP_{s,fh})}{N^{CP_{ij,e}} + N^{CP_{s,f}}}, & \varepsilon(TP_{ij,e}, TP_{s,f}) = 1 \end{cases}, \quad (7)$$

where $N^{CP_{ij,e}}$ and $N^{CP_{s,f}}$ represent the total number of port connections of $TP_{ij,e}$ and $TP_{s,f}$ with g = 1, …, $N^{CP_{ij,e}}$ and h = 1, …, $N^{CP_{s,f}}$; $\varepsilon(CP_{ij,eg}, CP_{s,fh}) = \begin{cases} 0, & \neg Sim(CP_{ij,eg}, CP_{s,fh}) \\ 1, & Sim(CP_{ij,eg}, CP_{s,fh}) \end{cases}$ indicating the similarity status of the *g-th* port connection of $TP_{ij,e}$: $CP_{ij,eg}$ and the *h-th* port connection of $TP_{s,f}$: $CP_{s,fh}$. $\varepsilon(CP_{ij,eg}, CP_{s,fh}) = 0$ if $CP_{ij,eg}$ and $CP_{s,fh}$ are different (i.e., $\neg Sim(CP_{ij,eg}, CP_{s,fh})$); otherwise (i.e., $Sim(CP_{ij,eg}, CP_{s,fh})$ is true), $\varepsilon(CP_{ij,eg}, CP_{s,fh}) = 1$.

In determining the validity of a first level-related new component, a threshold value: V_{Th}^{Sim} is used. This value can be specified by the domain experts. Similarly, the similarity values between the second level-related new components and predefined ones are calculated and the validity of the second level-related new components is determined.

3.4 Checking compatibilities

Compatibility checks are conducted for all valid initial, first and second level-related new components. The checks are conducted based on the *compatibility* constraints which are defined in the same databases. If a first (or second) level-related new component is incompatible with an initial new component, it is discarded for further consideration. If two first (or second) level-related new components or if a first level-related and second level-related new component are incompatible, the one with a lower similarity value with predefined components is discarded. For example, $\mathcal{N}_{11}^{FL}$ is incompatible with $\mathcal{N}_{21}^{FL}$ based on one of its *compatibility* constraints: $\mathcal{N}_{11}^{FL}.C_{11,1}^{c} = \neg\mathcal{N}_{21}^{FL}$. Further, the highest similarity value of $\mathcal{N}_{21}^{FL}$ is greater than that of $\mathcal{N}_{11}^{FL}$. $\mathcal{N}_{11}^{FL}$ is discarded.

Compatibility check is also conducted based on the *exterior* constraints of selected predefined and dynamic components, which define the possible external connections. For example, an *exterior* constraint of a selected predefined component is $\mathcal{P}_1.C_1^E = \neg ExteriorConnection$. It indicates that no exterior connection is allowed for $\mathcal{P}_1$. The initial new components that have connection relationships with $\mathcal{P}_1$ are, thus, discarded.

4 CASE STUDY

An example of laptop computer open configuration is introduced to demonstrate how new components are determined. There are 11 types of predefined components in the laptop open configuration, including *CPU*, *Motherboard*, *Memory*, *HardDisk*, *Monitor*, *Keyboard*, *Battery*, *ExtensionSocket*, *Upper Cover*, *Keyboard Panel* and *Bottom Casing*, as shown in Figure 4, and the product type $Type_{pd}$ = '*Laptop Computer*'.

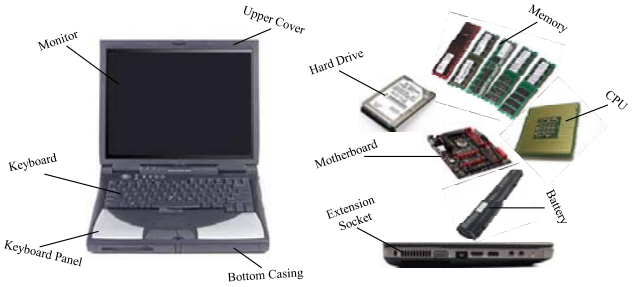


Figure 4. Predefined components of a laptop computer

Table 1 provides the details of some predefined components, including component types, attributes and attribute values, and ports and port connections. For illustrative simplicity, the table is truncated and provides these predefined components that are used in the rest of the example. (Note: the ID of each component is provided in the small brackets after each symbol. This is the same for the rest of the tables and text.)

Table 1. Some predefined components

Component	Component Type	Attribute Name and Value	Port Type and Connections
$\mathcal{P}_1$ (CPU #1)	CPU	(Processor, Speed): (Intel Xeon E5-2690, 2.6GHz)	P_{MB}: {MBIntel Xeon}
$\mathcal{P}_2$ (CPU #2)		(Processor, Speed): (Intel Xeon E5-2680, 2.7GHz)	
$\mathcal{P}_3$ (CPU #3)		(Processor, Speed): (Intel Core i7-3970X, 3.3GHz)	P_{MB}: {MBIntel Core}
$\mathcal{P}_4$ (CPU #4)		(Processor, Speed): (Intel Core i7-3960X, 3.5GHz)	
$\mathcal{P}_5$ (MB #1)	Motherboard	(Chip, SptCPU, MaxSptCPU, MaxSptRAM): (Intel B85, CPU Xeon, 2, 2)	P_{CPU}: {CPU Xeon} P_{Memory}: {SDRAM, RDRAM} P_{HD}: {ATA-100}
$\mathcal{P}_6$ (MB #2)		(Chip, SptCPU, MaxSptCPU, MaxSptRAM): (Intel B150, CPU Core, 2, 2)	P_{CPU}: {CPU Core} P_{Memory}: {SDRAM, RDRAM} P_{HD}: {ATA-100}
$\mathcal{P}_7$ (MB #3)		(Chip, SptCPU, MaxSptCPU, MaxSptRAM): (Intel Z97, CPU Core, 2, 2)	
$\mathcal{P}_8$ (UC #1)	Upper Cover	(Size, Material, Color, KPCut, MonCut): (12in., Hybrid, Black, KP-12, LCD-12)	$P_{Monitor}$: {LCD-12} P_{Casing}: {Keyboard Panel} $P_{ESocket}$: {Monitor Socket}
$\mathcal{P}_9$ (UC #2)		(Size, Material, Color, KPCut, MonCut): (14in., Hybrid, Black, KP-14, LCD-14)	$P_{Monitor}$: {LCD-14} P_{Casing}: {Keyboard Panel} $P_{ESocket}$: {Monitor Socket}
$\mathcal{P}_{10}$ (KP #1)	Keyboard Panel	(Size, Material, Color, UCCut, KBCut, PSCut, MSCut, USBCut): (12in., Hybrid, Black, UC-12, KB-12, PS-std, MS-std, USB-std)	$P_{Caseing}$: {Upper Cover} $P_{Keyboard}$: {KB-12} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket}
$\mathcal{P}_{11}$ (KP #2)		(Size, Material, Color, UCCut, KBCut, PSCut, MSCut, USBCut): (14in., Hybrid, Black, UC-14, KB-14, PS-std, MS-std,	$P_{Caseing}$: {Upper Cover} $P_{Keyboard}$: {KB-14} $P_{ESocket}$: {Power Socket, Modem Socket, USB

		USB-std)	Socket}
$\mathcal{P}_{12}$ (BC #1)	Bottom Casing	(Size, Material, Color, KPCut, BTCut, PSCut, MSCut, USBCut): (12in., Hybrid, Black, KP-12, BT-std, PS-std, MS-std, USB-std)	P_{Casing}: {Keyboard Panel} $P_{Battery}$: {Battery-NC, Battery-Lith, Battery-Hyb} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket}
$\mathcal{P}_{13}$ (BC #2)		(Size, Material, Color, KPCut, BTCut, PSCut, MSCut, USBCut): (14in., Hybrid, Black, KP-14, BT-std, PS-std, MS-std, USB-std)	
…	…	…	…

Suppose the requirements from a customer are as follows: CR = {*CPU*.Speed >= 3.3GHz, *Motherboard*.(Chip, SptCPU) = (Intel B 150, CPU Core), *Upper Cover*.Material = Hybrid, *SIM Socket*.Number = 1, *Fingerprint Reader*.Number = 1}. These customer requirements can be classified into two types: the first type of customer requirements CR_1 and the second type of customer requirements CR_2. While CR_1 = {*CPU*.Speed >= 3.3GHz, *Motherboard*.(Chip, SptCPU) = (Intel B 150, CPU Core), *Upper Cover*.Material = Hybrid}, CR_2 = {*SIM Socket*.Number = 1, *Fingerprint Reader*.Number = 1 $|Type^{pd}$ = '*Laptop Computer*'}. The predefined components: $\mathcal{P}_3$ (*CPU #3*), $\mathcal{P}_4$ (*CPU #4*), $\mathcal{P}_6$ (*MB #2*), $\mathcal{P}_8$ (*UC #1*), $\mathcal{P}_9$ (*UC #2*), $\mathcal{P}_{10}$ (*KP #1*), $\mathcal{P}_{11}$ (*KP #2*), $\mathcal{P}_{12}$ (*BC #1*), $\mathcal{P}_{13}$ (*BC #2*) are selected to fulfill CR_1.

In accordance with the product type: $Type^{pd}$ = '*Laptop Computer*', two databases: database$_1$ with a weight w_1 and database$_2$ with a weight w_2 are selected from the cloud. The weight values of the two database are w_1 = 0.9 and w_2 = 0.85. To meet CR_2, some initial new components are determined from these databases. Their details are provided in Table 2.

Table 2. Initial new components determined from the two databases

Component	Component Type	Attribute Name and Value	Port Type and Connections
Database$_1$			
$\mathcal{N}_1$ (ES #1)	SIM Socket	MaxCardinality: 2	P_{Casing}: {Keyboard Panel, Bottom Casing}
$\mathcal{N}_2$ (SG #1)	Fingerprint Reader	(Chip, Size): (v1.0, 1cm)	P_{Casing}: {Keyboard Panel}
$\mathcal{N}_3$ (SG #2)		(Chip, Size): (v2.0, 1.5cm)	P_{Casing}: {Keyboard Panel}
Database$_2$			
$\mathcal{N}_1$ (SG #1)	Fingerprint Reader	(Chip, Size): (v3.0, 1.3cm)	P_{Casing}: {Upper Cover, Keyboard Panel}

These four initial new components together with their weights are denoted as follows: $\{\mathcal{N}_1^I, w_1\}$, $\{\mathcal{N}_2^I, w_1\}$, $\{\mathcal{N}_3^I, w_1\}$ and $\{\mathcal{N}_1^I, w_2\}$.They have 13 *connects with* and *requires* constraints as follows :

1) $\{\mathcal{N}_1^I, w_1\}.C_{1,1}^p = \mathcal{N}_4(KP\ \#1)$;
2) $\{\mathcal{N}_1^I, w_1\}.C_{1,2}^p = \mathcal{N}_5(KP\ \#2)$;
3) $\{\mathcal{N}_1^I, w_1\}.C_{1,3}^p = \mathcal{N}_6(BC\ \#1)$;
4) $\{\mathcal{N}_1^I, w_1\}.C_{1,4}^p = \mathcal{N}_7(BC\ \#2)$;
5) $\{\mathcal{N}_2^I, w_1\}.C_{2,1}^p = \mathcal{N}_4(KP\ \#1)$;
6) $\{\mathcal{N}_3^I, w_1\}.C_{3,1}^p = \mathcal{N}_5(KP\ \#2)$;
7) $\{\mathcal{N}_1^I, w_2\}.C_{1,1}^p = \mathcal{N}_3(UC\ \#1)$;
8) $\{\mathcal{N}_1^I, w_2\}.C_{1,2}^p = \mathcal{N}_4(KP\ \#1)$;
9) $\{\mathcal{N}_2^I, w_1\}.C_{2,1}^r = \mathcal{N}_8(CPU\ \#1)$;
10) $\{\mathcal{N}_2^I, w_1\}.C_{2,2}^r = \mathcal{N}_9(CPU\ \#2)$;
11) $\{\mathcal{N}_3^I, w_1\}.C_{3,1}^r = \mathcal{N}_8(CPU\ \#1)$;
12) $\{\mathcal{N}_3^I, w_1\}.C_{3,2}^r = \mathcal{N}_9(CPU\ \#2)$;
13) $\{\mathcal{N}_1^I, w_2\}.C_{1,1}^r = \mathcal{N}_6(CPU\ \#1)$.

Take the first one: $\{\mathcal{N}_1^I, w_1\}.C_{1,1}^p = \mathcal{N}_4(KP\ \#1)$ as an example. In the *connects with* constraint $C_{1,1}^p$ of $\mathcal{N}_1^I$ from database1, $\mathcal{N}_1^I$ is connected with a specific keyboard panel: $\mathcal{N}_4(KP\ \#1)$ from the same database. The remaining constraints can be interpreted in a similar way.

Based on the determination of first level-related new components introduced earlier, $\mathcal{N}_4(KP\ \#1)$ and $\mathcal{N}_5(KP\ \#2)$ are determined as first level-related new components of $\mathcal{N}_1^I$. In a same way, all the first level-related new components for the above initial new components are determined and provided in Table 3.

Table 3. The first level-related new components

Component	Component Type	Attribute Name and Value	Port Type and Connections
Database$_1$			
$\mathcal{N}_4$ (KP #1)	Keyboard Panel	(Size, Material, Color, UCCut, KBCut, PSCut, MSCut, USBCut, SIMCut, FPCut): (12in., Hybrid, Black, UC-12, KB-12, PS-std, MS-std, USB-std, SIM-std, FP-v1.0)	$P_{Caseing}$: {Upper Cover} $P_{Keyboard}$: {KB-12} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket, SIM Socket} P_{SG}: {Fingerprint Reader}
$\mathcal{N}_5$ (KP #2)		(Size, Material, Color, UCCut, KBCut, PSCut, MSCut, USBCut, SIMCut, FPCut): (14in., Hybrid, Black, UC-14, KB-14, PS-std, MS-std, USB-std, SIM-std, FP-v2.0)	$P_{Caseing}$: {Upper Cover} $P_{Keyboard}$: {KB-14} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket, SIM Socket} P_{SG}: {Fingerprint Reader}
$\mathcal{N}_6$ (BC #1)	Bottom Casing	(Size, Material, Color, KPCut, BTCut, PSCut, MSCut, USBCut, SIMCut): (12in., Hybrid, Black, KP-12, BT-std, PS-std, MS-std, USB-std, SIM-std)	P_{Casing}: {Keyboard Panel} $P_{Battery}$: {Battery-NC, Battery-Lith, Battery-Hyb} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket, SIM Socket}
$\mathcal{N}_7$ (BC #2)		(Size, Material, Color, KPCut, BTCut, PSCut, MSCut, USBCut, SIMCut): (14in., Hybrid, Black, KP-14, BT-std, PS-std, MS-std, USB-std, SIM-std)	
$\mathcal{N}_8$ (CPU #1)	CPU	(Processor, Speed): (Intel Core i7-3970X, 3.3GHz)	P_{MB}: {MBIntel Core}
$\mathcal{N}_9$ (CPU #2)		(Processor, Speed): (AMD A10-7850K, 3.7GHz)	P_{MB}: {MBAMD}
Database$_2$			
$\mathcal{N}_3$ (UC #1)	Upper Cover	(Size, Material, Color, KPCut, MonCut, FPCut): (14in., Hybrid, Black, KP-14, LCD-14, FP-v3.0)	$P_{Monitor}$: {LCD-14} P_{Casing}: {Keyboard Panel} $P_{ESocket}$: {Monitor Socket} P_{SG}: {Fingerprint Reader}
$\mathcal{N}_4$ (KP #1)	Keyboard Panel	(Size, Material, Color, UCCut, KBCut, PSCut, MSCut, USBCut, SIMCut, FPCut): (14in., Hybrid, Black, UC-14, KB-14, PS-std, MS-std, USB-std, SIM-std, FP-v3.0)	$P_{Caseing}$: {Upper Cover} $P_{Keyboard}$: {KB-14} $P_{ESocket}$: {Power Socket, Modem Socket, USB Socket, SIM Socket} P_{SG}: {Fingerprint Reader}
$\mathcal{N}_6$ (CPU #1)	CPU	(Processor, Speed): (Intel Core i7-3950X, 3.4GHz)	P_{MB}: {MBIntel Core}

These first level-related new components are denoted as follows: $\{\mathcal{N}_{11}^{FL}, w_1\} = \{\mathcal{N}_4(KP\ \#1), w_1\}$; $\{\mathcal{N}_{12}^{FL}, w_1\} = \{\mathcal{N}_5(KP\ \#2), w_1\}$; $\{\mathcal{N}_{13}^{FL}, w_1\} = \{\mathcal{N}_6(BC\ \#1), w_1\}$; $\{\mathcal{N}_{14}^{FL}, w_1\} = \{\mathcal{N}_7(BC\ \#2), w_1\}$; $\{\mathcal{N}_{21}^{FL}, w_1\} = \{\mathcal{N}_4(KP\ \#1), w_1\}$; $\{\mathcal{N}_{22}^{FL}, w_1\} = \{\mathcal{N}_8(CPU\ \#1), w_1\}$; $\{\mathcal{N}_{23}^{FL}, w_1\} = \{\mathcal{N}_9(CPU\ \#2), w_1\}$; $\{\mathcal{N}_{31}^{FL}, w_1\} = \{\mathcal{N}_5(KP\ \#2), w_1\}$; $\{\mathcal{N}_{32}^{FL}, w_1\} = \{\mathcal{N}_8(CPU\ \#1), w_1\}$; $\{\mathcal{N}_{33}^{FL}, w_1\} = \{\mathcal{N}_9(CPU\ \#2), w_1\}$; $\{\mathcal{N}_{11}^{FL}, w_2\} = \{\mathcal{N}_3(UC\ \#1), w_2\}$; $\{\mathcal{N}_{12}^{FL}, w_2\} = \{\mathcal{N}_4(KP\ \#1), w_2\}$; $\{\mathcal{N}_{13}^{FL}, w_2\} = \{\mathcal{N}_6(CPU\ \#1), w_2\}$. To evaluate their validity, the similarity values of

these first level-related new components and the selected predefined ones are calculated. The similarity values of $\{\mathcal{N}_{23}^{FL}, w_1\}$ and $\{\mathcal{N}_{33}^{FL}, w_1\}$ are zero since there are no predefined components with the same component type. The highest similarity value of each of the rest components is provided below.

$S(\{\mathcal{N}_{11}^{FL}, w_1\}, \mathcal{P}_{10}) = S(\{\mathcal{N}_{21}^{FL}, w_1\}, \mathcal{P}_{10}) = 0.81$;
$S(\{\mathcal{N}_{12}^{FL}, w_1\}, \mathcal{P}_{11}) = S(\{\mathcal{N}_{31}^{FL}, w_1\}, \mathcal{P}_{11}) = 0.81$;
$S(\{\mathcal{N}_{13}^{FL}, w_1\}, \mathcal{P}_{12}) = 0.86$;
$S(\{\mathcal{N}_{14}^{FL}, w_1\}, \mathcal{P}_{13}) = 0.86$;
$S(\{\mathcal{N}_{22}^{FL}, w_1\}, \mathcal{P}_{3}) = S(\{\mathcal{N}_{32}^{FL}, w_1\}, \mathcal{P}_{3}) = 0.90$;
$S(\{\mathcal{N}_{11}^{FL}, w_2\}, \mathcal{P}_{9}) = 0.78$;
$S(\{\mathcal{N}_{12}^{FL}, w_2\}, \mathcal{P}_{11}) = 0.77$;
$S(\{\mathcal{N}_{13}^{FL}, w_2\}, \mathcal{P}_{3}) = 0.64$.

The threshold value: V_{Th}^{Sim} is given as 0.7. As the highest similarity value of $\{\mathcal{N}_{31}^{FL}, w_2\}$ is less than 0.7, $\{\mathcal{N}_{31}^{FL}, w_2\}$ is invalid for specifying the relationships between the initial new components and selected predefined ones. Similarly, $\{\mathcal{N}_{23}^{FL}, w_1\}$ and $\{\mathcal{N}_{33}^{FL}, w_1\}$, which have 0 as similarity value, are invalid as well. As the rest of components have similarity values larger than 0.7, they are valid and can be used as an alternative to replace the corresponding selected predefined components.

In a similar fashion, the second level-related new components are determined for the above invalid first level-related new components. As $\{\mathcal{N}_{23}^{FL}, w_1\}$ and $\{\mathcal{N}_{33}^{FL}, w_1\}$ are the same component, the second level-related components are the same. In this regard, $\{\mathcal{N}_{23}^{FL}, w_1\}$ is used to determine the second level-related new components. With the *requires* constraint of $\{\mathcal{N}_{23}^{FL}, w_1\}$: $\{\mathcal{N}_{23}^{FL}, w_1\}.C_{23,1}^{r} = \mathcal{N}_{10}(MB\ \#2)$ in database$_1$ and the *requires* constraint of $\{\mathcal{N}_{13}^{FL}, w_2\}$: $\{\mathcal{N}_{13}^{FL}, w_2\}.C_{13,1}^{r} = \mathcal{N}_{5}(MB\ \#1)$ in database$_2$, their second level-related components are determined and denoted as: $\{\mathcal{N}_{231}^{SL}, w_1\}$ = $\{\mathcal{N}_{10}(MB\ \#2), w_1\}$ and $\{\mathcal{N}_{131}^{SL}, w_2\}$ = $\{\mathcal{N}_{5}(MB\ \#1), w_2\}$ (see component details in Table 4).

Similarly, similarity values of these second level-related new components are calculated for evaluating their validity. As there is no predefined component with the same type, $\{\mathcal{N}_{131}^{SL}, w_2\}$ has a 0 similarity value, thus being invalid. $\{\mathcal{N}_{131}^{SL}, w_2\}$ has a similarity value of 0.85, thus being retained.

Table 4. The second level-related new components

Component	Component Type	Attribute Name and Value	Port Type and Connections
Database $_1$			
$\mathcal{N}_{10}$ (MB #2)	Motherboard	(Chip, SptCPU, MaxSptCPU, MaxSptRAM):(Intel B85, CPU Core, 2, 2); (AMD B85, CPU AMD, 2, 2)	P_{CPU}: {CPU AMD} P_{Memory}: {SDRAM} P_{HD}: {ATA-100}
Database $_2$			
$\mathcal{N}_{5}$ (MB #1)	Motherboard	(Chip, SptCPU, MaxSptCPU, MaxSptRAM): (Intel B150, CPU Core, 2, 2)	P_{CPU}: {CPU Core} P_{Memory}: {SDRAM, RDRAM} P_{HD}: {ATA-100}

In summary, the final initial new components are $\{\mathcal{N}_1^I, w_1\}$, $\{\mathcal{N}_2^I, w_1\}$, $\{\mathcal{N}_3^I, w_1\}$, and $\{\mathcal{N}_1^I, w_2\}$. The valid first and second level-related new components include $\{\mathcal{N}_{11}^{FL}, w_1\}$, $\{\mathcal{N}_{12}^{FL}, w_1\}$, $\{\mathcal{N}_{13}^{FL}, w_1\}$, $\{\mathcal{N}_{14}^{FL}, w_1\}$, $\{\mathcal{N}_{21}^{FL}, w_1\}$, $\{\mathcal{N}_{22}^{FL}, w_1\}$, $\{\mathcal{N}_{31}^{FL}, w_1\}$, $\{\mathcal{N}_{32}^{FL}, w_1\}$, $\{\mathcal{N}_{11}^{FL}, w_2\}$ and $\{\mathcal{N}_{131}^{SL}, w_2\}$. As they have *requires* constraints with $\{\mathcal{N}_2^I, w_1\}$, $\{\mathcal{N}_3^I, w_1\}$, and $\{\mathcal{N}_{13}^{FL}, w_1\}$, $\{\mathcal{N}_{22}^{FL}, w_1\}$, $\{\mathcal{N}_{32}^{FL}, w_1\}$, and $\{\mathcal{N}_{131}^{SL}, w_2\}$ are used as references for modifying and refining $\mathcal{P}_3$ (*CPU #3*), $\mathcal{P}_6$ (*MB #2*), $\{\mathcal{N}_2^I, w_1\}$, and $\{\mathcal{N}_3^I, w_1\}$. As they have *connects with* constraints with $\{\mathcal{N}_1^I, w_1\}$, $\{\mathcal{N}_2^I, w_1\}$, $\{\mathcal{N}_3^I, w_1\}$, and $\{\mathcal{N}_1^I, w_2\}$, $\{\mathcal{N}_{11}^{FL}, w_1\}$, $\{\mathcal{N}_{12}^{FL}, w_1\}$, $\{\mathcal{N}_{13}^{FL}, w_1\}$, $\{\mathcal{N}_{14}^{FL}, w_1\}$, $\{\mathcal{N}_{21}^{FL}, w_1\}$, $\{\mathcal{N}_{31}^{FL}, w_1\}$, and $\{\mathcal{N}_{11}^{FL}, w_2\}$ are the intermediate components for connecting the initial new components with the selected predefined ones as well as the referencing components for modifying and refining $\mathcal{P}_9$ (*UC #2*), $\mathcal{P}_{10}$ (*KP #1*), $\mathcal{P}_{11}$ (*KP #2*), $\mathcal{P}_{12}$ (*BC #1*), $\mathcal{P}_{13}$ (*BC #2*), $\{\mathcal{N}_1^I, w_1\}$, $\{\mathcal{N}_2^I, w_1\}$, $\{\mathcal{N}_3^I, w_1\}$, and $\{\mathcal{N}_1^I, w_2\}$.

With the above newly determined new components, the other two open configuration sub-processes contribute to constraint processing, component modifications and refinements, and the establishment of connections among final configuration components. At the end, 6 laptops are configured and each of them contains predefined, modified, and new components. For illustrative simplicity, the components of one configuration are provided below.

- Selected predefined components: $\mathcal{P}_3$ (*CPU #3*), $\mathcal{P}_6$ (*MB #2*), $\mathcal{P}_8$ (*UC #1*), $\mathcal{P}_{10}$ (*KP #1*), and $\mathcal{P}_{12}$ (*BC #1*);
- Modified components: $\mathcal{P}_{10}^{\mathcal{M}}$ (*KP #1*) and $\mathcal{P}_{12}^{\mathcal{M}}$ (*BC #1*);
- New components: $\{\mathcal{N}_1^I, w_1\}$ and $\{\mathcal{N}_2^I, w_1\}$.

Based on his/her personal preferences or other considerations, the customer can select one from these 6 laptop configurations.

5 CONCLUSIONS

In response to the limitations of available product configuration solutions, open configuration is put forward as a new approach to configuring customized products while meeting both unforeseen and anticipated customer requirements. This study proposed an approach based on cloud computing to address one of the sub-processes of open configuration: new component determination. In the approach, new components for meeting unforeseen customer requirements are determined from different databases in the cloud. An example of laptop open configuration is used to demonstrate how the new components are determined.

In determining new components, we considered the components which are connected with the initial new components at two levels and assumed that two leveled components were sufficient for finding solutions. In the future, this assumption might be relaxed and components at more levels might be considered. In this study, we detailed the new component determination sub-process and left the other two sub-processes: constraint processing and component modification and refinement unexplained. We plan to complete them in the near future. As the implementation of open configuration would bring companies a competitive edge in product customization: providing product functions and features for unforeseen customer requirements, more investigations in the future are required. In the long term, research efforts should be geared towards open configuration knowledge modeling and representation, open configuration system development, open configuration evaluation, and so on.

ACKNOWLEDGEMENTS

We are very grateful to Dr. Andreas FALKNER from Siemens AG, Vienna, Austria for his constructive comments and suggestions which helped improve both the quality and presentation of the earlier versions of the paper.

REFERENCES

[1] Trentin, A., Perin, E., and Forza, C.: 'Product configurator impact on product quality', International Journal of Production Economics, 2012, 135, pp. 850-859.
[2] Franke, D.W.: 'Configuration research and commercial solutions', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 1998, 12, (04), pp. 295-300.
[3] Zhang, L.L.: 'Product configuration: a review of the state-of-the-art and future research', International Journal of Production Research, 2014, 52, (21), pp. 6381-6398.
[4] Forza, C., and Salvador, F.: 'Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems', International Journal of Production Economics, 2002, 76, pp. 87-98.
[5] Haug, A., Hvam, L., and Mortensen, N.H.: 'The impact of product configurators on lead times in engineering-oriented companies', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2011, 25, (2), pp. 197-206.
[6] Forza, C., and Salvador, F.: 'Product configuration and inter-firm co-ordination: an innovative solution from a small manufacturing enterprise', Computers in Industry, 2002, 49, pp. 37-46.
[7] Salvador, F., and Forza, C.: 'Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities', International Journal of Production Economics, 2004, 91, (3), pp. 273-291.
[8] Aldanondo, M., Rouge, S., and Ve, M.: 'Expert configurator for concurrent engineering: Cameleon software and model', Journal of Intelligent Manufacturing, 2000, 11, (2), pp. 127-134.
[9] Mittal, S., Frayman, F., and Sridharan, N.S.: 'Towards a generic model of configuration tasks', Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, USA, 1989, pp. 1395-1401 .
[10] Sabin, D., and Weigel, R.: 'Product configuration frameworks-a survey', IEEE Intelligent Systems, 1998, 13, (4), pp. 42-49.
[11] Zhang, L.L., Chen, X., Falkner, A., and Chu, C.: 'Open Configuration: a New Approach to Product Customization', Proceedings of the 16th Workshop on Configuration, Novi Sad, Serbia, 2014, pp. 75-79.
[12] Buyya, R., Yeo, C. S., Venugopal, S., and et al.: 'Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility', Future Generation Computer Systems, 2009, 25,(6), pp. 599-616.
[13] Al-Anzi, F. S., Salman, A. A., Jacob, N. K.: 'New proposed robust, scalable and secure network cloud computing storage architecture', Journal of Software Engineering and Applications, 2014, 7,(5), pp. 347.
[14] Hernández, I., Sawicki, S., Roos-Frantz, F., and et al.: 'Cloud configuration modelling: a literature review from an application integration deployment perspective', Procedia Computer Scinece, 2015, 64, pp. 977-983.
[15] Zheng, P., Lu, Y., Xu, X., and et al.: 'A system framework for OKP product planning in a cloud-based design environment', Robotics and Computer-Integrated Manufacturing, 2016 (in press).
[16] Zhang, Q., Cheng, L., Boutaba, R.: 'Cloud computing: state-of-the-art and research challenges', Journal of Internet Services and Applications, 2010, 1,(1), pp. 7-18.
[17] Nurmi, D., Wolski, R., Grzegorczyk, C., and et al.: 'The eucalyptus open-source cloud-computing system', Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 2009, pp. 124-131.

Benchmark for configuration and planning optimization problems: Proposition of a generic model

Pitiot Paul[1,2] and **Garcés Monge Luis**[1] and **Vareilles Elise**[1] and **Aldanondo Michel**[1]

Abstract. *Computer science community is always interested in « benchmarks », e.g. standard problems, by which performance of optimization approaches can be measured and characterized. This article aims at presenting our work to achieve a benchmark for concurrent product configuration and process planning (CPCPP) optimization problems. A benchmark is a set of reference models that represents a particular kind of problem. Product configuration and process planning are classic problems abundantly handled in the literature. Their coupling in an integrated model is a more and more handled complex problem; but there is a lack of benchmark in spite of the need expressed by the community during last configuration workshops [1]. In this article, we propose a generic model of CPCPP problems that is representative of existing real applications and studies founded on literature. We define relevant concepts and propose some configuration and evaluation patterns for product and process modelling. Our generic model is adapted for a high level and multi-disciplinary (functional and/or physical) representation of these environments. It shall allow us to supply a representative and complete benchmark, in order to accurately estimate the contribution of existing optimization methods*.

1 INTRODUCTION

Benchmarking of optimization approaches is crucial to assess performance quantitatively and to understand their weaknesses and strengths. There is numerous academic benchmarks associate with various classes of optimization problem (linear / nonlinear problems, constrained problems, integer or mixed integer programming, etc.). Studies, reports and websites of [2] [3] [4] [5] are particularly accomplished examples of existing optimization benchmark with a multitude of articles and algorithms benchmarked on great variety of test functions (see for example [6], [7] or [8]).

More than an academic tool, a benchmark should also be representative of real-world problems. For a specific domain, a benchmark represents a reference which should be used by company's decision-makers to select an approach or an algorithm. But it is not always easy for them to know of which theoretical case covers their practical case. Benchmark on configuration field could illustrate this aspect with various industrial cases: automotive [9], [10], [11], power supply [12], train design [13], etc. A data-base of industrials cases was started on [14] but it is not anymore maintained.

[1] University Toulouse – Mines Albi, France, emails: paul.pitiot@mines-albi.fr, vareille@mines-albi.fr, aldanond@mines-albi.fr, luis_ignacio.garces_monge@mines-albi.fr.
[2] 3IL – CCI Aveyron, France

CPCPP problem is an interesting and increasingly studied industrial problem. It gathers in single model two classical domains for a stakeholder: which product corresponds to customer's needs and how it is going to be obtained? A decision aiding tool has to assist stakeholder to make the best decisions (product configuration and process planning choices) according to multiple objective. CPCPP problems take place in the first steps of the study of product and associated process. It thus manipulates a high level description model.

Ours previous research projects [15] aim at produce decision aiding tools for a specific problem subject to a growing interest in mass customization community: the coupling between product and process environments. Numerous authors showed the interest to take into account simultaneously the product and process dimensions in a decision aiding tool. This concurrent process has two main interests: i) Allowing to model, and thus to take into account, interactions between product and process (for example, a specific product configuration forbids using certain resources for process tasks), ii) Avoid the traditional sequence: configure product then plan its production which is the source of multiple iterations when selected product can't be obtained in satisfying conditions (mainly in terms of cost and delay).

In spite of the growing interest of the community and industrialists, there is no standard (benchmark) for this problem. In this article, we propose a generic model of the whole problem (configuration, planning and coupling). Therefore, the paper is organized as follow. The next section details the problem and its combinatorial aspect. The third section proposes a generic model of the problem. Some elements associated with cases diversity are discussed.

2 ADDRESSED PROBLEM

For our benchmark, the addressed problem is limited to the coupling between product configuration and process planning. We will describe both environments and the coupling of them in next sub-sections.

2.1 Concurrent configuration and planning

Product configuration problem is a multi-domain, multidisciplinary, multiobjective problem [16], [17]. That generates a wide diversity of possible models to represent. We will try to define a classification of existing product models and model it in the proposed generic model. Planning problems are generally more framed (e.g. temporal precedence, resources consumption, due date or delay, etc.). To generate various problem instances we

can act on the shape of the process graph and on the dispersal of the values assigned for the resources of tasks (cost, delay, etc.). Thus, we need to define in our generic model of the product / process a kind of generic model for each part and for the coupling.

Many authors, since [18], [19], [20] or [21] have defined configuration as the task of deriving the definition of a specific or customized product (through a set of properties, sub-assemblies or bill of materials, etc.) from a generic product or a product family, while taking into account specific customer requirements. Some authors, like [22], [23] or [24] have shown that the same kind of reasoning process can be considered for production process planning. They therefore consider that deriving a specific production plan (operations, resources to be used, etc...) from some kind of generic process plan while respecting product characteristics and customer requirements, can define production planning. More and more studies tackle the coupling of both environments [24] [25] [26] [27] or [28]. Many configuration and planning studies (see for example [29] or [30]) have shown that each problem could be successfully considered as a constraint satisfaction problem (CSP). CSP's are also widely used by industrials [31]. Considering that using a CSP representation, we could both represent constrained and unconstrained problems, we will use it to represent each environment and the coupling.

2.2 Combinatorial optimization problem

In previous concurrent model, some variables represent decisions of the user (customer or decision-maker on product or process environment). We assume that those decision variables are all discrete variables, so that an instantiation of all these decisions variables corresponds to a particular product / process. Indeed in reality and regardless of the environment, decisions correspond to choices between various combinations. In product environment, decisions correspond to architectural choices between various combinations of sub-systems, or to a choice among various variants for every sub-system. In process environment, decisions correspond to resources choices between various variants.

Combinatorial constrained optimization problem consists in a search of a combination for every decision variables that respects constraints of the problem [32]. Instantiation of every decision variable in CSP model corresponds to a specific product/process which could be analyzed and scored according user's multiple preferences or objectives (cost, delay, etc.). As those objectives could be antagonist, algorithm has to find in a short time a set of approximately efficient solutions that will allow the decision maker to choose a good compromise solution. Using Pareto dominance concept, the optimal set of solutions searched is called the optimal Pareto front.

This thus allows us to define a multi-objectives combinatorial constrained optimization problem: a search between various combinations to find a selection of solutions which are the closest possible of the optimal Pareto front.

3 GENERIC MODEL DESCRIPTION

Our goal is to define a benchmark for CPCPP optimization problems. A benchmark is a set of model instances representative of a specific optimization problem and which allows testing of optimization algorithms and validation of their accuracy for the addressed problem. We need to generate various instances of the CPCPP problem that represented diversity and complexity of real industrial cases. Thus we first need a definition of a generic model of addressed problem and then, to create the selection of varied model instances to get the benchmark.

In following sections, we will describe each part (Product configuration, process planning and their coupling) by a generic model using optimization constraint satisfaction problem (O-CSP) paradigm. That means that the problem is defined by a quadruplet <V, D, C, f > whit V the set of variables, D the set of domains linked to each variable, C a set of configuration constraints that correspond to compatibilities between values of variables ; and f a specific set of constraints, called in this study evaluation constraints, that allow to calculate multivalued objective function.

A sub set of V can also been identified as the set of decision variables, named Vd. In a decision aiding problem, Vd corresponds to the set of variables on which stakeholder can act. Each decision variable is related to one or more objective. Decision variables are discrete (numeric or symbolic). Product and process configuration corresponds to the selection of a value for the set of decision variables. The aim of O-CSP is to find the setting of decision variables that will maximize/minimize objectives. In this study, others variable of V will be called evaluation variables because they allow to calculate the value of the objectives.

In this study, we consider only "global selling price" and "process cycle time" objectives; others could be added as product performance or carbon foot print. Objectives are represented by two continuous variables (*global_sp* and *process_cyt*). Their domains are unions of intervals. Global selling price is impacted by both product and process domains, whereas process cycle time only steams from process domain.

We have to define what we mean by selling price. Cost and selling price are two different way to economically evaluate a product. Costs are what pays the enterprise (raw material, components or resources) while selling price is what customer pays. Coming from value analysis domain, the functional descriptive variables are related with selling price while components and resource are related with respectively the product cost and the process cost. As this decision aiding problem takes place during negotiation with customer, internal costs has to be hidden and they will be changed in selling price (commercial strategy is included like margins or discounts).

Definition 1: *Selling prices represent either costs of the components or of the operation, or selling prices of product characteristics. Their values are calculated using evaluation constraints according to configuration of decision variables.*

We decompose the analysis of the CPCPP problem in four parts: Product configuration, process planning, configuration constraints and evaluation constraints. In each part, we will define relevant concepts in O-CSP paradigm and also make some hypothesis according to reality of existing CPCPP industrial problem.

3.1 Product configuration

Definition 2 *Product or System gathers a set of physical-functional modules in a one level decomposition.*

Definition 3 *physical-functional module is a sub-set of a product that corresponds to a set of components and which fulfills some functions of the product.*

Those definitions come from theory of axiomatic design [33] and design structure matrix (DSM) [34]. In a DSM, modules are logical group of components linked to a set of functions. For example, if configured product is a car, modules can be audio system, engine system, etc.

Let define a configurable product as a group of *nb_modules* physical-functional modules with *nb_modules* greater than 1. This definition of a module could be used to model either physical or functional decomposition or also both at same time. This can be helpful to interact with a customer which as eventually requirements on the two aspects and also allow us to represent the wild diversity of existing models.

1. Description of a physical-functional module
   - Physical description (component view)

As we are in a configuration context, we assume that every component mentioned in the definition 2, is exchangeable by others component that fulfill same functions and thus each one corresponds in fact to a family of components (*foc*).

Definition 4 *family of component (module_i_foc_j) is a set of components that can fulfill some required functions in module i. module_i_foc_j are discrete decision variable in Vd.*

Each module can gathered multiple families of components. For example, the audio module gathers families of components "central unit" and "speakers". Each family corresponds to a set of equivalent component. For example, the family of component "central unit" contains possible catalogue references for the central unit. Let define *module_i_nb_foc* the number of family of components of module *i*.

   - Functional description (function view)

A module could be described by its functional point of view. Functions are linked to customer's requirements. In the same way as in physical description, this model takes place in a configuration context. Thus functions describe components abilities according to various discrete functional levels.

Definition 5 *function is a fulfillment of a customer's requirement over various discrete functional levels.*

Each function fulfilled by the module can be described using a functional description variable (*fdv*) in a one high-level functional description. Each function can be fulfilled according to a defined number of levels. For example, a functional description variable on audio module can be the "global audio power" with possible levels: 200w, 300W, 400W, etc. Another example could be "USB input" with two levels: yes or no.

Definition 6 *Functional description variable (module_i_fdv_j) refers to description of a function and is a set of possible functional level for a function in module i.*

Each module can gather multiple functional description variables. Let define *module_i_nb_fdv* the number of description variable of module *i*.

2. Decision aiding aspect

According to the high level description of CPCPP problem, each aspect (physical or functional) is a one level decomposition of modules, families of component or description of functions. In such high level description, a module corresponds to a small number of families of component and fulfills a limited number of functions. It has to be described by at least one family of component or one functional description variable and maximum 10 of both. In a decision aiding problem, user act on decision variable to analyze their impact on objectives.

Definition 7 *In product domain, decision variable are choice of components in each family of component and choice of functional level, for each functional description variable and this for every module of the product.*

3. Evaluation of a physical-functional configuration

Aim of study is to help configuration and planning according to multiple criteria. We thus need to model variables and constraints involved in each criterion calculation. In product configuration domain, only selling price criterion is taken into account (see section 3.3 for more details on selling price concept).

In a module, the selling price variables are linked either with a family of components variable or with a functional description variable. Every selling price variables are continuous and their domains are represented by intervals. They will allow calculating selling price of a module according to choices on decision variables. Let define *module_i_nb_sp* the number of selling price in module *i*. We also add a selling price for each module named *module_i_sp*. Value of this variable will be the sum of every selling price variables in this module *i*. In the same way, a product global selling price (*product_sp*) is added to the model. Value of this variable will be the sum of every *module_i_sp* variables.

4. Configuration/Evaluation patterns in product domain

The product configuration contains a high diversity due to the multiplicity of products. Nevertheless, we observe in the majority of the product configuration models, some small groups of strongly connected variables. Coming from our experience on product configuration, we identify a set of common generic sets of variables and constraints called Product Configuration/Evaluation Pattern (PCEP) in our generic model showed in figure 1. Each PCEP gathers a set of decision (*foc* and/or *fdv*) and objective (*sp*) variables linked by constraints (configuration and evaluation constraints, see section 3.3 for more details on constraint definitions) and corresponds to a physical-functional description of a part of a module. We thus define modules as a set of PCEP.

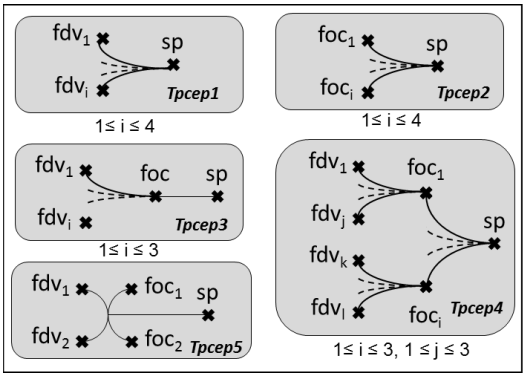


Figure 1. Various types of PCEP

On figure 1, the two first type of PCEP (Tpcep1 and Tpcep2) gather a configuration and an evaluation constraint. They correspond to a one point of view analysis, functional (Tpcep1) or physical (Tpcep2). They express the fact that a set of compatible components or functions is linked to a specific selling price. There is one selling price value or interval for each compatible value of foc or fdv variables.

The two next PCEP (Tpcep3 and Tpcep4) sequences configuration constraints between fdv variables and foc variables, and one evaluation constraint between foc variables and the selling price variable. It represents one or more set of functions carried out by one family of component. Selling price is then inferred from corresponding foc variables. With this kind of PCEP, user can express his requirements either at the functional level or at the physical level (components) but once one side is fulfilled, the other is deduced. For example for the audio sound system of a car, he could select the audio power output and other option he need, that will lead to a compatible selection of audio central unit or directly select this one (one or more specific references). Selling price is linked to selected component.

The last PCEP (Tpcep5) merge configuration and evaluation constraints in the same constraint. It represents more heterogeneous links between functional and components description where compatible combinations of functional levels correspond to compatible combinations of components. We limit this kind of behavior to two fdv and two foc.

3.2 Process planning

In this section, we define a generic model of production process associate to the configurable product. Generic aspect expresses facts that one process corresponds to one product, and that process decomposition (sequence of operation) is the same for each possible associate product configuration.

Definition 8 *process is the sequence of operations that leads to obtain relevant configurable product in one-level decomposition.*

The ordering and the number of operation are static. It means that there is neither OR node on the sequence nor operation activation according to a specific product configuration.

In this section, we define all variables needed to a generic description of an operation showed on figure 2. Each operation generates a specific work load (*operation_i_wl*) to achieve on a family of resource (described on next section). This work load could be constant (for example, a packing work load equal two man-month whatever the product is) or could be influenced by product choices (for example a big component need more work load to be assembled). The work load will be linked to selling price (*operation_i_sp*), duration and resources selection (type and quantity, see next section). Work load isn't a decision variable.

Our aim in this process planning domain is to support stakeholder to achieve a planning by acting only on the resources selection (type and quantity) for each operation.

Definition 9 *An operation is a step of production process that corresponds to specified workload to be achieved using a specified type and quantity of resource. It has a duration ensuing from the choice of the type and from the quantity of selected resource.*

We assume infinite capacity planning because, in such high level description, there is a little probability that two macro-operations use the same macro-resource, and we consider that production is launched according to each customer order and production capacity is adapted accordingly.

In our generic model, we define a generic process as a set of *nb_operation* generic operation linked by temporal relations and there is at least one operation.

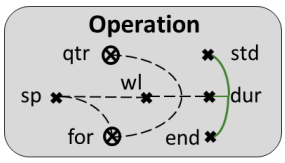


Figure 2. Generic description of an operation

1. Description of an operation
   - Temporal description

In order to plan the process, we need to model the temporal placement of each operation. In a constraints modeling, we need 3 continuous variables for each operation *i* to describe its temporal placement: starting date (*operation_i_start*), ending date (*operation_i_end*), and duration (*operation_i_duration*).

Value of duration variable can be obtained by constraints processing according to selected values of resource description variables (type and quantity) and to amount of workload to achieved, whereas start and ending dates will be planned according start/ending date of previous and next operations and according to selected duration.

   - Resource description

In this kind of configurable process, we define process configuration as selection of resource type and quantity. That leads that for each operation, there is a set of exchangeable resources that can be used to achieve works relevant to this operation. This set is represented by a discrete variable called a family of resource (*operation_i_for*). We assume that there is only one resource family for each operation. In order because if there were multiple resources needed for an operation; firstly they could be aggregate in one family of aggregated resource (for example, an operator resource and a machine resource could be aggregated in one operator-machine resource); and secondly in first steps of product/process study, stakeholder is interested in only critical resource dimensioning.

Definition 10 *A resource family is the set of resource that can achieve works relevant from a specific operation.*

To achieve an operation, stakeholder could also act on quantity of selected resource. So for each operation *i*, generic model include a discrete variable named quantity of resource (*operation_i_qtr*).

2. Decision aiding aspect

In process domain, decision variable are choices of resource in the family of resource and choices of quantity of resource for each operation *i*.

3. Objective valuation in process domain

In process planning domain, we consider two criterions: Process cycle time and selling price. The process cycle time will be computed according to precedence constraints between operation (see evaluation constraints in section 3.4) and duration of operations. Duration of an operation is linked to ability of resources and quantity of resource selected. Selling price of an operation is related to costs of resources selected according to the workload needed. To simplify formulae of selling price calculation, we also add a process global selling price named *process_sp*. Value of this variable will be the sum of every operation selling price variables.

3.3 Configuration constraints

Constraints could be an equation or a compatibility table according to nature of variables involved (continuous or discrete). Compatibility table shows allowed and/or forbidden combinations of values of variable. In our generic model, configuration constraints are exclusively represented by tables of compatibility that link decision variables; while evaluation constraints use equations and tables of compatibility.

Definition 11: *Configuration constraint describes compatibility between values of a set of decision variable. As decision variable are discrete, it corresponds to a compatibility table.*

For example, we could have incompatible components or a component incompatible with a specific functional level or with a specific family of resource (coupling configuration constraint), etc. A constraint is first defined by an arity, e.g. the number of involved variable. In a realistic CPCPP problem, arity of configuration constraint is from two to four decision variables.

In our generic model, a configuration constraint is defined by its type in the model (i.e. depending on nature of decision variable they link), by a type of configuration pattern they illustrate and a constraint density. We define in next sections those concepts.

1. Type of configuration constraints

Configuration constraints take place in different location of the model. We list those locations illustrated on figure 4 and define their placement rules according to each location:

1. Intra-PCEP constraints already defined in section 3.1
2. Inter-PCEP constraints: In a same module, PCEP's variables could be link to other PCEP's variables to describe incompatibilities between various components or functions from different PCEP. We assume that those configuration constraint link variables of the same kind so fdv to fdv variable and foc to foc variable. In order because PCEPs entirely describe links between functional and physical aspects. For a realistic model, we limit the arity of those constraints to three maximum.
3. Inter-Module constraints: In modular product, there are more relations between variables of a same module than relations between variables of different modules (definitions of architectural design and modularity). This kind of relation can nevertheless exist and they follow the same rules than inter-PCEP configuration placement (i.e. they link same kind of decision variable and there arity is limited to three).
4. Process configuration constraints: We assume that there are few configuration constraints in process domain (in comparison with product domain). Those constraints could link either two family of resource decision variables from two different operations (for example two incompatible subcontractor in two following operations) or decision variables of the same operation (for example two resources of the same family with different quantity available).
5. Coupling configuration constraints: Coupling constraint links one or two variables from one or two modules in product side, to the family of resource decision variable or/and workload variable of one operation on process side. Coupling constraints corresponds to compatibilities and incompatibilities between product and resources of process (for example a big component could need a big crane to be assembled) or a workload dimensioning according to product configuration. In a realistic CPCPP problem, arity of a coupling constraint is four maximum.

The last placement rule is to avoid circuit in constraints graph.

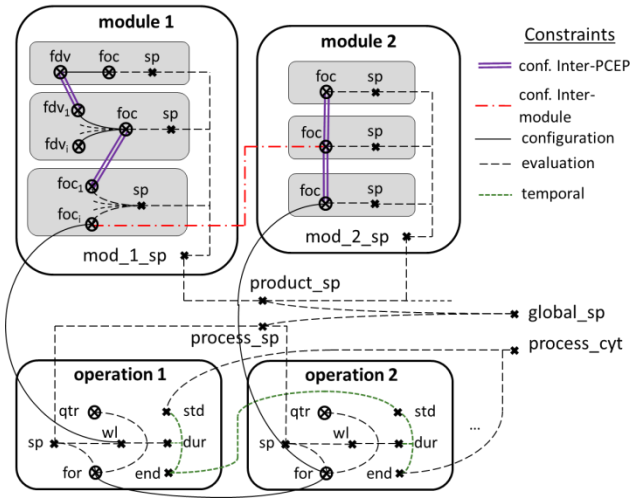


Figure 3. Example of the whole model

2. Type of configuration pattern

Coming from our experience and literature review, we identify various configuration shapes or behavior named type of configuration pattern (Tcp) presented in following list and showed on figure 3. Those patterns rest on supposed ordered values of variables in terms of performance or abilities. Examples on figure 3 show configuration constraints between two variable but they could be extended to three or four variables. Compatibility tables are illustrating by compatibility matrix. A cross in matrix corresponds to a compatible couple of values. Each value of each involved variable has to appear in at least one allowed tuple. (i.e. we have at least one cross in each line and each column in associated matrix).

1. Similarity or close level pattern (Tcp1): This pattern corresponds to a similarity between various levels of decision variables (parameter: *diff_level* difference of between similar levels). It can represent a conception rule to maintain coherence between choices on a product. For example, this could lead to forbid the simultaneous selection of luxurious and cheap components.
2. Dissimilarity pattern (Tcp2): This pattern corresponds to a dissimilarity constraint between various values of decision variables (parameter of pattern: *diff_level* difference of between dissimilar levels). It can represent a conception or organizational rule that forbids the selection of same level for selected variables. For example, this could lead to forbid the selection of similar dangerous material for various component or similar suppliers for different operations.
3. Limit Pattern (Tcp3): This pattern corresponds to a limit between various levels of decision variables (parameter of pattern: Extremum (min or max), operator (sum or product), value of the limit). It can represent a physical limitation. For example, it could be a limit for power supply that impacts components consumption or a size limit for a crane that handle product (coupling config.).

4. Comparing pattern (Tcp4): this pattern corresponds to a comparison between levels of decisions variables (Parameters of pattern: Comparison sign). It can represent for example an engine power requirement that could be fulfilled by various engines.

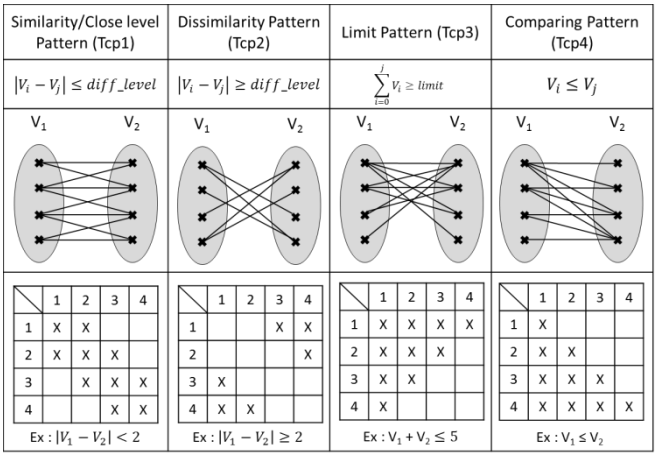


Figure 4. Types of configuration constraint pattern

3. Constraint density of a configuration constraint

Constraint density for one constraint corresponds to the ratio of allowed tuples over every possible tuple. It coincides to the ratio of crosses in the compatibility matrix.

3.4 Evaluation constraints

Evaluation constraints correspond to fitness function *f* calculation in O-CSP paradigm. They allow assigning a value to various objectives for a specified product or process.

Definition 12: *Evaluation constraint links an evaluation variable (selling price variables or temporal description variable for operations) either to a set of decision variable, or to a set of others evaluation variables. It allows calculating values of this evaluation variable according to either values of related decision variables using a table of compatibility, or values of others evaluation variables using equations.*

CPCPP problems are challenging for optimization tool for many reasons: multiobjective aspect, size of search space and complexity of objective function. This last difficulty mainly results from:

1. Close performance skills of components and resources (i.e. a dominated component (in the Pareto sense) will be a priori exclude from catalog before optimization and configuration steps).
2. Multiplicity of elementary behaviors that cannot be model by a global mathematical representation.
3. Marketing and innovation aspects in selling price concepts introduce some singularities in evaluation model

The evaluation constraints gather three kinds of constraints:

1. The first one is the set of constraints that links decision variables (*module_i_fdv*, *module_i_foc*, *operation_i_for* or *operation_i_qtr*) to one evaluation variable (selling prices in product side and *operation_i_sp* and *operation_i_dur* in process side). As decision variable are discrete, those constraints are tables of compatibility.
2. The second set corresponds to temporal description of the process and allows calculating cycle time objective. It gathers precedence links between duration, starting date and ending date variables of operations. Those constraints called temporal evaluation constraints correspond to some inequalities and one equation.
3. The third set of constraints will allow selling price aggregation by linking every elementary selling price variable to global selling price variable using some equations. Those ones will be named selling price aggregation constraint.

On following section, we will define evaluation constraints on each domain, for the coupling and global aggregation ones.

1. Evaluation constraints on product domain

Evaluation constraints on product domain gather one evaluation constraint for each PCEP on each module and aggregation constraints that link selling price variables.

As explained in PCEP description (see section 3.1.4), each PCEP gathers one evaluation constraint. This one links a set of decision variable to one selling price variable using a compatibility table. The compatibility table assigns a value or an interval of value to the selling price variable for every possible combination of decision variables. As we aim at produce a real-world behavior in generated model, the compatibility table has to be fulfilled with coherent values. Coming from our experience, we identify some evaluation patterns used to fulfill this table.

An evaluation pattern corresponds to the variation of an evaluation variable according to a choice on one or more decision variable. Considering ordered values of decision variable, we could define that a decision variable is either positively or negatively correlated to an objective. This correlation is not necessarily linear (Tep1), it could be quadratic (Tep2) or piecewise linear (Tep3). Those three behaviors are illustrated on figure 4. We illustrate on Tep1 a possible repartition of selling price values linked to one decision variable with 5 values. This kind of relation is adaptable to a combination of several variables of decision.

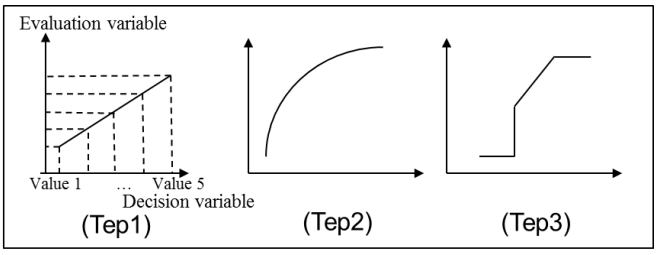


Figure 5. Types of evaluation pattern

As evoked in introduction of this section, we must include some singularities to those mathematical elementary behaviors. We could randomly select and change values of some combinations according to singularity rate settled globally for the whole model.

In each module, all elementary selling price variables are aggregate on *module_i_sp* variable:

$$\text{module\_i\_sp} = \sum_{j=1}^{mod\_i\_nb_{sp}} \text{module\_i\_sp}_j \qquad (1)$$

Then all *module_i_sp* are aggregate on *product_sp*:

$$\text{product\_sp} = \sum_{i=1}^{\text{nb\_mod}} \text{module\_i\_sp} \qquad (2)$$

2. Evaluation constraints on process domain

Evaluation constraints on process domain gather two evaluation constraint for each operation (one for the selling price objective and one for the duration), temporal evaluation constraints and finally a selling price aggregation constraints that link selling price variables (*operation_i_sp*) to global one (*process_sp*).

For each operation, evaluation constraints must define a value for each evaluation variable for each possible combination of decision variables (*operation_i_for* and *operation_i_qtr*). The work load is also involved in objective evaluation. It corresponds to work to be realized according to the chosen product.

The selling price of an operation is defined by a compatibility table that gathers all possible combination of workload and family of resource.

$$\text{process\_sp} = \sum_{k=1}^{\text{nb\_ope}} \text{ope\_k\_sp} \quad (3)$$

Global selling price is an aggregation of product and process selling prices.

$$\text{global\_sp} = \text{product\_sp} + \text{process\_sp} \quad (4)$$

On the other hand, duration of an operation is defined by a compatibility table. It gathers all possible combination of workload, quantity and family of resource.

On each operation,

$$\text{operation\_i\_dur} = \text{operation\_i\_end} - \text{operation\_i\_stt} \quad (5)$$

Then between operations, precedence constraints result from graph shape. Consecutive operations have to be linked by precedence constraints. If an operation j follows an operation i:

$$\text{Operation\_i\_end} < \text{operation\_j\_stt} \quad (6)$$

Finally, the process cycle time is the difference between starting date of the first operation and the ending date of the last one.

$$\text{process\_cyt} = \max_{\forall j} \text{operation\_j\_end} - \min_{\forall i} \text{operation\_i\_sdt} \quad (7)$$

4 MODEL GENERATION

The generic model described in previous section is used to create a set of instances in order to constitute the CPCPP benchmark. We currently develop a model generator that implements this generic model definition to achieve this task. As our aim is to create models representative of real-world problems, the model generator will be available on-line and proposed to some industrial partners so that they can create representative models of their products and process. We shall complete this set with models representing the existing cases in the publications on this subject.

To simplify model generation by a user, we have defined a procedure based on a set of simple questions and indicators. Without revealing sensitive industrial data, the user can define a representative model in term of: number of modules, number of operations, number of variables of decision, average number of alternatives for each decision variable, selection of various types of PCEP, configuration or evaluation patterns, etc.

We also define some simple ways to characterize and choose the distribution of constraints in the model. To connect PCEP in a module, modules themselves or for the coupling, we have defines four different levels: unconnected, weakly, averagely and strongly connected. For example for PCEP connecting, the first one (unconnected) corresponds to separate patterns; “Weakly connected” corresponds to some constraints between patterns (randomly from 1 to the number of pattern); “Averagely connected” corresponds to a module with every patterns connected (i.e. if there is n patterns, number of inter-PCEP constraints in the module will be n-1); and “strongly connected” corresponds to links between every pattern in the module (i.e. if there is n patterns, number of constraints in the module will be n! / (2*(n − 2)!), combination of 2 between n modules).

Finally, we define some useful indicators to handle globally the model: inter-PCEP constraints rate, modularity rate (i.e. the ratio of constraints between modules with regard to the total number of constraints in the product model), coupling rate (i.e. the ratio of constraints between product variables and process variables with regard to the total number of constraints in the model) and process configuration rate (i.e. the ratio of constraints between process variables with regard to the total number of constraints in the model).

CONCLUSION

The goal of this research paper was to present our research perspectives for a benchmark on concurrent configuration and planning. This problem is more and more studied. Although there are a lot of cases of Knowledge-based configuration systems applied on the industrial practice and project planning, there is a real lack of real-word inspired benchmark. In this study, we propose a generic model that can represent this diversity and that will allow to generate various test models. Coming from our experience in CPCPP problem and analysis of existing works, some configuration and evaluation pattern are proposed. We seek for comments of the community and industrials to improve our proposition. Then we will create a website with the benchmark and the model generator to diffuse our proposition and better evaluate optimization approaches on this CPCPP problem.

REFERENCES

[1] Workshops on configuration, 2013/2014/2015: 2013: http://ws-config-2013.mines-albi.fr/, 2014: http://confws.ist.tugraz.at/ConfigurationWorkshop2014/, 2015: http://blogs.helsinki.fi/confws-15/

[2] O. Shcherbina, COCONUT benchmark, http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html, 2009.

[3] F. Domes, M. Fuchs, H. Schichl and A. Neumaier. The Optimization Test Environment, *Optimization and Engineering*, vol. 15, pp. 443–468, (2014)

[4] H. Mittelmann, Benchmarks, http://plato.asu.edu/sub/benchm.html, (2009).

[5] J.C. Gilbert and X. Jonsson. LIBOPT - An environment for testing solvers on heterogeneous collections of problems - The manual, version 2.1. Technical Report RT-331, INRIA, (2009).

[6] O. Shcherbina, A. Neumaier, Djamila Sam-Haroud, Xuan-Ha Vu and Tuan-Viet Nguyen, Benchmarking global optimization and constraint satisfaction codes, pp.211--222 in: Ch. Bliek, Ch. Jermann and A. Neumaier (eds.), *Global Optimization and Constraint Satisfaction*, Springer, Berlin (2003).

[7] László Pál, Tibor Csendes, Mihály Csaba Markót, and Arnold Neumaier Black Box Optimization Benchmarking of the GLOBAL

Method, , *Evolutionary Computation*, Vol. 20, No. 4 , pp. 609-639, (2012)
[8] A. Auger and R. Ros, Benchmarking the pure random search on the BBOB-2009 testbed. In Franz Rothlauf, editor, GECCO, pp 2479–2484. ACM, (2009)
[9] J. Amilhastre, H. Fargier, P. Marquis, Consistency restoration and explanations in dynamic csp's - application to configuration, in: *Artificial Intelligence* vol.135, pp. 199-234, (2002)
[10] A. Kaiser, K. Wolfgang, S. Carsten. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design*, Analysis and Manufacturing, 17(2), Special Issue on configuration, (2003)
[11] Sinz, C., Kaiser, A., Küchlin, W., Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 17, pp.75-97, (2003)
[12] Jensen, R., Lars, S.: Power Supply Restoration, *Master's thesis*, IT University of Copenhagen, (2005).
[13] S.Han, J. Lee. Knowledge-based configuration design of a train bogie, *Journal of Mechanical Science and Technology*, Vol. 24, Issue 12, pp 2503-2510, (2011).
[14] Subbarayan, http://www.itu.dk/research/cla/externals/clib/, (2006)
[15] P. Pitiot, M. Aldanondo, E. Vareilles, P. Gaborit, M. Djefel, S. Carbonnel, Concurrent product configuration and process planning, towards an approach combining interactivity and optimality, in: *I.J. of Production Research*, vol. 51 n°2, pp. 524-541, (2013)
[16] V. Viswanathan and Julie Linsey, Spanning the complexity chasm: A re-search approach to move from simple to complex engineering systems. *AI EDAM* 28(4): pp. 369-384, (2014).
[17] I. Tumer and K. Lewis. Design of complex engineered systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28, pp 307-309, (2014).
[18] S. Mittal, F. Frayman. Towards a generic model of configuration tasks, *proc of IJCAI*, p. 1395-1401(1989).
[19] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, Towards a General Ontology of Con-figuration., *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol 12 n°4, pp. 357–372, (1998)
[20] M. Aldanondo, E. Vareilles. Configuration for mass customization: how to extend prod-uct configuration towards requirements and process con-figuration, *Journal of Intelligent Manufacturing*, vol. 19 n° 5, p. 521-535A (2008)
[21] P. Hofstedt, D. Schnee-weiss. FdConfig: A Constraint-Based Interactive Product Configurator. *19th International Conference on Applications of Declarative Programming and Knowledge Management*, (2011).
[22] K. Schierholt. Process configuration: combining the principles of product configuration and process planning *AI EDAM* , Volume 15, Issue 05, pp 411-424, (2001)
[23] R. Barták, M. Salido, F. Rossi. Con-straint satisfaction techniques in planning and scheduling, *Journal of Intelligent Manufacturing*, vol. 21, n°1, p. 5-15 (2010)
[24] L. Zhang, E. Vareilles, M. Aldanondo, Generic bill of functions, materials, and operations for SAP2 configuration, *I.J. of Production Research*, Vol. 51 n°2, pp. 465-478, (2013)
[25] Baxter, D. An engineering design knowledge reuse methodology using process modelling. *Research in Engineering Design*, 18 (1) pp. 37-48, (2007)
[26] G. Hong, D. Xue, Y. Tu,, Rapid identification of the optimal product configuration and its parameters based on customer-centric product modeling for one-of-a-kind production, *Computers in Industry* Vol.61 n°3, pp. 270–279, (2010)
[27] L. Li, L. Chen, Z. Huang, Y. Zhong, Product configuration optimization using a multiobjective GA, *I.J. of Adv. Manufacturing Technology*, vol. 30, pp. 20-29, (2006)
[28] Huang, H.-Z. and Gu, Y.-K., Development mode based on integration of product models and process models. *Concurrent Engineering: Research and Applications*. Vol.14, 1, pp 27-34, (2006)
[29] U. Junker, *Handbook of Constraint Programming*, Elsevier, chap. 24 Configuration, p. 835-875, (2006)
[30] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results, *Artificial Intelligence*, vol. 143, pp 151-188, (2003)
[31] A. Kaiser, K. Wolfgang, S. Carsten. Proving consistency assertions for automotive product data management. *J. Automated Reasoning*, 24(1-2):145-163, (2000)
[32] E. Mezura-Montes, C. Coello Coello, Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future, *Swarm and Evolutionary Computation*, Vol. 1 n°4, pp. 173-194, (2011)
[33] N.P. Suh, Axiomatic Design: Advances and Applications, Oxford University Press, (2001)
[34] Steward: The Design Structure System: A Method for Managing the Design of Complex Systems, *IEEE Transactions on Engineering Management*, vol 28(3), S. 71-74, (1981)

Optimal Feature Selection via Evolutionary Algorithms and Constraint Solving

Yibo Wang and **Lothar Hotz** [1]

Abstract. In software development, product lines especially with feature models are promising technologies to manage variability of software products. A new challenge in deriving software products from the product line is not only to select a set of features, which do not incur any feature inconsistencies, but to optimize multiple objectives (e.g. cost minimization and maximization of feature reuse) at the same time. This challenge is a constrained multi-objective optimization problem and has been proved to be difficult. In this paper, we approach this problem by utilizing the state-of-the-art method SATIBEA, which combines a multi-objective evolutionary algorithm with constraint solving techniques. The contribution of our approach is that we enhanced SATIBEA in two ways: by improving its mutation operator and by providing a novel crossover operator. Our empirical experiment results have shown that our approach SATIBEA+ improved SATIBEA noticeably, in providing more valid and more qualitative feature selections in terms of the standard measures such as hypervolume and Pareto front size.

1 INTRODUCTION

In many engineering fields, more and more emphasis is put on product line technology due to the need for customization with low efforts and in short terms. Planned variability, expressed in product models such as feature models [8], allows a smart configuration of products and services. However, products and services have to fulfill growing non-functional requirements, such as safety or cost, due to the increased competition on the markets. Such requirements more often demand for optimizations for achieving best fitted products to users' needs. Meanwhile, multiple non-functional requirements have to be considered together during optimization, although they might be non-commensurable or competing.

The field of product configuration is trying to cope with such problems. A configuration is a description of all parts with their appropriate parameters that are needed to build the product that hopefully will fulfill the given requirements. In product configuration, the configuration problem starts from certain user requirements and from a configuration model which implicitly describes all configurations of a certain domain. By using reasoning technology (e.g. a SAT-Solver [16]), the configuration problem is solved by automatically creating configurations. In the specific case of software product lines (SPL), configuration models are often expressed by feature models representing all features of a product and the configuration consists of selected features from such models. In this paper, this general task is further enhanced by taking optimization into account.

[1] Department of Informatics, University of Hamburg, Germany, email: wang@informatik.uni-hamburg.de

Technologies that approach this problems are SAT-solvers that allow the computation of valid feature configurations [7, 16] and evolutionary algorithms that are capable to compute solutions for multi-objective optimization problems [5, 10, 12, 14]. Evolutionary algorithms start with a set (*an initial generation*) of individuals (here, potential feature configurations), and modify them by mutation and crossover in the following generations. A mutation changes one individual (here, by selecting other features) and crossover combines a few individuals to a further individual (here, a new configuration generated by combination).

Syyad et al. [12] showed that for solving multi objective problems, especially with an increased number of objectives, indicator-based approaches (IBEA) outperform dominance-based ones. Dominance-based approaches rank solutions according to absolute dominance while indicator-based approaches provide an "amount" of dominance by computing a value which incorporates user preferences. Based on this, Henard et al. [5] introduced SATIBEA, which applies a SAT-solver in the mutation operator to correct a configuration for returning a valid mutation. However, we have found that SATIBEA often fails for complex feature models with thousands of features and constraints. In addition, the standard 1-point crossover used in SATIBEA generates in most cases worse offsprings (with more feature violations) than their parents, due to an arbitrary combination from parents. Thus, our work complements the existing work of SATIBEA (leading to SATIBEA+). The main contributions of our approach SATIBEA+ can be summarized as following:

- We improve mutation operator of SATIBEA so that it finds more valid configurations for complex feature models.
- We provide a novel crossover operator which reduces the number of feature violations for an invalid solution by learning from another configuration.
- We show that SATIBEA+ outperforms SATIBEA, in providing more valid and qualitative configurations in terms of hypervolume and Pareto front size.

The paper is organized as follows: Section 2 presents the underlying technologies which are used in our approach (Section 3). For evaluating this approach, we defined research questions (Section 4) and verify them through experiments (Section 5 and Section 6). The sections 7, 8, and 9 provide discussions, related work, and a conclusion.

2 BASIC TECHNOLOGIES

In SPL, the variability of products (i.e., the configuration space) is typically represented as feature models. They express all decision variables that are subject to select and optimize (Subsection 2.1). A main aspect of this paper is to support the combination of technologies that are capable to compute consistent configurations and

that are capable to optimize those configurations. In our approach, we use SAT-solvers for consistency checking (Subsection 2.2) and evolutionary algorithms for computing multi-objective optimization problems (Subsection 2.3). The rest of the paper shows, how we incorporate both technologies, i.e., the SAT-solver used as a mutation method for the evolutionary algorithm.

2.1 Feature Models and Feature Constraints

In software product lines, *feature models* are often used to express *variability* of products. Features can be modeled with mandatory, optional, and alternative constraints, as well as attributes (*extended feature models* [2]). Furthermore, relations between features can be expressed such as *exclude* or *require* which are all considered here as *integrity constraints* or simply *constraints* (see [11]). Thus, constraints relate to features. In this paper, we divide a feature model into features and constraints on one side to form the *consistency part* of the feature model and feature attributes on the other side to form the *optimization part* of the feature model. In the general form, feature attributes might also belong to the consistency part. Figure 1 presents an example also later used in the paper.

The task is now to select features from the feature model given some preferred features so that a valid configuration is created (feature selection) that fulfills the constraints. For simplicity, in this paper we consider the special case where the set of preferred features is empty. This can be considered as a constraint satisfaction problem, defined as follows [7]:

Definition (Constraint Satisfaction Problem – CSP). A constraint satisfaction problem (CSP) is defined by a triple (V, D, C) where V represents a set of finite domain variables $V = \{v_1, v_2, ..., v_n\}$, D represents variable domains $D = \{dom(v_1), dom(v_2), ..., dom(v_n)\}$, and C represents a set of constraints defining restrictions on the possible combinations of variable values ($C = \{c_1, c_2, ..., c_m\}$).

A *feature selection problem* is a CSP where variables represent features defined in a feature model.

A solution to a given CSP = (V, D, C) can be defined as follows:

Definition (CSP Solution). A solution for a given CSP = (V, D, C) is represented by an assignment $A = \{ins(v_1), ins(v_2), ..., ins(v_k)\}$ where $ins(v_i) \in dom(v_i)$. We require solutions to be *complete*, i.e., to be represented by an assignment where each variable in the definition of the CSP is instantiated and *consistent* which means that the assignment A is consistent with the constraints in C.

Thus, a *feature selection* (or *configuration*) is a CSP solution of a feature selection problem.

2.2 SAT-based constraint resolving

A propositional logic formula consists of binary variables and the operators AND, OR, and NOT. A truth value ($TRUE$ or $FALSE$) can be assigned to each binary variable. By assigning a truth value to each variable a formula can be satisfied, i.e., results to $TRUE$. A boolean satisfiability problem (SAT) is given by the task to check whether a given formula is satisfiable. As an extension, a further task is to assign values to not pre-assigned variables to make a formula satisfiable.

A feature selection problem can be mapped to a SAT problem by introducing binary variables for each feature and map constraints to formulas. A SAT solver, such as SAT4J [1] can be used to check the constraint violations of a solution or to compute valid variable assignments for not pre-assigned variables.

2.3 Multi-objective Evolutionary Optimization Algorithms (MOEAs)

In multi-objective optimization, multiple objective functions have to be optimized at the same time. We define a feature optimization problem as follows (see [5]): Compute $min(F_1(x), F_2(x), \ldots, F_k(x))$ with k the number of objective functions and $x \in X$ is the set of possible feature configurations. Each $F_i(x)$ is an objective function based on feature attributes. We introduce the boolean attribute "selected" to indicate if a feature is selected in the configuration or not. Each $F_i(x)$ has to be minimized. In evolutionary algorithms, different $F_i(x)$s are combined to a single value, the *fitness value*, to evaluate a feature configuration.

Let x_1 and x_2 be two potential solutions to the problem. We say that x_1 dominates x_2, if and only if $\forall i \in \{1, \ldots, k\} : F_i(x_1) \leq F_i(x_2)$ and $\exists i \in \{1, \ldots, k\} : F_i(x_1) < F_i(x_2)$. Given $x_1, \ldots, x_n$ potential solutions to the multi-objective optimization problem, the Pareto front corresponds to the subset of these potential solutions that are non-dominated by the others.

Each solution in the Pareto front is optimal in the sense that it cannot be improved in one objective function without degrading another one. Furthermore, all solutions in a Pareto front are equally optimal. However, solutions obtained by MOEAs are not exactly the Pareto front, but an approximation of it. Two properties are used to evaluate the quality of the obtained solutions: convergence and diversity. The first one describes how near they are to the Pareto front, while the second one indicates how uniformly they distribute. A good solution would have good convergence and diversity at the same time.

As pointed out in the introduction, indicator-based evolutionary algorithms (IBEA) provide a mean for solving multi-objective optimization problems (MOP). The rest of the paper will explain how those technologies are applied for computing optimal feature configurations.

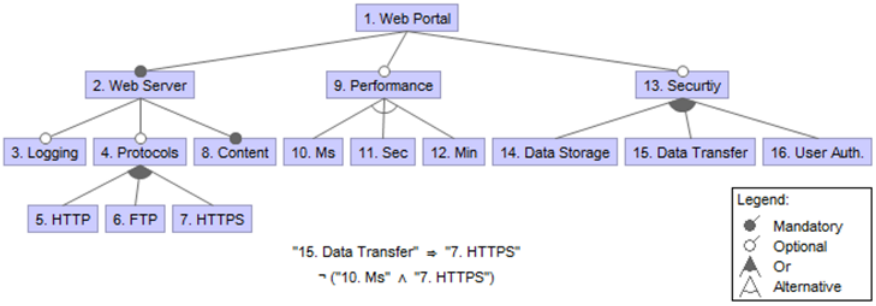
3 THE PROPOSED APPROACH - SATIBEA+

The main task of the automatic generation of configurations of SPLs considered here is to find a set of valid and optimal feature selections in consideration of multiple objectives. It means, at the end of the search process, the resulting configurations must be valid. In other words, the number of constraint violations should be zero in the final configurations. Although invalid configurations are permitted as intermediate results, minimization of the number of invalid constraints should be defined as an objective for the search process. Moreover, the more valid configurations at the end of the process are generated, the more useful is the search result, because more feasible configuration could be given to decision makers for the final selection.

In order to reduce the number of feature violations and increase the percentage of the valid configurations, the invalid configurations should be replaced by valid ones or at least by "better" ones (with less constraint violations) gradually during the search process. Our approach is based on the SATIBEA approach, so we name it "SATIBEA+". Similar to SATIBEA, we also use the SAT-solver to repair invalid configurations. In addition, SATIBEA+ can also change an invalid configuration into a valid or a "better" one by learning from an another configuration. In our approach, we extend the "smart" operators of "SATIBEA" to "smart+" operators to achieve this goal.

3.1 "Smart" and "Smart+" Operators

We introduce two "smart+" operators as an extension of "mboxsmart" operators of [5]. The operators are called "smart", because they



Feature	Cost	Defects	Used before
1. Web Portal	265.0	35	false
2. Web Server	50.0	5	false
3. Logging	19	2	true
...			
16. User Auth.	9.0	1	true

Figure 1. Example of a feature model including feature attributes based on [9]

can turn an invalid configuration into a full valid one or at least makes it better. Furthermore, they change a configuration only slightly (the change will be as little as possible).

3.1.1 Smart and Smart+ Mutation

This type of mutation operator is an unary operator acting on a single configuration aiming to change it from invalid to valid. Thus, the output of this operator should be a configuration without any constraint violation.

Smart Mutation of SATIBEA: Before mutation, constraint violations are calculated for a configuration. Then, features in this configuration are divided into two groups: the "bad" ones and the "good" ones. The former refers to the features, which are involved in at least one of the constraint violations. The latter refers to the features, which do not incur in any constraint violation. Then the feature assignments of bad ones will be removed while the assignments for good ones remain unchanged. After that, smart mutation will inquire the SAT-solver for a valid configuration under this assumption.

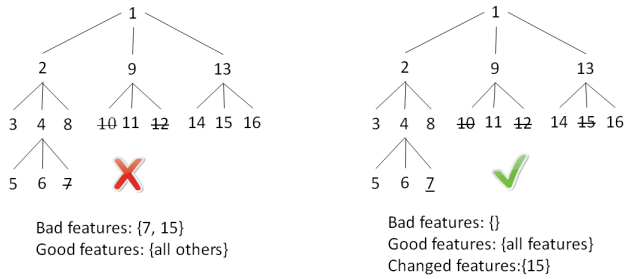
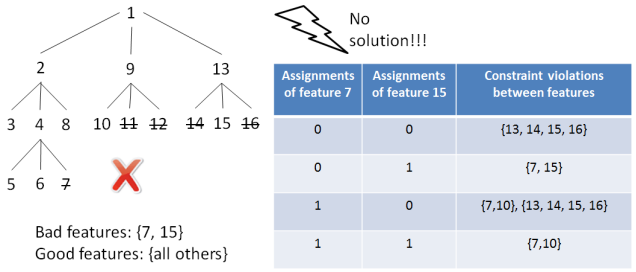


Figure 2. An positive example for the smart mutation in SATIBEA

Considering the configuration in the Figure 2 for the feature model defined in the Figure 1, it has a constraint violation of the *require* relation between 7 and 15 (the left side of Figure 2). In the following figures, the unselected features are struck out. If we remove the assignments of the features 7 and 15 and give the assignments of the rest features to the SAT-solver, then it will return a valid configuration without any violations (the right side). In this case, only the bad features will be changed (Feature 15). But what happens, if we apply this operator to repair the other invalid configuration shown in Figure 3 (the left side)? Like in Figure 2, the *exclude* relation between feature 7 and 15 is violated in this configuration.

According to all of the possible feature assignments of 7 and 15 (the right side), it is impossible to find a valid configuration. The more features and the more feature constraints a feature model has, the more likely no solution could be found by a SAT-solver. We have found that for the complex feature models (eCos, FreeBSD and Linux in Table 1) used in our experiment, the mutation operator of SATIBEA failed to find even just one valid configuration.



Assignments of feature 7	Assignments of feature 15	Constraint violations between features
0	0	{13, 14, 15, 16}
0	1	{7, 15}
1	0	{7,10}, {13, 14, 15, 16}
1	1	{7,10}

Figure 3. An negative example for the smart mutation in SATIBEA

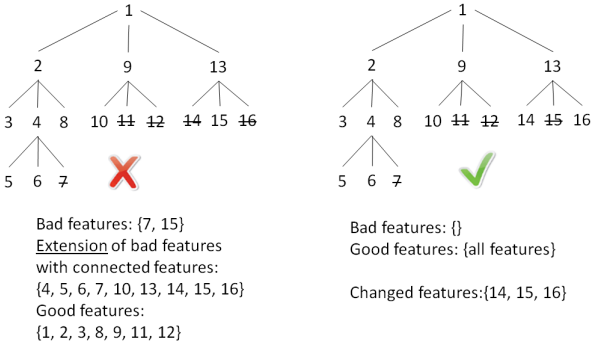


Figure 4. Using the new smart+ mutation to repair the configuration in the negative example

Smart+ mutation of SATIBEA+: In order to overcome this shortcoming, we introduce the concept of *connected features*. A feature x is "connected" with a feature y, if both features appear in the same feature constraint. For a feature x, we iterate all feature constraints and save all connected features with x in a set S. Then, we call S the *connected features* for the feature x. For example, the feature 7 has the connected features $\{4, 5, 6, 10, 15\}$ in the Figure 1, because there is at least one feature constraint defined between 7 and those. To search for an invalid configuration, we extended the set of the bad features with their connected features. It inquires the SAT-solver for a valid configuration while remaining the assignments of the good features (but without keeping the assignments of the connected features). Thus, we can now repair the invalid configuration in the Figure 3. As shown in Figure 4, not only the bad features, but also the their connected features, can be changed (features 14, 15 and 16 have been changed). With this extension, the failure rate of the repair operator for the Linux configuration model has been reduced up to 30% (see Section 6).

3.1.2 Smart+ Crossover

This operator is a binary operator acting on two configurations (called the parents in EA) aiming to reduce the constraint violations

of the first one. Thus, the output of this operator should be a new configuration (called the *offspring* in EA) with a reduced number of constraint violations. The crossover operator uses a crossover point which splits a configuration (a list of features) into two parts, *before* and *after* the point.

1-point Crossover of SATIBEA: This crossover operator combines two configurations in a new one by applying the feature assignments of the first one before the randomly selected crossover point and applying the feature assignments of the second one after the crossover point. Because the crossover point is selected randomly and feature values are simply copied without consideration of constraints, the number of constraint violations will be reduced also randomly. The more features and the more feature constraints a feature model has, the more likely no reduction of constraints could be achieved. We have found that for the complex feature models used in our experiment, the constraint violations has been even increased in most cases.

Algorithm 1 smart+ crossover

```
1:  Input: p1, p2
2:  offspring = p1.copy()
3:  violatedConstraintsP1 = getViolatedConstraints(p1);
4:  violatedConstraintsP2 = getViolatedConstraints(p2);
5:  for each constraint in violatedConstraintsP1 do
6:    if constraint not in violatedConstraintsP2 then
7:      for each featureX in constraint do
8:        valueOf(offspring, featureX) = valueOf(p2, featureX);
9:      end for
10:   end if
11: end for
12: return offspring
```

Smart+ crossover in SATIBEA+: Instead of generating an offspring by exchanging values of configurations arbitrarily, we correct feature violations in one configuration by learning "good" feature assignments from the other one. We introduce the smart+ crossover operator in Algorithm 1. The inputs are two configurations *p1* and *p2* as parents (line 1) and the output is the resulting configuration *offspring* (line 12). The algorithm begins with copying *p1* as the prototype of the *offspring* (line 2). Then it calculates the violated constraints of both parent configurations (line 3-4). For each violated constraint in *p1* (also in offspring, because it is the copy of *p1*), if we could find feature assignments in *p2*, which do not violate this constraint, then we use these the "good" feature assignments of *p2* to replace the "bad" ones in *offspring* (5-11). Please note that smart crossover is not aiming to resolve all invalid feature constraints, just as smart mutation does. It will only "improve" a configuration. It could generate an offspring without any invalid feature constraint, but not necessarily.

Suppose the configuration *Parent 1* in the Figure 5, which has 4 feature invalid constraints. Its counterpart *Parent 2* has only 2 feature constraints. Because the first three constraints of *Parent 1* are not violated in *Parent 2*, *Parent 1* could replace their problem features 4, 6, 7, 13, 14 with the "good" feature assignments of Parent 2. Thus, the generated *Offspring* has only one feature violation, which could not be fixed anyhow (because it is violated in both configurations). With this extension, the number of constraint violations for the Linux configuration model can be reduced in almost 70% of the cases (see Section 6).

3.2 Other changes against SATIBEA

3.2.1 Mating Selection

It is a selection operator which acts on a set of configurations aiming to generate parents for crossover. In evolutionary algorithms, solutions with better fitness values should have better opportunities to be selected as parents for the crossover.

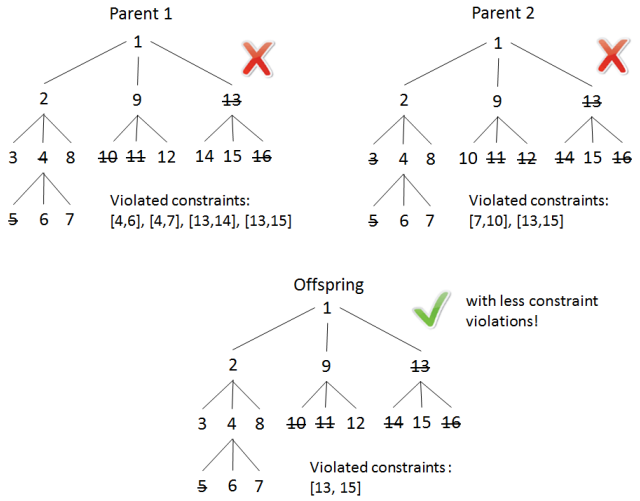


Figure 5. An example by using smart+ crossover to improve a configuration

Mutation selection of SATIBEA: Binary tournament strategy [14] has been applied in SATIBEA. From two randomly selected configurations, the better one (with the better fitness value) will be selected as the parent.

Changes in SATIBEA+: Instead of selecting parents by considering only fitness values, we make additional limitations on the selection process. Recalling the smart crossover defined in the last section, the "bad" configuration will get improved by learning from the "good" one. Thus, we limit a "good" configuration only on the ones without any constraint violations. It can be selected only from the set of valid configurations. In contrast, a "bad" configuration can be selected from any configurations, as in SATIBEA.

3.2.2 Smart Replacement

This operator is aiming to add new solutions randomly in the current population.

Smart Replacement of SATIBEA: In SATIBEA, it picks up a configuration from the current population randomly and replace it with a new valid configuration, which is also generated randomly.

Changes in SATIBEA+: Because this operator adds valid configurations into the population arbitrarily and periodically, it could produce uncontrolled influence on the final result so that effectiveness of smart operators could not be measured properly. In this paper, we investigated the influence of this operator.

3.3 The SATIBEA+ approach

The simplified activity diagram of SATIBEA+ can be seen in Figure 6. It augments SATIBEA with the new smart operators (steps 5-7). The other activities (steps 1-4 and 8) are described in detail in [12] and [5]. Like other MOEAs, it evolves a population of configurations from generation to generation (circles from 3 to 7 and back to 3) aiming to optimize given objectives. The difference is that features constraints are taken into account in the following activities:

- Steps 2 and 3: The number of constraint violations is considered as an extra objective to minimize. Its value is integrated in the calculation of fitness value by IBEA.
- Step 5: Smart mating selection selects a "good" parent (without constraint violations) and a "bad" parent (possibly with constraint violations) for crossover.

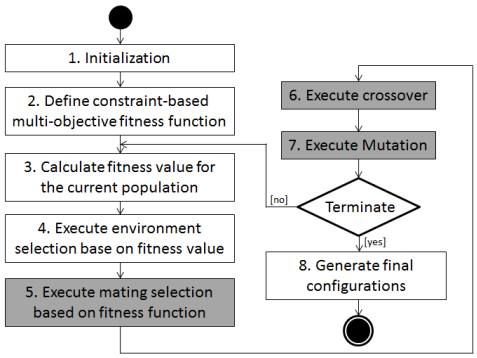


Figure 6. Activity diagram of SATIBEA+. Extensions compared to SATIBEA are marked in gray.

- Step 6: Smart+ crossover reduces the number of constraint violations in generating an offspring. It repairs the "bad" parent by learning from the "good" one.
- Step 7: Smart+ mutation eliminates constraint violations in changing an invalid configuration into a valid one.
- Step 8: Only the valid configurations will be considered as result, because configurations with any constraint violations are useless for the end user.

4 RESEARCH QUESTIONS

We conducted experiments to empirically compare the performance of SATIBEA+ with that of SATIBEA. Specially, we attempted to find answers for the following questions:

RQ1. How are the results found by SATIBEA+ compared to the results found by SATIBEA? Is the improvement repeatable on feature models with different sizes?

RQ2. How well does the new mutation operator affect the performance of search techniques?

RQ3. Is the new crossover operator actually more effective than the classic 1-point crossover used by SATIBEA?

RQ4. How much does the smart replacement operator affect the results?

RQ5. How does the algorithm implemented by SATIBEA+ work according to performance compared to SATIBEA?

5 EXPERIMENTAL DESIGN

5.1 Experimental Subjects

The study uses 4 feature models from the public feature model repositories SPLOT[2] and LVAT[3]. The characteristics of these feature models used in the experiment are summarized in Table 1 with the repository information (Repository), the name of the feature model (Model), the number of features (#Fea.), and the number of constraints (#Cons.).

Web Portal is a small demo feature model, which we are using to illustrate our approach (Figure 1). eCos and Linux X86 are examples of middle and big-sized feature models. They were reverse engineered by analyzing source codes, comments, and documentations of Linux kernel and eCos operation systems. Following the experiments used by [5, 9, 12, 15], each feature of used feature models is augmented by 3 attributes: *cost* ϵ $\mathbb{R}_{\geq 0}$, *defects* ϵ $\mathbb{Z}_{\geq 0}$ and *used_before* ϵ $Boolean$. The values for these attributes have been set arbitrarily in the feature model with an uniform distribution (*cost* in (5.0, 15.0), *defects* in (0, 10) and *used_before* in (true, false)). There is only one dependency which should be considered by the generation of attribute values. It can be formulated as: if (not *used_before*) then *defects*=0.

Repository	Model	#Fea.	#Cons.
SPLOT	Web Portal	16	26
LVAT	eCos	1244	3146
	FreeBSD	1369	62183
	Linux X86	6888	343944

Table 1. Feature models used in the experiment

5.2 The optimization problems

For the feature models introduced above, we are optimizing the following objectives formulated as minimization problems uniformly. They are calculated by the formula defined in Table 2.

- *Correctness*: Any constraint violation is not allowed in the final configurations. But as an objective in the optimization framework, we intend to minimize it.
- *Richness of features*: In a configuration, we want to have as many features selected as possible. It implies that the number of unselected features should be minimized.
- *Cost*: The total cost for a configuration should be minimized.
- *Defects*: The number of defects, which are caused by selected features, should be also minimized.
- *Feature used before*: In order to reduce the product risk, we are seeking to find the configurations that have minimized number of unused features.

Objective	Calculation formula for objectives
Correctness	$\sum Cons.(violated = \textbf{true})$
Richness of features	$\sum Fea.(selected = \textbf{false})$
Cost	$\sum Fea.(selected = \textbf{true}).cost$
Defects	$\sum Fea.(selected = \textbf{true}).defects$
Feature used before	$\sum Fea.(selected = \textbf{true} \& used\_before = \textbf{true})$

Table 2. Objectives used in the experiment

5.3 Implementation and Experimental Settings

1. Implementation:
   We used jMetal[3], an open-source Java framework for multiobjective optimization and SAT4j[1], a open-source library of SAT solvers to implement SATIBEA+. In addition, we compared SATIBEA+ with the original SATIBEA algorithm[4].
2. Parameter settings:
   All the experiments were performed on a computer with Quad Core@2.90 GHz CPU and 16 GB RAM, running on Windows 7. In order to compare with SATIBEA under fair conditions, we used exactly the same parameters for the evolutionary algorithm as the ones of [5]. Thus, SATIBEA+ differentiates SATIBEA only from EA-operators. Table 3 lists the used parameter settings and gives for each parameter a short description. To evaluate the influence of the smart replacement, we execute our experiment in two variations, namely with smart replacement and without smart replacement (last row of Table 3). In order to avoid the problem of genetic drift (diversity loss) described in [14], the smart crossover and smart mutation is only executed with low frequencies. Please

[2] Software Product Line Online Tools, `http://splot-research.org/`
[3] Linux Variability Analysis Tools, `http://code.google.com/p/linux-variability-analysis-tools/`
[4] Implementation of SATIBEA, `http://research.henard.net/SPL/ICSE_2015/`

note, the focus of this paper, was on comparing SATIBEA+ with SATIBEA, rather than tuning the parameters to achieve the best performance, which could be explored in future work.

Parameter	Explanation	Setting
Population size	Number of new configurations in the current population	300
Archive size	Number of configurations from the last population	300
Crossover probability	Probability that a crossover is executed	0.05
Probability for using standard mutation	Probability that a standard mutation is executed	0.98
Probability for bit flipping in the standard mutation	Probability that a changeable feature (not mandatory and dead feature) is flipped	0.001
Probability for using smart mutation	Probability that a smart mutation is executed	0.01
Probability for using smart replacement	Probability that a smart replacement is executed	0.01 or 0

Table 3. Parameter settings

5.4 Performance metrics

To evaluate the studied approach, we measure the calculated Pareto front in three directions: convergence, diversity, and computation time. Convergence metrics evaluate the effectiveness of the solutions in terms of their closeness to the optimal Pareto front, while diversity metrics measure the distribution of the solution set. Computation time is the length of time required to perform an algorithm, which represents its computational complexity. In our approach, they are represented by the following three indicators:

1. Hypervolume (HV)
   The hypervolume indicator, associated with a solution set S is given by the volume of the objective space portion that is weakly dominated by the set S [14]. It combines convergence and diversity measurement in a single indicator. In jMetal, all objectives are to be minimized and the Pareto front is inverted before the HV is calculated. Thus, the more HV value a solution set has, the more qualitative it is.
2. Pareto front size (PFS)
   Like HV, it is a combined indicator for the measurement of convergence and diversity. Although correctness is defined as a separate objective to be minimized in Table 2, there might be still some invalid configurations (with constraint violations) in the results. We use PFS to measure the number of unique and valid solutions in the obtained Pareto front. Because duplicated configurations are treated as a single one, it also gives a hint about the diversity of the solution set. A higher PFS value is preferred, because more valid configurations can be presented to the user.
3. Execution time (ET)
   This run-time indicator calculates the duration of the evolutionary algorithm for a given number of iterations. In each iteration, operations such as crossover and mutation will be executed. A higher ET value denotes a higher time complexity of an algorithm.

6 EXPERIMENTAL RESULTS

This section presents the results when applied to the 4 feature models. The performance metrics achieved by SATIBEA+ and by SATIBEA have been compared to each other. To investigate the contribution of the smart operators independently, three combinations are designed as followings:

1. The original SATIBEA with Smart Mutation: **the original approach**
2. SATIBEA+ with Smart Mutation+ but without Smart Crossover: **the filtered SATIEBA+ approach**
3. SATIBEA+ with Smart Mutation+ and with Smart Crossover: **the SATIBEA+ approach**

For each combination, we run the algorithm 30 times and for each run with a given number of objective evaluations. The number of objective evaluations equals the execution times of crossover and mutation in jMetal. Then we reported the medium values of the metrics.

6.1 With smart replacement vs. without smart replacement

As described above, the smart replacement will add a valid configuration to the population with a probability of one percent. If it had a strong influence on the result, then it would "flood" the contribution of the other smart operators. Thus, we executed our experiment with and without smart replacement separately. Each run is executed with 25000 objective evaluations (default value set by jMetal). The results are recorded in Table 4 and 5. When interpreting the results, we make the following observations:

Model	Performance indicators	SATIBEA	Filtered SATIBEA+	SATIBEA+
Web Portal	HV	0.066	0.067	0.067
	PFS	24	27	27
	ET (ms)	12909	12938	12980.3
eCos	HV	0.244	0.236	0.227
	PFS	82	203	215
	ET (ms)	16161	15925	15869
FreeBSD	HV	0.257	0.254	0.258
	PFS	41	47	148
	ET (ms)	64558	67647	67639
Linux X86	HV	0.237	0.241	0.238
	PFS	40	112	151
	ET (ms)	247084	256192	257110

Table 4. Evaluation results *with* smart replacement (*25000* objective evaluations)

Model	Performance indicators	SATIBEA	Filtered SATIBEA+	SATIBEA+
Web Portal	HV	0.069	0.066	0.069
	PFS	24	26	26
	ET (ms)	13073	13809	13385
eCos	HV	0	0.203	0.209
	PFS	0	196	190
	ET (ms)	16971	17832	16184
FreeBSD	HV	0	0.255	0.254
	PFS	0	55	148
	ET (ms)	70034	66574	71213
Linux X86	HV	0.010	0.245	0.243
	PFS	0	138	177
	ET (ms)	224468	264858	270617

Table 5. Evaluation results *without* smart replacement (*25000* objective evaluations)

Answering RQ4 (compare Table 4 and Table 5): The result of the experiment is "glamorized" with smart replacement. Although

SATIBEA was not able to find any valid solutions for complex FMs (eCos, FreeBSD and Linux) without using smart replacement (Table 5), a couple of valid configurations could be found by using it (Table 4). A main reason is that more conflicting features can be adapted by the SAT-solver in consideration of connected features. In addition, no big difference in terms of HV can be seen by using smart replacement (Table 4). It is obvious that the result has been drastically affected by using smart replacement, because this operation adds valid solutions to the population periodically. Thus, we treated the result in Table 4 as "invalid" and only used the results of Table 5 for further analysis.

Answering RQ1 (compare column 3 with 5 in Table 5): SATIBEA+ outperforms SATIBEA in terms of HV and PFS significantly, particularly for the middle-complex (eCos and FreeBSD) and high-complex FMs (Linux X86). For the simple FM (Web Portal), the difference is less notable. In addition, it is notable that for the complex FMs (eCos, FreeBSD and Linux), no valid configurations could be found by using SATIBEA. In contrast, SATIBEA+ could find many valid configurations for them.

Answering RQ2 (compare column 3 with 4 in Table 5): Filtered SATIBEA+ outperforms SATIBEA in terms of HV and PFS significantly, particularly for the middle-complex and high-complex FMs. For the simple FM, the difference is less notable. Fitlered SATIBEA+ found also valid configurations for the complex FMs by using the enhanced smart mutation.

Answering RQ3 (compare column 4 with 5 in Table 5): SATIBEA+ outperforms filtered SATIBEA+ in terms of PFS significantly, particularly for the middle-complex and high-complex FM (except for the FM of FreeBSD). For the simple FM, the difference is less notable. In addition, there is no remarkable performance improvement in terms of HV by using smart crossover.

Answering RQ5 (compare ETs in Table 5): For each feature model, there is no significant differences in terms of ET. The execution time of all algorithms are in a comparable range.

Other findings: For the simple feature model, there is no notable differences between SATIBEA, filtered SATIBEA+ and SATIBEA+ for all performance indicators. Thus, this feature model was not considered in further experiments.

6.2 Further Runs

In order to analyze the development of performance metrics by increased objective evaluations (also by increased execution times), we performed further runs with 12500 and 50000 objective evaluations on the 4 FMs separately. Because smart replacement changed the results too radically, it was not applied by the further executions. Considering the results in Table 5 and Table 6, we make the following observations:

- PFSs get improved significantly by the increased times of objective evaluations. One exception is the feature model FreeBSD. It generates less valid solutions with more objective evaluations.
- HVs also get improved by the increased times of objective evaluations, but slightly.
- ETs are proportional to the times of objective evaluations.
- For the feature model eCos, SATIBEA+ performs a little bit worse in terms of PFS than filtered SATIBEA+.

7 DISCUSSION

Comparison of SATIBEA+, filtered SATIBEA+, and SATIBEA In this section, we reason about our findings and discuss their implications. First of all, we ask why SATIBEA+ performs much better than SATIBEA? The performance improvement is essentially achieved by the smart+ operators used by SATIBEA+. Firstly, the smart+ mutation beats the native smart mutation in repairing an invalid configuration. The reason for that is that we "relax" the scope of the mutation. In SATIBEA+, a feature and its related (by constraints) features will be considered as a whole mutable-able unit. Thus, the SAT-solver searches for a valid solution in an expended range. Secondly, the smart+ crossover repairs an invalid configuration much better than the standard 1-point crossover, because it follows the motto "learn from the best". In SATIBEA+, a configuration will get repaired (at least partly) by learning "good" feature assignments from a valid configuration. Thus, the number of feature violations of a configuration will be reduced and finally more valid configurations will be produced. To sum up, we embed the specific problem information (feature configuration) into an evolutionary algorithm, together with a problem-dependent local search method (with smart+ operators); therefore, the performance of the algorithm will be improved according to the No Free Lunch theorem [18].

Model	Objective evaluations	Performance indicators	SATIBEA	Filtered SATIBEA+	SATIBEA+
eCos	12500	HV	0	0.200	0.199
		PFS	0	158	141
		ET (ms)	8113	8393	8358
	50000	HV	0	0.231	0.194
		PFS	0	228	209
		ET (ms)	32281	31768	31894
FreeBSD	12500	HV	0	0.248	0.248
		PFS	0	83	135
		ET (ms)	34887	36768	41682
	50000	HV	0	0.260	0.267
		PFS	0	28	135
		ET (ms)	132204	134914	134129
Linux X86	12500	HV	0	0.242	0.241
		PFS	0	122	126
		ET (ms)	116686	144918	145726
	50000	HV	0	0.247	0.244
		PFS	0	138	183
		ET (ms)	457070	492015	451761

Table 6. Evaluation results *without* smart replacement (*12500* and *50000* objective evaluations)

The results of experiments reveals also some abnormality. One is for the feature model eCos, less valid configurations could be found by SATIBEA+ than by filtered SATIBEA+. The other is for the feature model FreeBSD, the performance indicator PFS could not get converged with increased number of runs. It indicates that the performance is sensible to the characteristics of a feature model (such as the number of cross-tree constraints). We are planing to investigate this phenomena in the future work.

Threats to Validity A prerequisite for applying optimization technologies, such as SATIBEA+ is that the objectives have to be calculated from the feature attributes. For attributes such as *cost* this is given by the trivial sum, however, other attributes might be more complex to evaluate. The test set for the Web-Portal is a theoretical one, i.e., the attributes and their values are set randomly, however, this will not influence our experimental results. Moreover, the execution time of our experiments are limited to 10 minutes, because SATIBEA+ could already outperform the original algorithm SATIBEA during this short time.

More interesting is the choice of the connected features (see Sec-

tion 3). In our case, we only select the directly connected features (i.e., depth 1). Hence, in very complex domains it might be the case that no solution will be found. In future work, we will investigate in features connected indirectly through multiple constraints.

8 RELATED WORK

A key challenge in the Software Product Line community is to determine how to select a set of features from a feature model, which not only satisfies feature constraints but also optimize the different objectives of customers. [17] uses Filtered Cartesian Flattening to calculate the optimal feature sets subject to the given resource constraints. They transform a feature selection problem as a Multi-Dimensional, Multiple-Choice (MDMC) knapsack problem and introduce a heuristic to filter choices.

Using the task planning technique HTN (Hierarchical Task Network), [13] proposes a framework to select suitable features that satisfy both the stakeholders' functional requirements (FRs) and non-functional requirements (NFRs). For optimization, they aggregate qualitative and quantitative properties into a single object value. [4] presents a genetic algorithm GAFES to optimize feature selections in the face of resource constraints. In the fitness function, they use weighting to reflect the different importance of resources. However, it is not trivial to find a proper utility function to change a multi-criteria problem to a single-criterion problem. Moreover, only one "best" solution will be calculated, while a set of solutions (Pareto set) is expected for multi-criteria problems.

[12] shows that Indicator-Based Evolutionary Algorithm (IBEA) works better than other dominance-based EAs in solving feature selection problems for large models and many optimization objectives. [9] propose a multi-objective evolutionary algorithm IVEA to optimize the selection of features with FRs and NFRs. They treat constraint violations in a feature model as a separated dimension besides the optimization of NFRs. They define a violation-dominance function to guide the environment selection and mating selection. However, only the standard crossover and mutation operators are applied in these approaches and therefore the performance improvements are limited.

[15] incorporate feedback-directed mechanism into the EA for multi-objective feature selection problem. Similar as the smart crossover of SATIBEA+, an invalid configuration gets repaired by copying all of the non-error features from an another configuration. The difference is that our approach is mini-invasive which copies only a minimal set of feature assignments (only for "bad" features). For configuration optimization, [10] propose a two tasks approach. Firstly, a rough approximation of Pareto front is searched and presented to the user. Then the user indicates the areas which he interested in. In [6], optimal products are chosen from a feature model by using SIP, a two steps multi-objective evolutionary algorithm. They concentrate primarly on the number of constraints that hold and then on the other objectives.

9 CONCLUSION AND FUTURE WORK

In this work, we have demonstrated that our approach SATIBEA+, for the multi-objective feature selection problem, outperforms the state-of-the-art algorithm SATIBEA [5] in the quality and amount of found configurations. In addition, we have also show that our approach scales to the complex feature models with thousands of features and constraints. Still, the question may arise regarding the effectiveness of SATIBEA+ in consideration of other characteristics of feature models (such as number of cross-tree constraints, which may be the subject of further investigation). Other directions for future work regarding optimal feature selection problems could be:

1. Comparison of performance of constraint-based approaches and constraint-first approaches [6].
2. Incorporate customer requirements during the search in all phases of the optimization process.
3. Further experiments with complex feature models with real attribute values.

REFERENCES

[1] Daniel Le Berre and Anne Parrain, 'The sat4j library, release 2.2, system description', *Journal on Satisfiability, Boolean Modeling and Computation*, **7**(2010), 59–64, (2010).

[2] K. Czarnecki, S. Helsen, and U. Eisenecker, 'Formalizing Cardinality-based Feature Models and their Specialization', *Software Process: Improvement and Practice*, **10**(1), 7–29, (2005).

[3] Juan J Durillo and Antonio J Nebro, 'jMetal: A Java framework for multi-objective optimization', *Advances in Engineering Software*, **42**(10), 760–771, (October 2011).

[4] Jianmei Guo, Jules White, Guangxin Wang, Jian Li, and Yinglin Wang, 'A genetic algorithm for optimized feature selection with resource constraints in software product lines', *Journal of Systems and Software*, **84**(12), 2208–2221, (December 2011).

[5] Christopher Henard and Mike Papadakis, 'Combining multi-objective search and constraint solving for configuring large software product lines', in *ICSE' 15 Proceedings of the 37th International Conference on Software Engineering*, pp. 517–528, (2015).

[6] Robert M Hierons, Miqing Li, Xiaohui Liu, Sergio Segura, and Wei Zheng, 'SIP: Optimal Product Selection from Feature Models Using Many-Objective Evolutionary Optimization', *ACM Transactions on Software Engineering and Methodology*, **25**(2), 1–39, (April 2016).

[7] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, and K. Wolter, 'Configuration Knowledge Representation & Reasoning', in *Knowledge-based Configuration – From Research to Business Cases*, eds., A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, chapter 6, 59–96, Morgan Kaufmann Publishers, (2013).

[8] K C Kang, J Lee, and P Donohoe, 'Feature-Oriented Product Line Engineering', *IEEE Software*, **19**(4), 58–65, (2002).

[9] Xiaoli Lian and Li Zhang, 'Optimized feature selection towards functional and non-functional requirements in Software Product Lines', in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 191–200. IEEE, (March 2015).

[10] Paul Pitiot, Michel Aldanondo, Elise Vareilles, Thierry Coudert, and Paul Gaborit, 'Improving configuration and planning optimization: Towards a two tasks approach ', in 15^{th} *International Configuration Workshop*, eds., M. Aldanondo and A. Falkner, pp. 35–40, Vienna, Austria, (2013).

[11] *Handbook of Constraint Programming*, eds., F. Rossi, P. van Beek, and T. Walsh, Elsevier, 2006.

[12] AS Sayyad, T Menzies, and H Ammar, 'On the value of user preferences in search-based software engineering: a case study in software product lines', in *ICSE '13 Proceedings of the 2013 International Conference on Software Engineering*, pp. 492–501, (2013).

[13] Samaneh Soltani, Mohsen Asadi, and D Gašević, 'Automated planning for feature model configuration based on functional and non-functional requirements', in *SPLC'12*, pp. 56–65, (2012).

[14] El-Ghazali Talbi, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, Inc., Hoboken, NJ, USA, June 2009.

[15] Tian Huat Tan, Yinxing Xue, Manman Chen, Jun Sun, Yang Liu, and Jin Song Dong, 'Optimizing selection of competing features via feedback-directed evolutionary algorithms', in *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, pp. 246–256, New York, New York, USA, (2015). ACM Press.

[16] Edward Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, San Diego, New York, 1993.

[17] Jules White, Brian Dougherty, and Douglas C. Schmidt, 'Selecting highly optimal architectural feature sets with Filtered Cartesian Flattening', *Journal of Systems and Software*, **82**(8), 1268–1284, (August 2009).

[18] Xinjie Yu and Mitsuo Gen, *Introduction to Evolutionary Algorithms*, Decision Engineering, Springer London, London, 2010.

Interactive Configuration of Insulating Envelopes

Andrés F. Barco and **Philippe Chantry** and **Élise Vareilles** and **Michel Aldanondo**[1]

Abstract. This paper discusses how the functional programming paradigm may support interactive product configuration. We do so by describing our experience on the construction of an interactive product configuration software dedicated to the configuration of façades external insulating envelopes. The solution provides visual feedback to the user configuration actions thus assisting his/her decision-making in real-time.

1 Introduction

In the aim of reducing energy consumption, buildings may be externally retrofitted by covering them with insulating envelopes [1]. These insulating envelopes are based on the technical concept of rectangular panels: Wood-made insulating structures configurable in the sense that their size and their position over the façade must be defined prior to their manufacturing and shipment. An insulating envelope is a set of configured panels that respects the geometry and structure of the façade. The configuration is subject to the following constraints:

1. Panels size (width and height) is restricted to a given interval (manufacture, transportation and installation conditions),
2. partially overlapping windows or doors is forbidden (manufacture conditions),
3. panels overlapping is not allowed (insulation conditions),
4. insulating envelopes cannot contain holes (insulation conditions),
5. panels are attached by their corners in specific areas of the façade (installation conditions).

The envelopes configuration presents a challenge to academics and practitioners given the differences in geometry and structure of each building. In regard of these conditions, one of the key problems of this retrofit is to propose computational processes to support the manual and automatic configuration of panels, consequently envelopes, with respect to each façade to be retrofitted.

The authors have been working in the envelopes configuration problem since 2013. Previous reports in the *International Configuration Workshop* discussed the goals of the research [14] (2013), a first greedy solution [2] (2014) and an on-line constraint-based support system for assisting this configuration [3] (2015). The proposed solutions, however, do not allow manual configuration of envelopes but rather automatic ones. In these setups, the user may adapt any solution by removing and redefining panels (mainly to improve their aesthetics) only at the end of the configuration process. Further, the aesthetic flair, such as symmetric solutions, has not been taken into account, leaving this critical architectural requirement overlooked.

This paper presents a manual interactive configuration of insulating envelopes. The solution applies a constructive approach [10] allowing the user to visualize in real-time the impact of her/his own panels configuration in regard to the industrial conditions. We do so by using the methodology of divide and conquer implemented under the functional programming paradigm. The solution may be implemented in any functional language without relying on complex black-box tools as constraint solvers, linear programming libraries or meta-heuristics. Further, we propose a web-oriented Java-script implementation that gives the possibility to have a real-time interaction with the user by evading potential network traffic and delays (see concept in Figure 1). What is more, partial envelopes configurations may be finished by automated algorithms in a web-service setup (as presented in [3]). Lastly, we show that interactive configuration improves the transparency of the configuration process and generates subjective-oriented envelopes.

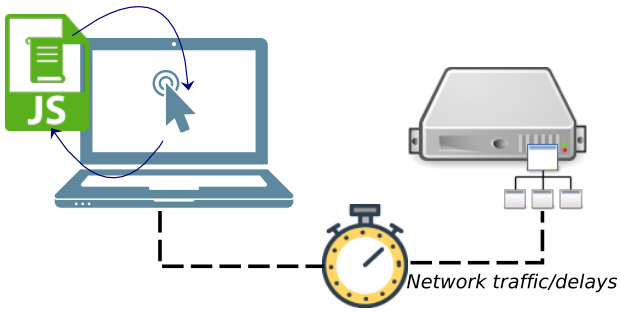


Figure 1. Real-time manual interactive configuration.

The document is structured as follows. In Section 2, details about the configuration problem are given. In Section 3, the interactive configuration scheme is introduced. In Section 4, the benefits for using functional programming over other paradigms is discussed. In Section 5, the functionality division is presented. Finally, in Section 6, a demonstration of the solution and conclusions are drawn.

2 Envelopes Configuration

The configuration problem here addressed raises when retrofitting buildings to reduce their energy consumption levels. The retrofit is done by attaching an insulating envelope composed of rectangular insulating panels over each façade. Façades have a size, are of rectangular shape and are composed of rectangular frames (windows and doors) and rectangular supporting areas (to attach panels' corners). Panels are rigid 2D rectilinear rectangles. Therefore their sides are parallel to the façade reference axis. All the panels covering a façade

[1] Université de Toulouse, Mines d'Albi, Campus Jarlard, 81013 Albi Cedex 09, France, email: abarcosa@mines-albi.fr

belong to an unique vertical plane with origin $(0, 0)$ at the bottom-left corner of the façade. Panels are configured by setting their size and setting the position of the bottom-left corner. They have a given orientation: If the panel's height is bigger than its width, the panel is vertical, otherwise it is horizontal. This information impacts the inner structure of the panel and its laying direction onto the façade. Due to the industrial conditions of the retrofit, panels have specific size lower and upper bounds that must be respected when instantiating them. Plausibly, configuring panels using their size upper bound leads to envelopes with a small number of panels. Their position must be accurately set in such a way that all frames over the façade are covered with the condition that no partial overlapping is allowed. Also, panels in an insulating envelope must avoid holes and overlapping in order to guarantee the complete insulation. According to the above description, the arguments and main decision variables refer to:

- Arguments: Façade size (width fac_w and height fac_h), for each frame (window/door) its position of bottom-left corner (fr_x, fr_y) and its size (width fr_w and height fr_h), for each supporting area its position of bottom-left corner (sa_x, sa_y) and its size (width sa_w and height sa_h), width bounds $[min_w, max_w]$ and height bounds $[min_h, max_h]$.
- Decision variables: Panels' width $p_w \in [min_w, max_w]$, length $p_h \in [min_h, max_h]$, coordinates $p_x \in [0, fac_w]$ and $p_y \in [0, fac_h]$ of bottom-left corner of the panel and its orientation p_{or}.

The industrial retrofit of buildings has to cope with a multiple and diverse requirements, guidelines and constraints coming from urban design guidelines, owners' expectations, tenders' wishes and architects' skills and ability in design art. Within these, two key functional requirement have been identified: a) been able to do manual and (semi) automatic configuration of panels and b) have a web-oriented support system. In the next section we discuss how the manual interactive configuration is conceived for assisting architects decision-making.

3 Scheme for Interactive Configuration

An alternative for manual configuration is to provide instantaneous feedback to the architect actions. This means that the configuration may be *interactively guided* by the support system. For when drawing a panel the system may restrict a) its possible size to be smaller than the upper bound and bigger than the lower bound, b) visually inform of conflicts with frames and supporting areas, and c) completely avoid panels overlapping. An interactive manual configuration of panels would work as follows:

1. The system presents a two-dimensional drawing of the façade.
2. The architect draws a panel while the system informs:
   - If the size of the panel is too small or too big given panel bounds,
   - if the panel is entering in conflict with windows and/or doors,
   - if the panel cannot be installed because its corners cannot be attached,
   - if the panel is entering in overlapping conflict with an already configured panel,
   - if the panel will block the configuration of further panels.
3. The architects iterates step 2 until satisfaction.

As a matter of choice, and if the graphical user interface (GUI) capabilities allows it, a given drawn panel may be re-configured by the architect as part of aesthetics considerations or because, as far the architect can tell, current constraint conflicts may get solved. This means that in manual configuration, ill definition of panels should be possible. Further, stopping ill definition of panels may be counter-productive for the architects aesthetic flair. The underlying validation algorithms inform the architect which constraints are violated but at the end the architect decides the exact size and position of each panel. Among the set of alternatives to support architects manual interactive configuration we have chosen the following:

- Informing about constraint conflicts is done visually: Our design choice is to set different colors for well-defined panels (green) and ill-defined ones (red). Informing about these conflicts is done in real-time.
- As an invariant, for configuring a new panel, each of the previously drawn panels must be well configured.
- Re-configuration of a given panel is possible. Colors of the panel are changed interactively depending on constraint conflicts.
- Gaps between panels are not conflictive for the result. In other words, when doing manual interactive configuration covering the entire area surface is not mandatory (holes are irrelevant).

According to the established decisions, the support system has two main responsibilities. Firstly, knowing that constraint conflicts exist for the selected/drawn panel. Secondly, it must inform the user about those conflicts. The former responsibility is fulfilled by the validation algorithms described in Section 5. The latter responsibility is fulfilled with the GUI capabilities[2]

4 Advantages of the functional programming over other paradigms

As explained before, our efforts focus on providing to the architects an interactive configuration of insulating envelopes in real-time. An interactive configuration refers to the system reactions to the user actions in order to help him/her to reach her/his (configuration) goals [4]. Interactive behavior has been widely studied in many knowledge areas and industry sectors [7, 8, 9, 13]. Among other things, the human interface allowing the interactive communication is one of the major study topics in computer science and informatics [4]. In regard to interactive product configuration, operational research (OR) techniques, such as constraint satisfaction, have proven their robustness and capabilities [5, 6, 12].

On the other hand we have real-time support. Real-time support refers to the capabilities of the support system to react to the user actions in "no time". Real-time interaction is needed, mostly, when the user actions require a response within the next 100 milliseconds (cf. Chapter 17 in [4]). For instance, for activities involving hand-eye coordination, the system answer must be fast enough as to not block the activity or deteriorate the results. In the envelopes configuration case, immediate support must be given to architects when configuring each panel. This means that the underlying support system must execute validation or resolution algorithms in such a way that the configuration process is continuously fed by the system responses.

At the beginning of our research, we have considered that constraint satisfaction techniques were appropriated for addressing the envelopes configuration problem. Our results have shown us right

[2] No discussion about the GUI capabilities are provided as they are of marginal interest to our work.

when assisting the automatic and (semi) automatic configuration of envelopes. Further, the framework of constraint satisfaction is known to address support interactive configuration. Nevertheless, for providing a real-time feedback we have relied on the simple yet powerful concept of task division instead of OR and AI techniques. For when using the OR and AI tools, such as filtering engines and constraint solvers, the time expend in having an output may exceed the real-time requirements. Further, due to the functional requirement of having a web-oriented solution, the communication with the underlying constraint services (or linear solver, genetic library, etc) involves a considerable time even when implemented with low latency protocols as AJAX [11]. In consequence, we have chosen to rely in the most known paradigm of computer science; functional programming.

The main concept from functional programming is divide and conquer. This concept is implemented by means of functions that are assembled together to provide a major functionality. The manual configuration of envelopes is addressed under the functional programming paradigm, meaning that the task to solve constraint conflicts has been delegated to different functions. In this case, each constraint is linked to a unique function.

5 Entrusting Responsibilities

In this section we present how functional programming is used to implement validation algorithms for the constraints presented in the problem. We give a brief behavior description for every function. The return value for each function is **true** if the panel is well configured and **false** otherwise. Examples of each function result are presented in the next section as screen-shots.

1. Size constraint. The size constraint states limits for the width and height of panels. The size constraint is implemented with respect to the panels orientation and lower and upper bounds. Recall that if the ratio $\frac{p_w}{p_h}$ is less than one, the panel orientation is vertical, otherwise, horizontal. Taking into account this information, the current panel width lies between p_{wl} and p_{wu} whereas its height must lie between p_{hl} and p_{hu}, for horizontally oriented panels. Conversely, for vertically oriented panels a swap between width and height bounds is executed. This function is executed when decreasing or increasing the size of panel by means of the graphical interface.
2. Non-overlapping constraint. The non-overlapping constraint states that panels cannot overlap in at least one dimension. It uses as input the position and size of the panel currently being configured, and a list of already-defined panels (adp). To verify panels overlapping, the current configured panel must be checked against every already-defined panel. An overlapping exists between panels p and q, if the projection of their widths and heights overlaps. If an overlapping exists, the color of the current configured panel is changed to red while the already defined panel remains green. This function is executed in two cases. First, when decreasing or increasing the size of panel by means of the graphical interface. Second, when the defined panel is moved (re-configured) around the façade surface.
3. Frames and interference constraint. The function checks whether an initial origin point and size of a given panel, bottom-left corner (p_{x0}, p_{y0}) and width (p_w) and height (p_h), violate frames constraint. It uses a stack to perform an ordered check of all frames partially or completely overlapped by the panel. In the case there is a conflict, the algorithm terminates informing that a conflict exists. This function is executed every time a new panel is drawn.
4. Installation constraint. The last function ties up a single panel definition and implements the validation of the installation constraint. Recall that in order to attach panels, their corners must be matched with supporting areas (sa). Also, supporting areas at the bottom corners must be strong enough to support half the weight of the panel. In essence, the function checks that every corner matches a supporting area. To do so, it tests corners against every supporting area, increasing a counter each time a corner is well located. The panels' weight and the supporting area load bearing capabilities are only checked for the bottom corners. If at least one corner is out of supporting areas then the panel is not well configured. It is worth noticing that supporting areas do not overlap. This means that a given corner can match at most one supporting area.

6 Conclusions

This short paper has discussed the manual configuration of insulating envelopes for façades. We have shown first the industrial conditions and details surrounding the configuration problem. In a second step we have discussed how the manual configuration is made interactive with a visual communication with the user (architect). Then, we have argued that the functional programming paradigm may be better suited to address the interactive configuration in comparison with other paradigms and technologies. Further, we have discussed the fact that functional programming allows the configuration support to be executed less than 100 milliseconds thus providing real-time interaction with users. Finally, we have briefly presented four of the key functions to support the configuration problem. The authors consider important to highlight that the simple and well-known methodology of divide and conquer may be used for teaching purposes in design related fields (e.g. architecture, industrial design, applied arts). Further, configuration, design and even manufacturing concepts may be used in introductory courses of computer programming given real ties with current industry problems.

To conclude the paper, interface and configuration process of our support system over a real-life France façade are illustrated in Figure 2.

REFERENCES

[1] M. Aldanondo, A.F. Barco, E. Vareilles, M. Falcon, P. Gaborit, and L. Zhang, 'Towards a bim approach for a high performance renovation of apartment buildings', in *Product Lifecycle Management for a Global Market*, eds., Shuichi Fukuda, Alain Bernard, Balan Gurumoorthy, and Abdelaziz Bouras, volume 442 of *IFIP Advances in Information and Communication Technology*, 21–30, Springer Berlin Heidelberg, (2014).

[2] A.F. Barco, E. Vareilles, M. Aldanondo, and P. Gaborit, 'Calpinator: A configuration tool for building facades', in *Proceedings of the 16th International Configuration Workshop, Novi Sad, Serbia, September 25-26, 2014.*, eds., A. Felfernig, C. Forza, and A. Haag, volume 1220 of *CEUR Workshop Proceedings*, pp. 47–54. CEUR-WS.org, (2014).

[3] A.F. Barco, E. Vareilles, M. Aldanondo, P. Gaborit, and J.G. Fages, 'Coupling two constraint-based systems into an on-line facade-layout configurator', in *Proceedings of the 17th International Configuration Workshop, Vienna, Austria, September 10-11, 2015.*, eds., Juha Tiihonen, Andreas Falkner, and Tomas Axling, pp. 47–54, (2015).

[4] A. Dix, J.E. Finlay, G.D. Abowd, and R. Beale, *Human-Computer Interaction (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[5] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1 edn., 2014.

[6] E. Gelle and R. Weigel, *Knowledge Intensive CAD: Volume 1*, chapter Interactive Configuration based on Incremental Constraint Satisfaction, 127–136, Springer US, Boston, MA, 1996.

Figure 2. Demonstration: System actions through normal execution.

[7] D. Glover, D. Miller, D. Averis, and V. Door, 'The interactive whiteboard: a literature survey', *Technology, Pedagogy and Education*, **14**(2), 155–170, (2005).

[8] V.B. Godin, 'Interactive scheduling: Historical survey and state of the art', *A I I E Transactions*, **10**(3), 331–337, (1978).

[9] J. Jankowski and M. Hachet, 'A Survey of Interaction Techniques for Interactive 3D Environments', in *Eurographics 2013 - STAR*, Girona, Spain, (May 2013).

[10] R. S. Liggett, 'Automated facilities layout: past, present and future', *Automation in Construction*, **9**(2), pp. 197 – 215, (2000).

[11] L. D. Paulson, 'Building rich web applications with ajax', *Computer*, **38**(10), 14–17, (Oct 2005).

[12] D. Schneeweiss and P. Hofstedt, *Applications of Declarative Programming and Knowledge Management: 19th International Conference, INAP 2011, and 25th Workshop on Logic Programming, WLP 2011, Vienna, Austria, September 28-30, 2011, Revised Selected Papers*, chapter FdConfig: A Constraint-Based Interactive Product Configurator, 239–255, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[13] W.S. Shin and A. Ravindran, 'Interactive multiple objective optimization: Survey icontinuous case', *Computers & Operations Research*, **18**(1), 97 – 114, (1991).

[14] E. Vareilles, C. Thuesen, M. Falcon, and M. Aldanondo, 'Interactive configuration of high performance renovation of apartment buildings by the use of CSP', in *Proceedings of the 15th International Configuration Workshop, Vienna, Austria, August 29-30, 2013.*, eds., Michel Aldanondo and Andreas A. Falkner, volume 1128 of *CEUR Workshop Proceedings*, pp. 29–34. CEUR-WS.org, (2013).

StudyBattles: A Learning Environment for Knowledge-based Configuration

Alexander Felfernig[1] and **Amal Shehadeh**[1] and **Michael Jeran**[1] and **Christian Gütl**[2] and **Trang Tran**[1] and **Müslüm Atas**[1] and **Seda Polat Erdeniz**[1] and **Martin Stettinger**[1] and **Arda Akcay**[1] and **Stefan Reiterer**[3]

Abstract. E-learning is a complementary channel for learners to acquire relevant knowledge. On the basis of an example e-learning environment (StudyBattles) we show how configuration-related knowledge can be transferred to end-users (e.g., sales representatives) as well as to knowledge engineers. In addition, we discuss an approach to automatically generate product domain as well as engineering learning content to be used in StudyBattles. Finally, we report the results of an initial qualitative study on the applicability of StudyBattles.

1 Introduction

Knowledge-based configuration is one of the most successfully applied Artificial Intelligence technologies [1, 4, 9, 15]. Configuration systems improve business processes in various dimensions such as *reduced error rates* and *time efforts* in product offering and *reduced costs* of error management in follow-up production processes. Despite the successful application of configuration technologies, there are still issues related to the transfer of configuration related knowledge to employees.

In dialogs with customers, sales representatives should not only rely on solutions and related explanations provided by the configuration environment but should also have the needed domain knowledge. Furthermore, engineers and domain experts engaged in knowledge engineering processes should have the needed technical foundations and be aware of engineering practices to minimize overheads in knowledge engineering processes. Finally, domain experts in charge of documenting configuration knowledge should be aware of standards on how to document knowledge in such a way that knowledge engineers can formalize this knowledge easily. The goal of this paper is to show how e-learning technologies [16] can be applied as a means (in addition to traditional training programs such as sales force training or trainings related to knowledge acquisition and maintenance) to support the mentioned knowledge transfer.

E-learning systems are often applied for creating a *corporate memory* that is exploited to improve process-relevant knowledge of employees (e.g., sales, marketing, and product management). Improvements triggered by the application of e-learning technologies are manyfold. They reach from the *increased accessibility* of learning contents (users are much more flexible with regard to the time of learning and training), *increased opportunities to analyze the strengths and weaknesses of employees with regard to organization-relevant knowledge*, and *increased consumption frequency of learning content* due to the application of different types of motivation mechanisms (e.g., gamification and persuasion [7]).

In this paper we focus on two configuration-related types of knowledge. *First*, we show how *sales-relevant configuration knowledge* can be represented in an e-learning environment. Examples of such knowledge types are *product knowledge* (e.g., for a specific set of customer requirements, which configurations should be recommended) and *analysis knowledge* (e.g., if no solution (configuration) can be identified for a given set of customer requirements, which alternatives should be proposed to the customer in order to maximize the probability that the customer will accept the offer).

Second, especially less experienced knowledge engineers and domain experts should be educated with regard to *best practices in knowledge acquisition and maintenance*. Examples of such knowledge types are *documentation knowledge* (e.g., in which way should incompatibilities between components types be documented on a textual level) and *knowledge representation knowledge* (e.g., in which context one should use compatibilities or incompatibilities to express allowed combinations of component types).

The existing demand for complementary means of transferring configuration-relevant knowledge to employees has already been identified in earlier works. For example, Felfernig et al. [5] introduce a gamification-based approach to learning the major technical concepts of knowledge-based configuration and model-based diagnosis [13] – initial results of their studies show that the learning success of students can be increased. Compared to this approach, StudyBattles does not only support the dissemination of technical product configuration knowledge but also allows to include product domain knowledge into e-learning processes. Furthermore, StudyBattles includes gamification concepts which are implemented as *duels* where different users can play against each other in the context of a specific pre-selected learning application.

Felfernig et al. [6] analyze existing misconceptions of knowledge engineers when interpreting textual domain descriptions and recommend different measures that can help to reduce efforts related to configuration knowledge acquisition and maintenance. Finally, an analysis of the cognitive complexity of different types of knowledge formalizations is presented in [14] – the authors show that different types of representing logical implications can lead to significantly different outcomes in terms of knowledge understandability. Fur-

[1] Institute for Software Technology, Graz University of Technology, Austria, email: alexander.felfernig@ist.tugraz.at. ashehade@ist.tugraz.at. mjeran@ist.tugraz.at.
[2] Institute for Information Systems and Computer Media, Graz University of Technology, Austria and Curtin University, Western Australia, email: Christian.Guetl@iicm.tugraz.at.
[3] SelectionArts, Austria, email: s.reiterer@selectionarts.com.

thermore, different approaches to structure constraints can also have an impact on the underlying degree of understandability. Results of these studies have been integrated into a StudyBattles learning application (see Figure 1).

Besides exploiting user communities, e-learning content creation can be made more efficient by *automated generation generation mechanisms* – see, for example, [8, 12]. In this paper we show how configuration-related learning content (questions and related answers) can be automatically generated from a given configuration knowledge base. In addition to existing approaches to question generation from formal representations, we do not only focus on questions that refer to the set of possible solutions (configurations). We also show how questions can be generated that refer to *inconsistent situations* (e.g., no solution can be identified for a given set of customer requirements or the knowledge base becomes inconsistent with a given set of test cases) and to *qualitative properties of knowledge bases* (e.g., redundancies and further well-formedness properties).

Our contributions in this paper are the following. First, we provide an overview of the StudyBattles e-learning environment and show how the mentioned types of configuration-related knowledge can be represented in the system. Second, we show how configuration knowledge bases can be exploited to automatically generate e-learning content. Finally, we report initial results of a qualitative study related to the applicability of StudyBattles.

The remainder of this paper is organized as follows. In Section 2, we provide a short overview of the different functionalities provided in StudyBattles. As a basis for introducing a question generation approach, we define a configuration knowledge base from the domain of financial services that serves as a working example throughout this paper (Section 3). In Section 4 we show how questions can be automatically generated on the basis of a given configuration knowledge base. In Section 5 we present the results of a qualitative study related to the applicability of StudyBattles. In Section 6 we discuss future research issues and conclude the paper.

2 StudyBattles

The StudyBattles[4] start screen is shown in Figure 1 – it includes a list of subscriptions to learning applications (*LearnApps*) and further information regarding the ranking of the user in specific learning applications. Mobile clients for StudyBattles are available in *Android*, *iOS*, and *HTML-5* – Figure 1 depicts an example screenshot of an *iOS version*. The system can be deployed in a company's intranet and is also available as global server solution.[5] Users can join communities and subscribe to learning applications in which they can add learning content, practice exercises, and compete against other learning application users in a (quiz-based) duel. Contents within learning applications are organized in terms of categories, for example, the learning application "*Master Of Configuration*" includes the categories *Sales Knowledge*, *Conflicts*, *Diagnosis*, *Incompatibilities*, *Knowledge Acquisition*, and *Knowledge Representation*.

Deployments of StudyBattles. One version of the system has already been deployed and is applied by a large municipality and two universities in Austria. At the two mentioned universities, StudyBattles is applied in three *Software Engineering* courses (*Object-oriented Analysis and Design*, *Software Paradigms*, and *Requirements Engineering*) and in two *Artificial Intelligence* related courses

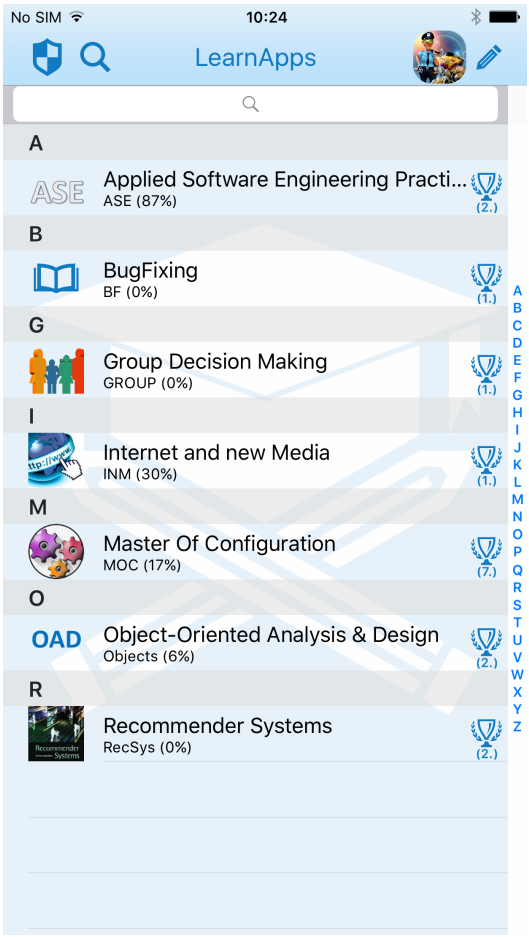


Figure 1. StudyBattles start screen (iOS version) consisting of learning applications (*LearnApps*) that can be subscribed by the user. Percentages report the share of already successfully answered questions. "*Master Of Configuration*" is the learning application that includes configuration-related knowledge.

(*Configuration Systems* and *Recommender Systems*). The goal of the StudyBattles instance deployed at one Austrian municipality is to increase employee's knowledge in security-related topics and also to transfer application-oriented knowledge related to a new accounting system. Currently, StudyBattles is also deployed for one of the largest financial service providers in Austria. The goal in this context is to support sales representatives in learning processes related to *product knowledge and sales practices*. Experts from these domains participated in a qualitative study where they gave feedback on system applicability (see Section 5).

Learning and training. After a StudyBattles learning application has been subscribed, users of this application can select categories and questions they want to answer. After having selected an answer to a question, immediate feedback is provided on the correctness of the answer. If an answer is wrong, related explanations can be provided to the user. Explanations can only be shown if these have been included by the expert who entered a question and related answers, i.e., in the current version of StudyBattles explanations are not determined automatically.

Content creation and question types. StudyBattles follows the concept of *crowd sourcing* where users can enter questions/answers

[4] StudyBattles has been developed within the scope of the PeopleViews research project which is funded by the Austrian Research Promotion Agency (843492).
[5] www.studybattles.com.

and expert users can evaluate the quality of the questions. The status of a *domain expert* is reached if a certain treshold of correctly answered questions is passed. Users are allowed to add additional content in terms of documents, pictures, and movies which serve as a basis for answering questions. When a user interacts with a learning application, questions are recommended [10] depending on their relevance for the user. Questions related to a category where a user has a low learning performance have a higher probability of being recommended to the user.

STUDYBATTLES supports the definition of different types of questions – examples thereof will be discussed in the following. Figure 2 depicts an example of a *multiple-choice question* that is related to the category *Sales Knowledge*. This question reflects relationships between customer requirements and financial services (equity fund, investment fund, and bankbook). The used abbreviations reflect the set of customer requirements $\{wr$ = *willingness to take risks*, di = *duration of investment*, and rr = *expected return rate*$\}$.

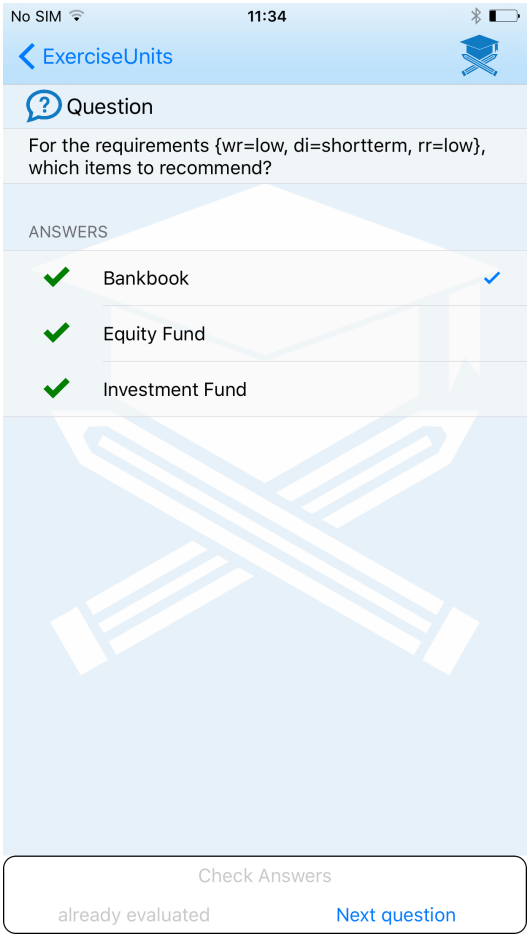


Figure 2. STUDYBATTLES: representation of multiple-choice questions – the check mark on the right hand side represents the answer of the user, check marks on the left hand side represent the correctness feedback.

An example of an *association task* is depicted in Figure 3 – the corresponding HTML-based definition interface is depicted in Figure 9. In association tasks, terms on the right-hand side have to be combined (associated) with the terms on the left-hand side. In the example of Figure 3, association tasks are exploited for asking questions that are related to the compatibility of customer requirements and products. Association tasks can also be applied to ask questions, for example, about the incompatibility of specific customer requirements. In the example of Figure 4 users are requested to combine individual customer requirements of the left and right hand side in such a way that the connected requirements become *inconsistent*. In Figure 5, a question is posed to educate users with regard to logical entailment (which situations lead to an empty set of solutions). In this example, only a high willingness to take risks is logically entailed in the item *Equity Fund*.

Figure 6 includes a question that is related to the correct usage of implications [6]. If a certain constraint is specified on a textual level (e.g., *a low willingness to take risks can only be combined with a bankbook*), the corresponding logical representation should be clear for all knowledge engineers. In order to avoid faulty translations, exercises such as the example depicted in Figure 6 can help to establish a standard of formalizing such properties. Finally, Figure 7 depicts an example question related to the identification of redundant constraints in configuration knowledge bases [4]. A constraint is considered as redundant if its deletion from the knowledge base is semantic-preserving, i.e., the solution space remains the same. On the logical level a constraint $c_a \in C$ is considered redundant if $C - c_a \models c_a$.

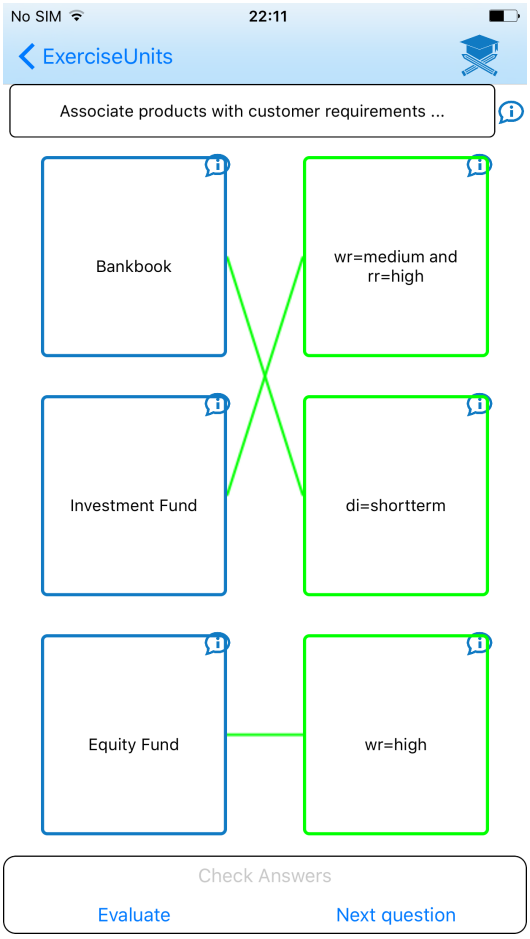


Figure 3. STUDYBATTLES: representation of associations tasks (concepts on the left have to be connected with the correct counterparts on the right).

Gamification. Users who trigger duels are then randomly assigned

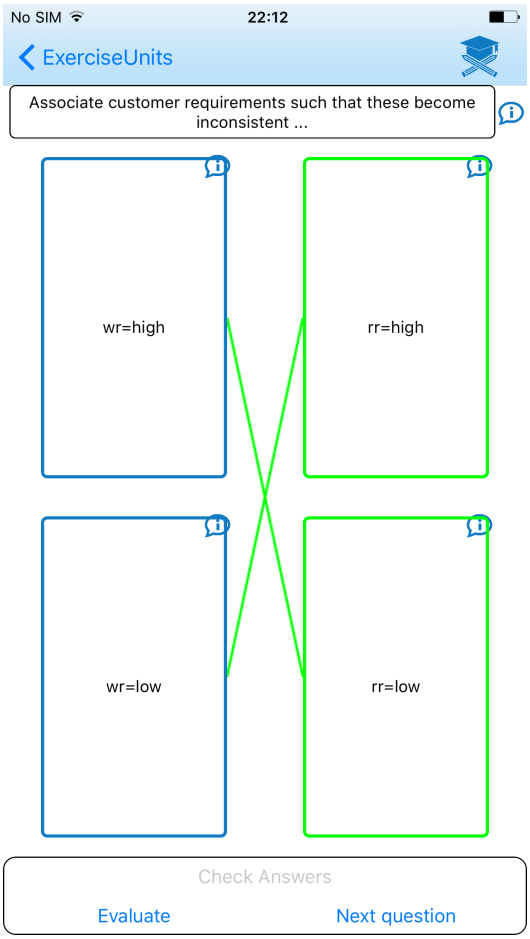


Figure 4. StudyBattles: Association task related to the association of incompatible requirements.

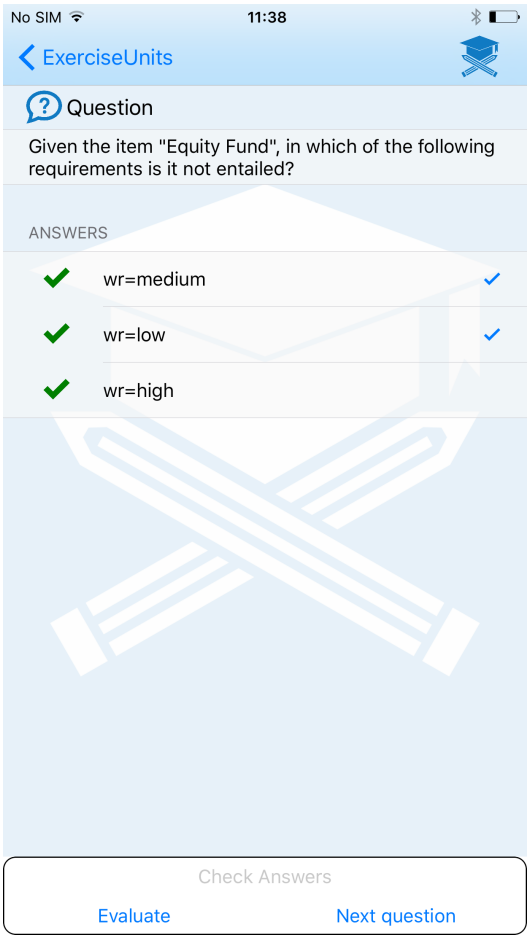


Figure 5. StudyBattles: representation of inconsistency-related knowledge in terms of a multiple-choice question.

to opponents – duels can be performed asynchronously. User ranking is visible on a local level (users are only able to see opponents directly ranked before or after them in a certain learning application). If a user wins a duel, he/she receives corresponding StudyBattles Points which is a major motivation for users to engage in games. The higher the complexity of an answered question, the higher the amount of received StudyBattles Points. The complexity of a question can be evaluated directly after having answered the question (see the *Evaluate* link, for example, in Figure 5).

Analysis of learning performance. StudyBattles supports different types of statistics that help to analyze the strengths and weaknesses of the user community and to establish needed counter measures (e.g., improving/adapting some parts of the learning material). In each learning application, each user has access to a ranking where the "direct-neighbor" opponents including their StudyBattles Points are shown. Administrators of a StudyBattles community have access to statistics that indicate the overall learning performance per learning application and also per topic inside a learning application. This way, strengths and weaknesses of a StudyBattles learning community can be identified and corresponding counter-measures, for example, in terms of improving specific learning contents, can be triggered.

3 Example Configuration Knowledge Base

As a working example we introduce a simplified financial services configuration task. Before introducing the example, we provide a basic definition of a configuration task and a corresponding solution (configuration).

Definition 1 (Configuration Task). A configuration task can be defined as a Constraint Satisfaction Problem (CSP) (V, D, C, REQ) where V are variables, D are domain definitions for the variables, C is a set of constraints[6], and REQ is a set of customer requirements.

Definition 2 (Configuration). A configuration (solution) for given configuration task (V, D, C, REQ) is a complete set $conf$ of variable assignments $v_i = a$ to the variables $v_i \in V$ ($v_i = a \rightarrow a \in$ domain(v_i)) with *consistent*($conf \cup C \cup REQ$). The set of solutions for a given configuration task is denoted as $CONFS$.

Configurations (one configuration task can have more than one solution) can be ranked according their utility for the user (customer). In this context, configurations with the highest utility for the user can be regarded as recommendations – for details we refer to [4].

The following is a simple *financial services configuration knowledge base* formulated as configuration task (see Definition 1). The variables in V are the following: *willingness to take risks* (wr), *du-*

[6] Note that $C = \{c_1, c_2, ..., c_n\}$ can also be represented as $C = \{c_1 \wedge c_2 \wedge ... \wedge c_n\}$ since constraints are assumed to be connected conjunctively.

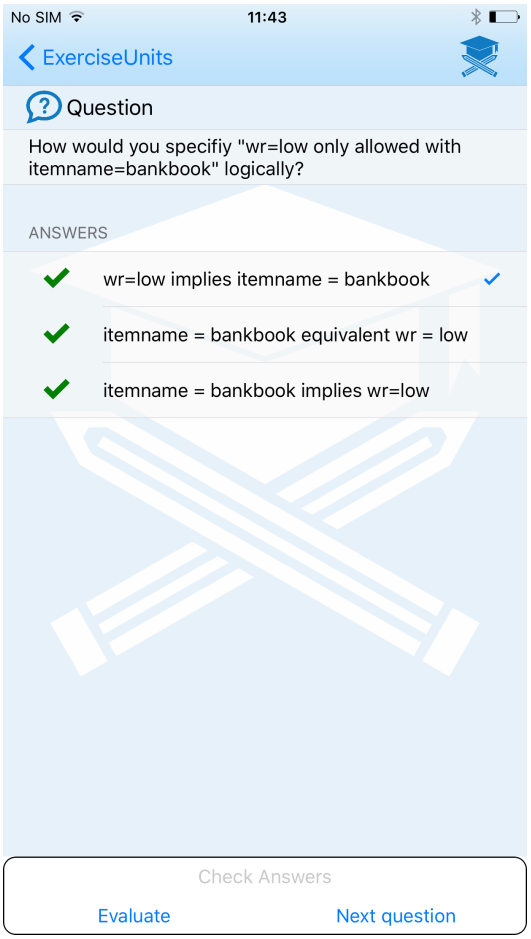


Figure 6. STUDYBATTLES: question related to the usage of implications.

Figure 7. STUDYBATTLES: question related to the identification of redundant constraints in configuration knowledge bases.

ration of investment (di), *expected return rate* (rr), and $itemname$ is a variable which represents the name of a financial service.

- $V = \{wr, di, rr, itemname\}$
- D = {domain(wr) = {low, medium, high}, domain(di) = {shortterm, mediumterm, longterm}, domain(rr) = {low, medium, high}, domain($itemname$) = {equityfund, investmentfund, bankbook}}
- $C = \{c_1 : \neg wr = low \vee itemname = bankbook, c_2 : \neg wr = medium \vee itemname \neq equityfund, c_3 : \neg di = shortterm \vee itemname = bankbook, c_4 : \neg di = mediumterm \vee itemname \neq equityfund, c_5 : \neg(rr = high \vee rr = medium) \vee itemname \neq bankbook, c_6 : \neg(wr = low \wedge rr = high), c_7 : \neg(di = shortterm \wedge rr = high), c_8 : \neg(wr = high \wedge rr = low)\}$
- $REQ = \{r_1 : wr = low, r_2 : di = shortterm, r_3 : rr = low\}$

A configuration for our example configuration task is the set of variable assignments $conf = \{wr = low, di = shortterm, rr = low, itemname = bankbook\}$ since the customer requirements included in REQ are consistent with the constraints in C. If we change the specification of REQ this can lead to situations were requirements become inconsistent with the constraints in C, i.e., no solution can be found. Such a situation is triggered in the case that $REQ = \{r_1 : wr = low, r_2 : di = shortterm, r_3 : rr = high\}$, i.e., $REQ \cup C$ is inconsistent. In such situations, model-based diagnosis [13] can be exploited to identify minimal sets of requirements that have to be adapted or deleted such that a solution (configuration) can be found. The identification of such adaptations can be formulated as diagnosis task (see Definitions 3–4).

Definition 3 (Diagnosis Task). A diagnosis task is defined as a tuple (C, REQ) where C is a set of constraints, REQ is a set of customer requirements, and $REQ \cup C$ is inconsistent.

Definition 4 (Diagnosis). A set $\Delta \subseteq REQ$ for a given diagnosis task (C, REQ) is a diagnosis if $REQ - \Delta \cup C$ is consistent, i.e., Δ is a set of requirements to be deleted from REQ such that consistent($REQ - \Delta \cup C$). Δ is *minimal* if $\neg\exists\Delta'$: $\Delta' \subset \Delta$.

Diagnoses are often denoted as hitting sets [13]. The original algorithm for determining minimal hitting sets is introduced in [13].

Finally, conflicts (also denoted as conflict sets) represent sets of requirements (in REQ) that are able to induce an inconsistency with C (see the following definition). Conflict sets can be exploited for the determination of diagnoses but also for the determination of redundant constraints in knowledge bases (see, e.g., [4]). There is a duality between conflicts and diagnoses: a conflict set is a hitting set for a set of minimal diagnoses and – vice versa – a diagnosis is a hitting set for a set of minimal conflicts [4].

Definition 5 (Conflict Set). A set $CS \subseteq REQ$ is a conflict set if $CS \cup C$ is inconsistent. CS is *minimal* if $\neg\exists CS'$: $CS' \subset CS$.

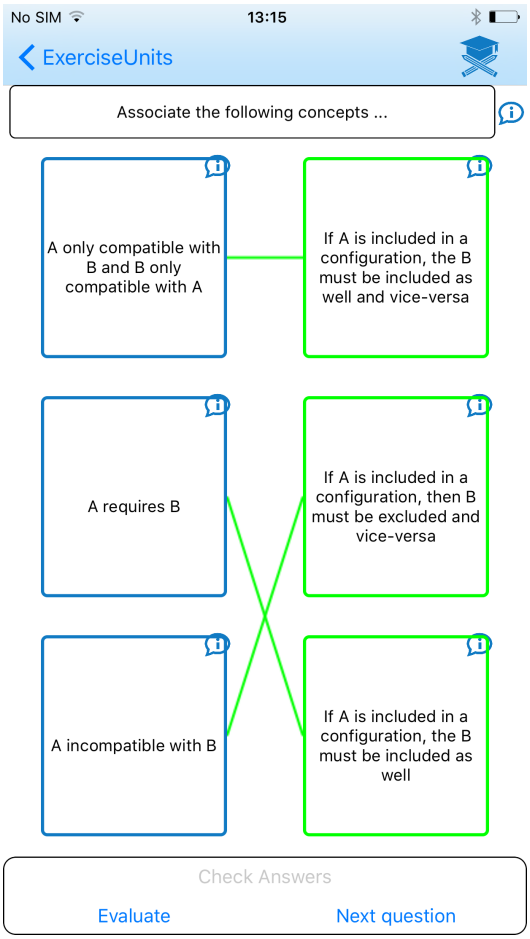


Figure 8. StudyBattles: association task related to the modeling concepts of a configuration language.

Types of sales knowledge. Sales knowledge is included in different forms in configuration knowledge bases. A configurator can determine items that can be recommended (*product knowledge*) on the basis of a given set of customer requirements (REQ). In situations where no solution can be identified for a given set of customer requirements, diagnosis algorithms [13] can determine the needed minimal changes to help the user out of the *no solution could be found dilemma* (*analysis knowledge*). Finally, given a set of requirements (REQ), a conflict detection algorithm [11] can determine a minimal set of customer requirements which induce an inconsistency with C (*inconsistency knowledge*). In the following we discuss how configuration task definitions can be exploited for the automated generation of related questions for StudyBattles.

Types of engineering knowledge. Engineering knowledge entails relevant practices when building and maintaining configuration knowledge bases. Knowledge engineers need to understand how to *formalize* different product related constraints (e.g., how should requirements relationships be specified on a logical level – see Figure 6). It is also important to understand *well-formedness criteria* relevant for knowledge base development and maintenance (e.g., what are redundant constraints – see Figure 7). Furthermore, knowledge engineers need to know the building blocks of the language used for configuration knowledge representation. A simple related StudyBattles question is depicted in Figure 8.

4 Generating Questions from Configuration Knowledge Bases

Some of the StudyBattles questions can be automatically generated from a configuration task definition. In the following we show how questions (and related answers) can be generated for some of the configuration knowledge types discussed in Section 3. Generated questions can be included in StudyBattles and then used for training purposes. These questions can be exploited by employees to improve their configuration-related knowledge.

Product knowledge. The task of a user is to identify configurations that are consistent with a given set of customer requirements ($REQ = \{r_1, r_2, ..., r_m\}$). Related correct and faulty answers can be generated by a constraint solver on the basis of a configuration task (V, D, C, REQ). A constraint solver calculates configurations that satisfy $REQ \cup C$. Furthermore, *non-solutions* satisfy $REQ \cup \overline{C}$.[7] $REQ = \{r_1, r_2, ..., r_m\}$ then is the basis of a question, *solutions* represent correct answer(s), and *non-solutions* represent faulty answer(s). Since the potential number of solutions and non-solutions can be high, a random number thereof is selected for inclusion in StudyBattles.

Product knowledge (example). Given our example configuration task definition, customer requirements could be $REQ = \{r_1 : wr = low, r_2 : di = shortterm, r_3 : rr = low\}$, a solution (correct answer) is $\{name = bankbook\}$, and non-solutions are $\{name = equityfund, name = investmentfund\}$ (see Figure 2). A related question posed in StudyBattles is: *For the requirements ..., which items to recommend?*

Analysis knowledge. Assuming that $REQ \cup C$ is inconsistent (and C is consistent), the task is of a user is to figure out which minimal set of $r_i \in REQ$ has to be deleted such that consistency can be restored. More formally, analysis knowledge related questions can be generated using a diagnosis task (C, REQ). The diagnosis task definition (C, REQ) can be used for question representation, related answers are represented by the diagnoses Δ_i. Non-diagnoses can be easily determined on the basis of a calculated set of diagnoses: if, for example, $\Delta_i = \{r_a, r_b, r_c\}$ is a minimal diagnosis, then $\Delta_{in} = \{r_a, r_b\}$ is a corresponding non-diagnosis since a proper subset of minimal diagnosis is not a diagnosis. Since the potential number of diagnoses and non-diagnoses can be high, the answer set is composed of a random number of selected diagnoses and non-diagnoses.

Analysis knowledge (example). Given our configuration task definition with the customer requirements $REQ = \{r_1 : wr = low, r_2 : di = shortterm, r_3 : rr = high\}$. Alternative minimal sets of customer requirements (diagnoses Δ_i) that have to be deleted from REQ such that a solution can be identified, are: $\{\Delta_1 = \{r_1, r_2\}, \Delta_2 = \{r_3\}\}$, i.e., deleting the requirements r_1 and r_2 restores consistency between REQ and C. An example of a non-diagnosis related to diagnosis Δ_1 is $\{r_1\}$. A related question posed in StudyBattles is: *Given the configuration task definition ... which one is a minimal set of requirements that have to be deleted from REQ such that consistency can be restored?*

Inconsistency knowledge. Assuming that $REQ \cup C$ is inconsistent (and C is consistent), the task is of a user is to figure out which minimal set of $r_i \in REQ$ is inconsistent with C. More formally, inconsistency knowledge related questions can be generated on the basis of a conflict detection task (C, REQ). The conflict detection task (C, REQ) can be used for question representation, related correct answers are represented by the conflict sets CS_i. Non-conflicts (faulty answers) can be easily determined on the basis of a calculated

[7] If $C = \{c_1 \wedge c_2 \wedge ... \wedge c_n\}$ then $\overline{C} = \{\neg c_1 \vee \neg c_2 \vee ... \vee \neg c_n\}$.

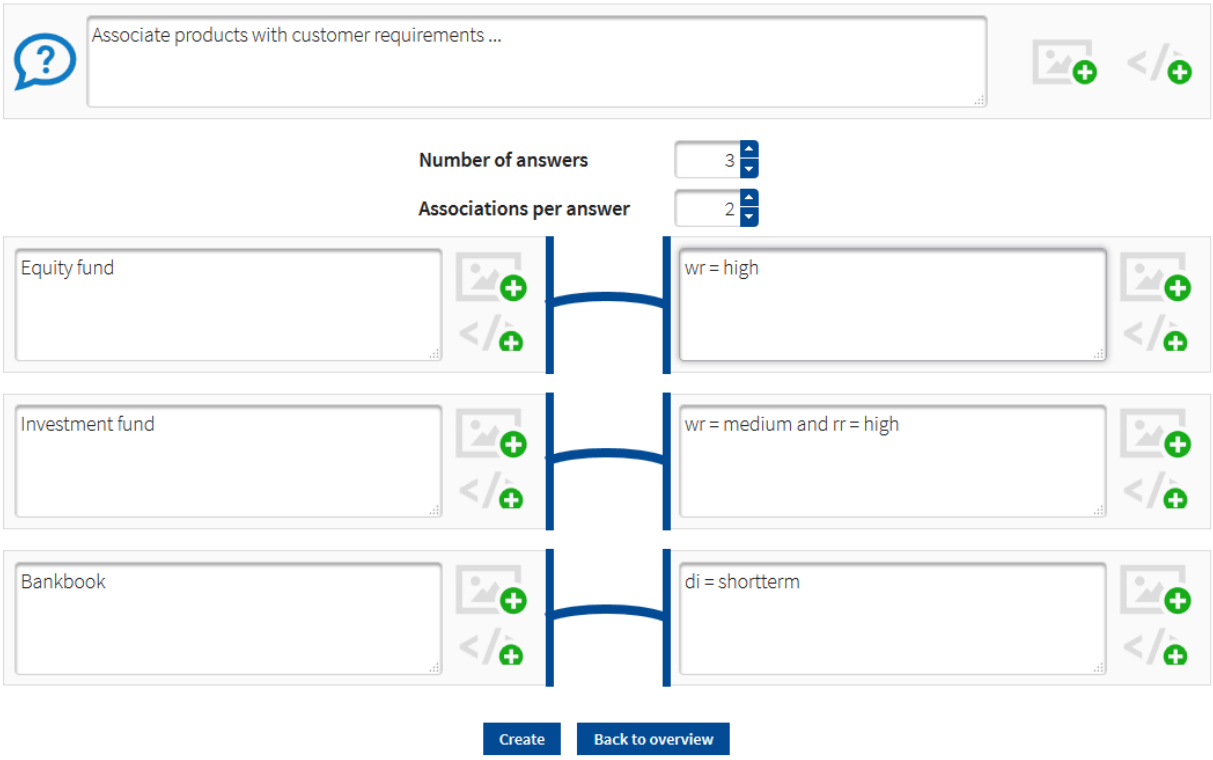


Figure 9. StudyBattles: definition of an association task (specification of correct pairs) in the HTML-5 version.

set of conflicts: if, for example, $CS_i = \{r_a, r_b\}$ is a minimal conflict, then $CS_{in} = \{r_a\}$ is a non-conflict since subsets of minimal conflicts do not have the conflict property. Since the potential number of conflicts and non-conflicts can be high, the answer set is composed of a random number of selected conflicts and non-conflicts.

Inconsistency knowledge (example). Given our configuration task definition with the customer requirements $REQ = \{r_1 : wr = low, r_2 : di = shortterm, r_3 : rr = high\}$. Alternative minimal sets of customer requirements subset of REQ that are inconsistent with C are: $\{CS_1 = \{r_1, r_3\}\}$ and $\{CS_2 = \{r_2, r_3\}\}$. An example of a non-conflict related to conflict set CS_1 is $\{r_1\}$. A related question posed in StudyBattles is: *Given the configuration task definition ... which one is a minimal set of requirements that have to be deleted from REQ such that consistency can be restored?*

For questions related to the *formalization* of informal domain descriptions and constraints (see Figure 6) we do not offer an automated question generation mechanism. The same holds for *modeling concepts* for the development and maintenance of configuration knowledge bases. Questions related to *well-formedness criteria* for the development of configuration knowledge bases can be automatically generated. For example, a configuration knowledge base containing redundant constraints (the question part) can be presented to the user. The correct answers (redundant constraints) can be identified by corresponding redundancy detection mechanisms – for details we refer to [4]. Other examples of well-formedness rules are discussed in [3].

5 StudyBattles Evaluation

StudyBattles has been evaluated within the scope of a qualitative study (N=15 participants). Participants from different domains (financial services, public administration, telecommunications, and universities) provided feedback on the applicability and usefulness of StudyBattles. Major mentioned potential improvements that come along with StudyBattles are, improved knowledge retention in organizations, improved knowledge sharing between users on the basis of community-based (crowd-sourced) knowledge acquisition processes, increased motivation to learn, improved skills, increased fun and interest in the topic, increased competition level between users (e.g., sales representatives), improved quality of service with regard to customers, increased learning efficiency, and enhanced possibilities of community knowledge analysis which provide a basis for a fine-grained adaptation of learning material.

The *application of an e-learning environment in configuration scenarios* was motivated by discussions with different companies applying configuration technologies. Especially in distributed scenarios where large-scale configuration knowledge bases have to be developed and maintained, additional learning mechanisms have to be provided to establish a standard knowledge level that helps to reduce erroneous maintenance practices as well as suboptimal sales practices. A major issue in this context is that existing commercial configuration environments still do not provide intuitive knowledge representation mechanisms and there is a need to further educate domain experts and knowledge engineers.

6 Conclusions and Future Work

In this paper we introduced the idea of applying an e-learning environment (StudyBattles) as a complementary approach to transfer configuration-related knowledge to employees (e.g., sales representatives and knowledge engineers). We provided an overview of differ-

ent system functionalities such as community-based content development, gamification, and automated question generation. In the context of automated question generation we focused on the two aspects of generating sales and engineering related knowledge. More fine-grained question generation techniques that provide mechanisms to more systematically pre-select answers to be included are within the scope of future work. In this context we will also analyze potential synergies with existing approaches to test case generation in software engineering [2]. Especially in the context of educating sales representatives, automated question generation becomes a key functionality, since this reduces the overheads of manual content generation and management which is often the task of only a small group of persons. In future versions of StudyBattles, additional question types will be included. For example, we will provide mechanisms that allow to generate not only questions related to diagnoses (analysis knowledge) but also to related repair actions (i.e., changes in the requirements that lead to the identification of at least one solution).

REFERENCES

[1] M. Aldanondo and E. Vareilles, 'Configuration for mass customization: how to extend product configuration towards requirements and process configuration', *Journal of Intelligent Manufacturing*, **19**(5), 521–535, (2008).

[2] S. Anand, E. Burke, T. Chen, J. Clark, M. Cohen, W. Grieskamp, M. Harman, M. Harrold, and P. Mcminn, 'An orchestrated survey of methodologies for automated software test case generation', *Journal of Systems and Software*, **86**(8), 1978–2001, (1987).

[3] D. Benavides, S. Segura, and A. Ruiz-Cortez, 'Automated analysis of feature models 20 years later: A literature review', *Information Systems*, **35**(6), 615–636, (2010).

[4] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann Publishers, 1st edn., 2014.

[5] A. Felfernig, M. Jeran, T. Ruprechter, A. Ziller, S. Reiterer, and M. Stettinger, 'Learning games for configuration and diagnosis tasks', in *16th International Workshop on Configuration*, pp. 111–114, Vienna, Austria, (2015).

[6] A. Felfernig, S. Reiterer, M. Stettinger, and J. Tiihonen, 'Towards understanding cognitive aspects of configuration knowledge formalization', in *Vamos 2015*, pp. 117–124, Hildesheim, Germany, (2015).

[7] BJ Fogg, *Persuasive Technology*, Morgan Kaufmann Publishers, 1st edn., 2003.

[8] C. Gütl, K. Lankmayr, J. Weinhofer, and M. Höfler, 'Enhanced automatic question creator eaqc: Concept, development and evaluation of an automatic test item creation tool to foster modern e-education', *Electronic Journal of e-Learning*, **9**, (2011).

[9] L. Hvam, N. Mortensen, and J. Riis, *Product Customization*, Springer, 1st edn., 2010.

[10] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*, Cambridge University Press, New York, NY, USA, 1st edn., 2010.

[11] U. Junker, 'Quickxplain: Preferred explanations and relaxations for overconstrained problems', in *19th National Conference on Artificial Intelligence (AAAI04)*, pp. 167–172, (2004).

[12] S. Rakangor and Y. Ghodasara, 'Literature review of automatic question generation systems', *International Journal of Scientific and Research Publications*, **5**, (2015).

[13] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).

[14] A. Felfernig S, Reiterer, M. Stettinger, F. Reinfrank, M. Jeran, and G. Ninaus, 'Recommender systems for configuration knowledge engineering', in *Workshop on Configuration*, pp. 51–54, Vienna, Austria, (2013).

[15] M. Stumptner, 'An Overview of Knowledge-based Configuration', *AI Communications*, **10**(2), 111–126, (1997).

[16] D. Zhang and J. Nunamaker, 'Powering E-Learning In the New Millenium: An Overview of E-Learning and Technology', *Information Systems Frontiers*, **5**(2), 207–218, (2003).

Finding pre-production vehicle configurations using a Max-SAT framework

Marcel Kevin Tiepelt [1] and **Tilak Raj Singh** [2]

Abstract. The increase in vehicle configuration induces new components and interfaces requirement at the final assembly. Some of these interfaces have to be tested prior to the series production. Each test has a prerequisite set of product features required for testing. The pre-production vehicles are required to cover each and every test defined for the product. To minimize the cost of the testing, a minimal number of the pre-production vehicles is required. However, finding a small set of special configured vehicles in an enormously large product variant is a daunting task. Each configuration is required to satisfy complex feasibility rules among product attributes, in order to be producible at the final assembly. Thus decision of finding a minimal set of configuration for pre-production requires to satisfy all configuration related rules as well. In this paper a Satisfiability based framework, using SAT and Max-SAT models to find a possible set of pre-production vehicles, is being discussed. Product configuration rules and test requirements are converted to propositional logic which then are solved using Max-SAT framework. The formulation leads to a large scale optimization model which is solved using a greedy algorithm applying a Max-SAT framework. Initial results are applied to the automobile data.

1 Introduction

The configuration like product setup in automotive industry allows customers to design their very own car. While the modular setup allows to build near to unique vehicles for each customer, the larger number of components greatly increases the product complexity [5]. The vast number of possibilities impact pre-production and planning phases. Many features of the vehicle are being evaluated before proceeding to the production phase. Features tested on prototypes may have a different behavior under real world conditions. To counteract unexpected behavior, tests can be performed on pre-production vehicles. During the testing phase the pre-production vehicle is exposed to extreme conditions, such as cold weather, extreme heat or high speeds. In opposition to the prototype vehicle the pre-production vehicle consists of a configuration aimed to behave similar to the end product produced for the customer. The resulting complete car features many of the characteristics of the costumer product, hence can be used to identify flaws prior to mass production. Therefor pre-production vehicles support the quality assurance by applying extreme real world conditions on properties and parts before vehicle features are released to the sales department.

The abstract representation of a vehicle is based on a large set of car parts including all possible vehicle configurations and customer choices. The assemble-to-order strategy used by many car-manufactures allow customers to customize the vehicle based on a predefined set of features, such as the built-in engine, or the interior. Providing a large set of customization options allows customers to build an almost unique vehicle. Additionally to the customer exposed set of features, an abstract car consists of a set of components required to compose a complete vehicle. Many of these components are implicitly implied by the customer choices, hence are necessary in order to install the features chosen by the customer. Every physical part is based on a characteristic part-rule consisting of instructions which parts have to be included or excluded within the construction process. Each feature chosen by the customer, as well as the conditional interaction with mandatory parts, influences a large set of additional physical parts and restricts the composition of car parts. The set of rules and inter-dependencies between the components determine whether a configuration is valid. The large set of components are their inter relations make it non-trivial to determine if a set of components evaluate to a valid vehicle.

The extensive amount of automation used in the automotive industry reduces the amount of work necessary to construct a single vehicle, unlike the construction of a pre-production vehicle. Pre-production vehicles require separate assembly lines and manual processing attempting to cover as many characteristics as possible and meeting special construction constraints. In order to reduce the development and evaluation cost of new features and considering the complex construction process of pre-production vehicles, it is desired to cover multiple features in each vehicle in order to conduct multiple tests. The aim is therefor, to reduce the number of vehicles required to cover all testing setups. Within the quality assurance process different evaluation methods are applied. Identifying flaws of vehicle features is either based on the built-in parts or on characteristics of the vehicle, such as sound isolation or cruise control. One of the test setups may therefor include the exposure of pre-production vehicles to extreme real-world conditions such as noise, or functionality of driver assistance systems. The characteristics are based on rule-sets, composites of physical parts, in order to implement the functionalities. A different setup may evaluate the effect of everyday use on the physical components such as the tires, the engine or electrical equipment.

Within the planning of abstract pre-production vehicles as well as the construction multiple challenges can be identified:

1. To evaluate the effect on physical parts, finding a minimal set valid vehicle configurations to cover all predefined parts.
2. To evaluate vehicle features and characteristics, finding a minimal set of valid vehicle configurations covering certain rule-sets.

The main contribution of this work is a large-scale framework that attempts to find a minimal number of pre-production vehicles based on the above challenges. The framework is based on the part and

[1] Aalen University, Germany, email: tiepelt@dev-nu11.de
[2] Mercedes Benz R & D, India, email: tilak.singh@daimler.com

rule-set configuration and utilizes a Max-SAT solver as a black box to issue optimization. The contribution of this paper is the following:

1. An abstract representation of testing setups on pre-production vehicles.
2. An heuristic approach to satisfy rule-sets and part coverage of pre-production test setups.
3. A large-scale greedy like framework using Max-SAT to compute a minimal set of pre-production vehicle configurations.

The work is organized as follows: The next section presents previous work in the field of product configuration and previous attempts to the mentioned challenges. The third section introduces notations and terms used throughout the paper. The forth section specifies the challenges in detail and gives examples to illustrate problem statement. In the following sections a general Max-SAT framework is introduced and problem based specializations are described to engage the problem of finding the pre-production vehicle configurations. The sixth section describes the industrial sizes problem instances used to evaluate the approach while the last section discusses the computational results obtained and gives an insight on further work that can be done.

2 Previous work

Computation on product configurations based on a set of constraints is a well-studied problem in many industrial areas. Carsten Sinz was the first to apply Boolean Satisfiability methods to large automobile configuration problem [11]. Walter et. al. [14] has applied Max-SAT based approach to transform a in feasible configuration into a feasible configuration by considering configuration rules and customer preferences. This would help to reconfigure a configuration while preserving a certain set of features. Singh and Rangaraj [10] have used Hamming distance based models to describe distinction of multiple configurations. Singh et. al. [9] also introduce an approach to generate a pool of configurations to cover a set of test rules using a column generation model.

The Ford Motor company [2] has built a multi-phase mathematical model to solve a problem instance using a set cover approach. The possible configurations are derived from a compatibility matrix and computed with an integer linear programming model. In the first phase of the model a set covering problem is solved in order to find the minimal number of configurations needed to cover a set of predefined test, each featuring different requirements and constraints.

In [6] Limtanyakul approaches a similar set cover problem paired with the problem of scheduling tests. With regard to the set cover Limtanyakul assumes that all prototype configurations are the same, such that the computation of these is not necessary anymore. A different approach by [3] uses a heuristic to assign product configurations to test variants and allows relaxing of the component configurations to find better fitting results. A recent paper by Walter et. al [13] suggests a similar approach to cover the equipment options of test vehicles. In their paper they present a greedy like algorithm utilizing SAT calls to test the configuration compatibility of equipment options. Additionally they introduce exact algorithms using linear search and branch & price.

3 Notations

In the following the problem of finding a valid and complete product configuration is equivalent to finding a satisfying assignment to the Boolean satisfiability problem (SAT), where the dependencies of car parts and attributes are represented by Boolean constraints.

Given a propositional φ over a set of Boolean variables V, if an assignment to V evaluates φ to *true*, this assignment is called satisfying or a model of φ. In the following, car parts and attributes are represented as a set of variables. The set of variables is limited to a finite domain as the framework aims to be a practicable real-life model. The dependencies and requirements between those parts and codes are described by propositional Boolean formulas. To describe a product configuration each part and code is assigned a unique variable, such that a model consists of an assignment to the variables. Each variable has to be assigned *true* if this part or code is included in the configuration, or *false* otherwise. As for the SAT problem the propositional formulas are in conjuncture normal form (CNF), that is a conjunction of clauses C, such that each clause consists of a disjunction of literals. A literal l is a variable or its negation:

$$SAT(\varphi) = \quad C_1 \cap C_2 \cap ... \cap C_n \tag{1}$$

$$C = \quad (l_1 \cup l_2 \cup ... \cup l_m) \tag{2}$$

The Maximum satisfiability problem (Max-SAT) is a generalization of SAT asking for the maximum number of the clauses that can be satisfied. Partial Max-SAT is an extension to Max-SAT such that the set of clauses is divided into two subsets $hard, soft$. A solution to Partial Max-SAT (PMax-SAT) must satisfy all $hard$ clauses and the maximum number of $soft$ clauses. The extension Weighted Partial Max-SAT (WPMax-SAT) adds weights ω_i to the clauses. Each hard clause gets infinite weight $\omega_{hard} = \infty$, each soft clause gets a finite weight. An optimal solution to the WPMax-SAT problem asks for the maximum sum of weight assigned to a model. Remark that WPMax-SAT with same weight for all soft clauses is equivalent to Partial Max-SAT. [1]

To abstract the task of covering a set of vehicle components the set covering problem is introduced. Given a universe $U = \{u_1, u_2, ..., u_n\}$ and a collection of subsets of U : $S := S_1, S_2, ..., S_m$. The goal is to find the smallest collection of subsets to cover all elements in U. [12]

4 Problem description

With thousands of components and attributes combined in a single vehicle, finding a valid configuration tends to be a computationally non-trivial task. Generating all possible unique configurations may be infeasible in many cases, reducing to the well-known $\#P$-complete problem of model counting when only trying to find their number [5]. The possibly exponential number of unique configurations may make it infeasible to iterate over all models and select a certain subset. Instead one can think of the set of constraints as an abstract model implicitly containing all possible configurations. In the following, the two different problems are considered leading to different choices of configuration subsets.

It is assumed that the vehicle parts and constraints are represented by Boolean formulas, such that each car part is mapped to a variable. A car part can either be included in a vehicle configuration, resulting in the variable being assigned *true*, or respectively excluded resulting in the assignment *false*. With a vehicle being constructed by a finite number of components the domain (respectively the co-domain) for any SAT related problem statements or solutions in the following are also finite.

Problem I The introduction of new components into a product line, in e.g. a face lift for an existing generation, the components introducing new features have to be evaluated using a complete vehicle. Facing the complex construction and planning process it may be desired to get the most out of each constructed vehicle. Therefor each vehicle must cover as many different new components as possible.

The abstract problem is to find a minimal choice of configurations to cover the predefined set of new components, constrained by inter-dependencies and part relations. In Problem I the universe U represents all available vehicle components. It may suffice to conduct test on a subset of newly introduced car parts represented by a subset $U_T \subseteq U$ of test components. The subsets S contains all the valid configurations that can be possibly build with the parts contained in the universe U, but constrained by the rule-set. The goal is to find a subset of configurations $S_T \subseteq S$ to cover all the parts of the test-universe U_T.

Example I The universe U for the first vehicle configuration consists of an $engine(E)$, a $chassis(H)$, a $steering\ wheel(S)$ and different sets of $tires(T_1, T_2)$, hence $U = \{E, H, S, T_1, T_2\}$. The aim is to evaluate the effect of high speed on the performance of the tires, hence $U_T = \{T_1, T_2\}$, while utilizing the least number of different vehicle configurations as possible.

In order to perform a speed test a complete car is required, hence the configuration must include each component as depicted in Table 2. The configuration constraint restricts the built-in tire sets to one per vehicle. The goal is to find a set of configurations covering all elements in U without raising a conflict with the constraints given in Table 2.

Table 1. Component description

Component	Variable name
Engine	E
Chassis	H
steering wheel	S
tires	T_1, T_2

Table 2. Configuration constraints for building a feasible configuration

Restriction	Configuration constraint
[1]	$E \wedge S \wedge H \wedge (T_1 \vee T_2)$
[2]	$-T_1 \vee -T_2$

Problem II The evaluation of vehicle characteristics such as noise control, or heat resistance is being performed on complete vehicles. In order to assure the smooth functioning of all features the successful integration of all involved components must be guaranteed. Therefor each characteristic features a rule-set of components which have to be present in order to implement a certain characteristic.

Hence, finding the minimal number of pre-production vehicles can be described finding a minimal number of configurations such that the rule-sets mapped to all characteristics are satisfied. In Problem II the universe U represents all available vehicle components. In contradiction to Problem I it is no longer required to cover a set of parts, but rather a set of rules. Therefor the universe U_T consists of rule-sets r_i, each featuring a set of rules constraining the included components. The subset S contains all valid configurations to construct a vehicle. The aim is to find a minimal subset of configurations $S_T \subseteq S$ such that all rule-sets $r \in U_T$ are satisfied.

Example II For the second problem the universe U used in Example I is extended by a $active\ noise\ cancellation\ system(A)$, a $sealing(L)$ and a diesel $engine(D)$. The aim is to provide configurations for tests on noise protection and the speed limit with the new diesel engine. The universe to cover consists of the rule-sets $U_T = \{r_1, r_2, r_3\}$ restricted by the constraints given in Table 3. The minimal set of configurations to is constrained by the dependencies given in Table 4.

Table 3. Test rule-set

Test requirement	rule-set
r_1: Noise cancellation	A
r_2: Noise protection	L
r_3: Speed limit	D

Table 4. Configuration constraints for building feasible configuration, Example II

Restriction	Configuration constraints
[3]	$(E \vee D) \wedge S \wedge H \wedge (A \vee L)$
[4]	$-D \vee -E$
[5]	$-D \vee -A$

5 Max-SAT Framework

The configuration constraint set is the implicit representation of all possible configurations of a vehicle. Assuming it is impossible to sample all valid configurations, the presented greedy like approach is considered to select the most fitting subset of configurations. The greedy algorithm has the best approximation to the set cover problem, when choosing the subset with the largest number of uncovered elements in each iteration. As proven in [4], this results is the best approximation ratio that can be expected unless $P \neq NP$.

To approach the first problem the set of elements U represents the components which need to be covered. Therefore, the greedy like algorithm selects a subset S which contains the largest number of components who have not been included in any configuration. The algorithm makes a choice for the SAT variables such that the Boolean formulas are satisfied and simultaneously satisfy as many uncovered components as possible. As it seems to be impossible to select a configuration from a pre-computed list, a Max-SAT framework is considered to work on the implicit configuration set and generate candidate configurations.

Problem II is being approached by assigning each element u in U a set of constraints which have to be satisfied in order to cover the element u. The algorithm will therefor make a choice of variables which simultaneously cover the configuration constraints as well as the constraints assigned to a test u. In the following, the test constraints are referred to as test types.

The selection of the configurations covering the elements u builds upon consecutively checking if a configuration including a set of elements evaluates to true. The internal state of the Max-SAT frame-

work consists of the Max-SAT solver, building on the configuration constraints and the elements within the universe. Undergoing the solution finding process the Max-SAT solver will choose a set of elements, enforce their presence in a configuration and evaluate the validity. The objective of maximizing the number of elements that are chosen within a configuration is obtained by keeping track of the so far maximal amount of elements satisfying a configuration. Once the amount of cover-able elements does not increase any more, the Max-SAT solver will terminate. All covered elements are excluded from the next solution process.

While the framework will find the optimal (maximal) number of interfaces to cover in each iteration, it cannot be guaranteed that the number of configurations obtained to cover all elements is optimal.

5.1 Problem 1: Part coverage

The method used to approach Problem I selects a candidate configuration that covers a set of components, determines if the candidate is a valid configuration and decides if the configuration can be added to the solution set. The components to be covered are referred to as interfaces. Given a set of Boolean constraints in CNF one can validate a product configuration by checking for satisfiability. The SAT model to validate a product configuration can be converted to a equivalent partial Max-SAT model featuring hard clauses only. Due to the fact that all of the configuration constraints have to be satisfied to determine a valid configuration, all clauses within the configuration constraints are considered *hard* clauses. To easily distinguish between hard and soft clauses, the hard clauses are assigned weight $\omega_{hard} = \infty$ and the soft clauses are assigned weight $\omega_{soft} = 1$. Let Σ be the set of constraints and $C_1, ..., C_{m'}$ the respective clauses. In the partial Max-SAT model the resulting clauses are

$$\Sigma_{hard} = \{(C_1, \omega_1) \wedge (C_2, \omega_2) \wedge ... \wedge (C_{m'}, \omega_{m'})\} \tag{3}$$

such that the weight for all hard clauses is infinite:

$$\omega_k = \infty \quad \forall k = \{1, 2, ..., m'\} \tag{4}$$

The resulting Partial Max-SAT instance does not have any soft clauses and is equivalent to a SAT instance. We extend this Partial Max-SAT instance by adding a set of unit clauses $C_1, ..., C_k$ such that each interface is assigned a unit clause. The weights assigned to the interface clauses are equal to 1, such that the instance is weighted, but it suffices to distinguish between hard clauses with infinite weight, and soft clauses with weight one. Let m be the number of components, than the following clauses are conjunctive with Σ:

$$\Sigma_{soft} = \{\Sigma \wedge (C_1, \omega_1) \wedge (C_2, \omega_2) \wedge ... \wedge (C_m, \omega_m)\} \tag{5}$$

such that the weight of all clauses is equal to one:

$$\omega_k = 1 \quad \forall k \in \{1, 2, ..., m\} \tag{6}$$

As a result an instance that aims to find a configuration satisfying as many components as possible while retaining integrity within the basic constraints is obtained. The hard clauses represent the interdependencies and constraints for each component, such that an assignment, satisfying the hard clauses, results in a valid configuration. Therefore, each component is implicitly associated with a set of valid configurations satisfying the hard clauses while choosing this component. To ensure that the predefined test components are chosen, the soft clauses map a weight to each component as unit clause that is listed for testing. The algorithm maximizes the sum of all weights, hence covers as many components $C_1, ..., C_m$ in each iteration as possible. At all time the infinite weight of the hard clauses $C_{m+1}, ..., C_{m+m'}$ guarantees that no dependencies are violated:

$$\Sigma_I = \{ \quad (C_1, 1) \wedge ... \wedge (C_m, 1), \tag{7}$$

$$(C_{m+1}, \infty) \wedge ... \wedge (C_{m+m'}, \infty)\} \tag{8}$$

The presented greedy approach assumes the input of a partial Max-SAT instance Σ_I as described in Section (7) and (8). For each variable to cover there exists a soft clause C_k where $k = \{1, ..., m\}$. The input consists of the Boolean formula Σ and the set of soft clauses C. Resolving the instance in each iteration results in a configuration $\mathcal{S}$ which covers as many uncovered interfaces as possible. The algorithm computes a set of configurations $\mathcal{C}$, such that each configuration covers at least one C_k. In Algorithm 1 there is a black box function $solve(\varphi)$ considered that gives a solution to a Partial Max-SAT instance φ.

Algorithm 1.

PartCoverage(Σ_I, C)
$\mathcal{C} \leftarrow \emptyset$
WHILE ($C \neq \emptyset$)
{
$\mathcal{S} \leftarrow solve(\Sigma_I)$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{S}\}$
$C \leftarrow C \setminus \{C_k | \mathcal{S}(k) = satisfied\}$
}
Return $\mathcal{C}$

Taken the Example I in Section 4, the generated hard clauses contain the configurations restrictions [1] and [2], the soft clauses contain the unit clauses T_1, T_2, such that the Max-SAT instance in Equation (9) follows:

$$\Sigma_I \quad = \{(T_1, 1), (T_2, 1), \tag{9}$$

$$(E, \infty), (S, \infty), (H, \infty), \tag{10}$$

$$(T_1 \vee T_2, \infty), (-T_1 \vee -T_2, \infty)\} \tag{11}$$

The algorithm consecutively tries to add the tire sets to a configuration. When conflicting with the hard clauses each one of the tire-sets is relaxed. Possible results for configurations covering the tire sets are presented in Table 5.

Table 5. Possible configurations to cover tire sets

Configuration	Included parts
1	E, S, H, T_1
2	E, S, H, T_2

5.2 Problem II: Test coverage

Problem II deals with the task of finding a minimal number of configurations satisfying a set of test types, each assigned a constraint set required to be satisfied in order to perform a test. In opposition to Problem I in Section 5.1 it is no longer sufficient to satisfy a single component, but rather necessary to satisfy all assigned constraints.

Let Σ be the basic partial Max-SAT model described in Section 5.1. Let m be the number of tests to be conducted. For each test

there exists a variant σ_i with $i = \{1, 2, ..., m\}$ consisting of a set of propositional formulas in CNF.

$$\sigma_i = \{C_1, C_2, ..., C_r\} \tag{12}$$

The index r represents the number of clauses assigned to variant i. A new Boolean variable t_i is introduced for each test which is being used to determine the variants that are covered in a vehicle configuration. The Max-SAT instance is being extended by a set of hard clauses ensuring that a variable assignment satisfies the configuration constraints if and only if all of the variant clauses evaluate to *true*. For a given test type σ_i the clauses C_i are added such that the assignment of t_i implies each included variant:

$$\Sigma_{hard_i} = \{(t_i \rightarrow C_1, \omega_1) \wedge ... \wedge (t_i \rightarrow C_r, \omega_r)\} \tag{13}$$

$$\omega_k = \infty \quad \forall\, k = \{1, 2, ..., r\} \tag{14}$$

The implication $t_i \rightarrow C_r \equiv \neg t_i \vee C_r$ in (13) requires that the assignment of $t_i = true$ forces the satisfaction of C_r, since the clause has infinite weight. While the clause $(C_r \rightarrow t_i)$ would guarantee that a certain configuration only satisfies those test, which are explicitly assigned. This need not be reasonable in practical applications, since the coverage of additional test may make the configuration instance flexible for further processing, e.g. scheduling. Therefore, the implication of the relaxing variable t_i is omitted for the rest of the work without loss of functionality if such an assumption is required.

To complete the Max-SAT framework a set of soft clauses s_i with weight one is added, featuring the relaxing variables t_i:

$$\Sigma_{soft'} = \{s_1 \wedge s_2 \wedge ... \wedge s_m\} \tag{15}$$

$$s_k = \{(t_k, \omega_{k1}) \wedge (t_k, \omega_{k_2}) \wedge ... \wedge (t_k, \omega_{k_r})\} \tag{16}$$

$$\omega_{k_i} = 1, \quad k = \{1, 2, ..., m\}, i = \{1, 2, ..., r\} \tag{17}$$

The Max-SAT framework for Problem II consists of the hard clauses containing the vehicle configuration constraints Σ_{hard} and the additional constraints featuring the vehicle characteristics Σ_{hard_i}. Each characteristic is assigned a new variable representing its coverage. The additional constraints must be satisfied if and only if the characteristic is satisfied in a configuration. The new variables represented by the soft clauses $\Sigma_{soft'}$ with a finite weight ensure that the Max-SAT optimization satisfies as many characteristics in each configuration as possible:

$$\Sigma_{II} = \Sigma_{hard} \cup \Sigma_{hard_i} \cup \Sigma_{soft'} \tag{18}$$

The greedy approach is applied in the same way as in Problem 5.1, with a change in removing covered test types. While removing the soft clauses relaxing the covered test types suffices to ensure that the greedy algorithm finds a solution set, the relaxing variables t_i will remain as decision variables and enlarge the problem, while not having direct impact on the generated configuration. To avoid this extra computation, for each relaxing variable whose test type has been covered, one can add an additional hard clause forcing the variable to be $false$. While this ensures that the variable will not be considered in future decisions as well as not enforcing the assigned variant to be satisfied, it allows the variant to be covered. If the aim is to ensure that each test type is covered in a single configuration only, one can include the reverse implication $(C_k \rightarrow t_k)$.

If it is desired to enforce to cover the remaining test types and additionally to cover as many variants as possible, this could be established by a change in weights. Each relaxing variable is assigned to a test type which is not covered by any configuration and is assigned a weight $\omega_i = w(m) + 1$. The weight $w(m)$ is any number larger than the sum of all covered variants while covered variants are assigned weight $\omega_i = 1$. This ensures that the Max-SAT framework primarily satisfies the uncovered variants, since their weight is larger than the sum of weights of all covered variants. Additionally the hard clauses enforcing the relaxing variables of the covered variants to be $false$ do not have to be added. During the rest of the paper this is not considered any further.

The Algorithm 2 assumes the input of the Partial Max-SAT instance as described in Equation (18) and the set of soft clauses C. In each iteration C_t represents the clause featuring the additional variable t. The algorithm computes a set of configurations $\mathcal{C}$, such that each configuration covers at least one additional variable, hence one test type.

Algorithm 2.

TestCoverage(Σ_{II}, C)
$C \leftarrow \emptyset$
WHILE ($C \neq \emptyset$)
{
$\mathcal{S} \leftarrow solve(\Sigma_{II})$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{S}\}$
$C \leftarrow C \setminus \{C_t | \mathcal{S}(t) = satisfied\}$
$\Sigma_{II} \leftarrow \Sigma_{II} \wedge \{(\neg t, \infty) | \mathcal{S}(t) = satisfied\}$
}
Return $\mathcal{C}$

Completing Example II in Section 4, the generated hard clauses contain the configurations restrictions as well as the rule sets with the newly generated variables v_1, v_2, v_3 for the respective rule-sets. The soft clauses contain the unit clauses with the newly generated variables, such that the Max-SAT instance in Equation (19) follows:

$$\Sigma_{II} = \{(v_1, 1), (v_2, 1), (v_3, 1) \tag{19}$$

$$(-v_1, A, \infty), (-v_2, L, \infty), (-v_3, D, \infty) \tag{20}$$

$$(E \vee D, \infty), (S, \infty), (H, \infty), (A \vee L, \infty) \tag{21}$$

$$(-D \vee -E, \infty), (-D \vee -A, \infty)\} \tag{22}$$

Within each iteration the algorithm tries to satisfy all unsatisfied hard clauses and the maximal number of new variables. If a conflict with a hard clause occurs, the algorithm may relax one of the new variables and find a solution to cover the rule-sets r_1 and r_2. Variable v_1 and v_2 and then added as negative unit hard clauses satisfying the enforced satisfaction in Equation (20). In the next iteration the algorithm finds a configuration that covers the diesel engine. A possible result for a minimal number of configuration is presented in Table 6.

Table 6. Possible configurations to cover test rule-sets

Features covered	Solution set
r_3	D, S, H, L
r_1, r_2	E, S, H, A, L

6 Computational Experiments

An initial demonstration of the solution approach has been evaluated with an industrial sized problem. The following section presents the demonstrated problem, originating in the automobile industry.

In Table 7 a representative snippet of the data used in the computational experiments is shown. While the snippet represents only a minor part of the configuration rules, each part has additional dependencies leading to a large set of clauses.

The raw data consists of propositional formulas. In order to allow generalized procession, the formulas are converted to CNF using the improved version of the Tseitin transformation introduced by Plaisted and Greenbaum [8]. The transformation into CNF requires the generation of many help variables to avoid exponential growth of clause numbers.

Table 7. Description of cover-able parts

Part ID	Name	Rule-snippet
23308	Mirror_1	$-15179 \wedge -14927 \wedge -10382 \wedge 14520$
23309	Mirror_2	$10382 \wedge 14729 \wedge 15055 \wedge 15097 \wedge 15179$
23310	Locking System	$(-10380 \wedge 15073) \vee (-14074 \wedge 15073)$
23311	Door interior	$-17462 \wedge -14520 \wedge 14923$
23312	Switch Block	$(-23312 \vee -10380) \wedge 15179 \wedge -14074$
23314	Steering wheel	$-23333 \wedge -23332 \wedge 15569 \wedge 23337$
23322	Pedal systems	$-9452 \wedge 9485 \wedge 13965 \wedge 13970 \wedge 14959$
23323	Host assembly	$(14565 \vee 15205) \wedge -13953$

The data in raw data snippet 1 shows some of the CNF clause related to the parts in Table 7 and their rule sets using the DIMACS CNF format. Each lines represents a clause, such that the literals are disjunctive. The trailing zero marks the end of a clause. The large number "2147483647" is the signed integer maximal value and represents the infinite weight of the hard clauses.

Raw data snippet 1.

```
2147483647 −17462 −14520 14923 23311 0
2147483647 −15179 −14927 −14074 10380 23312 0
2147483647 −23332 14729 23314 23333 0
2147483647 −23332 15471 23314 23333 0
2147483647 −23334 −23333 −23332 15099 23314 23335 0
2147483647 −23343 −23342 15224 23317 23344 0
2147483647 −23343 −23342 15392 23317 23344 0
2147483647 −23343 −23342 14642 23317 23344 0
2147483647 −15084 −14954 −14729 23317 23342 0
2147483647 −15084 −14954 −14642 23317 23342 0
2147483647 −23349 14641 23319 23350 0
2147483647 −23349 15222 23319 23350 0
2147483647 −23349 10380 23319 23350 0
2147483647 −23349 15392 23319 23350 0
[...]
```

The problems features 28413 configuration constraints as hard clauses which have to be satisfied. The size of the universe U, including all available parts, is 23355. The test-setup for the approach of Problem I consists of 40 components, hence soft clauses, which need to be covered in configurations. The approach for Problem II was tested on the same configuration constraints with an additional 17 rule-sets resulting in 28430 hard clauses and 17 new soft clauses representing the introduces test-variables. The results, as in Table 8, were calculated in less than a second.

The computations were run on a Intel(R) Core(TM) i7-4800MQ CPU with 2.7 GHz and 2 GB of main memory running 64 Bit Arch Linux. The solution approach is implemented around the open-source Max-SAT solver open-wbo [7]. The Max-SAT solver is utilized as a black box to simulate the generation of a valid configuration.

Table 8. Computational scenarios

Experiment	Coverage	#Configurations
PartsCoverage	40 parts	3
RuleCoverage	17 rules	3

7 Conclusion

Large sets of components and interfaces included in vehicle configurations may lead to impracticable complexity when computing vehicle configurations. Special requirements can result in unnecessary development cost. Due to the vast number of possible configurations it may not be feasible to choose an optimal configuration set to conduct component and attribute tests. It is however, possible to describe the set of valid configurations and respectively tests with an abstract model represented by their constraints and inter-dependencies. For different use cases it may be desirable to cover a predefined set of components or a defined set of attributes, each being identified by a rule-set.

The presented Max-SAT framework builds upon the Boolean constraint product configuration used in previous studies. The greedy like approximation framework can compute near to optimal sets of configurations to cover both component tests as well as attributes test setups. Initial results demonstrate that the framework performed well on industrial sized problems. The solution provided can be implemented with an arbitrary Max-SAT solver as black box optimization function. Future work may include the integration of different optimization models and heuristics as a substitution for the Max-SAT solver.

REFERENCES

[1] A. Biere, M. Heule, and H. van Maaren, *Handbook of Satisfiability: Handbook of Satisfiability*, volume v.185 of *Frontiers in artificial intelligence and applications*, IOS Press, Amsterdam, 2009.

[2] Kenneth Chelst, John Kuechlin, WolfgangSidelko, Alex Przebienda, Jeffrey Lockledge, and Dimitrios Mihailidis, 'Rightsizing and Management of Prototype Vehicle Testing at Ford Motor Company', *Interfaces*, **31**(1), 91–107, (2001).

[3] Uwe Clausen and Jörg Weber, 'Prototypenplanung im nutzfahrzeugbau', *ATZ - Automobiltechnische Zeitschrift*, **108**(9), 740–744, (2006).

[4] Uriel Feige, 'A threshold of ln n for approximating set cover', *J. ACM*, **45**(4), 634–652, (July 1998).

[5] Andreas Kübler, Christoph Zengler, and Wolfgang Küchlin, 'Model counting in product configuration', in *Proceedings First International Workshop on Logics for Component Configuration, LoCoCo 2010, Edinburgh, UK, 10th July 2010.*, eds., Inês Lynce and Ralf Treinen, volume 29 of *EPTCS*, pp. 44–53, (2010).

[6] Kamol Limtanyakul, *Scheduling of tests on vehicle prototypes*, Ph.D. dissertation, Dortmund University of Technology, 2009.

[7] Ruben Martins, Vasco Manquinho, and Inês Lynce, *Theory and Applications of Satisfiability Testing – SAT 2014: 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, chapter wbo WBO: A Modular MaxSAT Solver,, 438–445, Springer International Publishing, Cham, 2014.

[8] David A. Plaisted and Steven Greenbaum, 'A structure-preserving clause form translation', *J. Symb. Comput.*, **2**(3), 293–304, (September 1986).

[9] Tilak Raj Singh and Narayan Rangaraj, 'Generation of predictive configurations for production planning', *15 th International Configuration Workshop*, 79, (2013).

[10] Tilak Raj Singh and Narayan Rangaraj, 'Optimization based framework for transforming automotive configurations for production planning.', in *Configuration Workshop*, pp. 31–38, (2014).

[11] Carsten Sinz, *Baubarkeitsprüfung von Kraftfahrzeugen durch automatisches Beweisen*, Diplomarbeit, Universität Tübingen, December 1997.
[12] Petr Slavík, 'A tight analysis of the greedy algorithm for set cover', in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 435–441, New York, NY, USA, (1996). ACM.
[13] Rouven Walter, Thore Kübart, and Wolfgang Küchlin, *Optimal Coverage in Automotive Configuration*, 611–626, Springer International Publishing, Cham, 2016.
[14] Rouven Walter and Wolfgang Küchlin, 'Remax – a maxsat aided product (re-) configurator', in *Proceedings of the 16th International Configuration Workshop, CEUR Workshop Proceedings*, volume 1220, pp. 59–66. CEUR Workshop Proceedings, (2014).