Towards Recommending Configurable Offerings

Tiihonen Juha [1] and **Felfernig Alexander** [2]

Abstract. Configuration technologies provide a solid basis for the implementation of a Mass Customization strategy. A side-effect of this strategy is that the offering of highly variant products and services triggers the phenomenon of Mass Confusion, i.e., customers are overwhelmed by the size and complexity of the offered assortments. In this context, recommendation technologies can provide help by supporting users in the identification of products and services fitting their wishes and needs. Recommendation technologies have been intensively exploited for the recommendation of simple products such as books or movies but have (with a few exceptions) not been applied to the recommendation of complex products and services such as computers or financial services. In this paper we show how to integrate case-based and content-based recommendation approaches with knowledge-based configurators.

1 Introduction

In many domains, customers do not know and understand in detail the complete set of options supported by a given configurable product. On the one hand configuration options are represented on a rather technical level and customers are overwhelmed by the offered set of alternatives [6] - this phenomenon is well known as Mass Confusion [14]. On the other hand customers do not know their preferences beforehand since preferences are typically constructed [13] within the scope of a configuration session. Even experienced sales persons tend to propose configurations they are used to thus overlooking configuration alternatives which better suit to the customers' wishes and needs. This can cause unsatisfied customers as well as the sales of less profitable configurations. Consequently, users of configuration systems are in the need of more intuitive interaction mechansisms effectively supporting the configuration and selection of interesting product and service alternatives. With a few exceptions [5, 12, 16] existing recommender technologies [8] are primarily applied for the recommendation of simple products and services such as books, movies or compact discs. These technologies have not been integrated into configuration environments dealing with complex products and services. The major goal of this paper is to discuss scenarios in which recommendation technologies can be exploited in configuration sessions. This work has been conducted within the scope of the COSMOS project[3].

[1] Deptartment of Computer Science and Engineering, Helsinki U of Technology
[2] Intelligent Systems and Business Informatics, U Klagenfurt
[3] COSMOS (Customer-Oriented Systematically Managed Service Offerings) is a project supported by the Finnish Funding Agency for Technology and Innovation.

Recommendation Technologies. *Collaborative filtering* (see, e.g., [1]) is one of the most commonly used recommendation technologies. It provides recommendations on the basis of opinions of users (e.g., ratings or purchasing data). A similarity function on opinions about items is exploited to calculate nearest neighbors which are users with similar preference structures. The basic idea is to identify similar users and to recommend their highly rated items that are unknown to the active user. In many B2C scenarios an individual user may purchase too few configurable products to establish a dense enough user profile to be suitable as a basis for pure collaborative filtering. However, collaborative filtering technologies can be used in very specific settings supporting the collaborative development of *product innovations* [10]. This issue will not be further discussed in this paper.

Content based filtering (see, e.g., [15]) approaches recommend items similar to those that the active user has preferred in the past. Items are described by a number of keywords or features. A user model contains previous opinions about items, often presented as keywords or features. A similarity function is used to calculate nearest neighbors [4] which are in this case those items with the highest similarity compared to the given preference information in the user profile. The approach is typically applied for recommending text-based items such as articles or web pages. One challenge of applying this technique to configurable offerings is that - beside the availability of textual component descriptions - building a user profile requires repetitive configurations of one user. Furthermore, a profile may soon become outdated in rapidly evolving domains such as PC's. Beside the calculation of the k-nearest neighbors predictions [4] on interesting items, recommendations can be based on naive Bayes predictors [15] which allow the prediction of interesting items on the basis of their probability of being selected given the existing user preferences. *In this paper we will focus on the application of naive Bayes predictors for the identification of interesting configurations.*

Utility based recommendation estimates the utility of an item for a customer and recommends items with the highest utility. Domain-specific interest dimensions have to be identified in this context. For PC's, interest dimensions could be *economy*, *reliability*, *graphics performance* and *weight*. Items are given numeric utility values with respect to the interest dimensions. The user specifies his preferences in terms of importance (weight) of each interest dimension. Given this information, item utilities can be computed for the active user. An example for the application of utility-based recommendation approaches in the financial services domain is given in [9]. A further discussion of utility-based approaches in the configuration context is outside the scope of this paper.

Proceedings of the ECAI 2008 Workshop on Configuration Systems
pp. 29-34, July 21 – 22, 2008, Patras, Greece.
Proceedings: http://www.soberit.hut.fi/configws08/
pages/WorkshopOnConfigurationSystemsECAI2008W12.pdf

Knowledge-based recommenders exploit explicit information about items and explicit knowledge on how user requirements on those items can be specified [4][9]. *Constraint-based recommendation* is a knowledge-based approach where alternative items and potential customer requirements are described on the basis of a set of features and the corresponding constraints. Filter constraints match customer requirements to suitable items. Compatibility constraints ensure the consistency of requirements. To resolve inconsistencies, explanation and repair functionalities are provided [9]. Constraint-based approaches exploit the same technologies as many knowledge-based configuration environments and are additionally combined with, e.g., utility-based recommendation approaches supporting the ranking of candidate configurations. *Case-based recommendation* [3] is another type of knowledge-based recommendation. It exploits similarity functions to determine items fitting to the wishes an needs of users. In contrast to content-based filtering and collaborative filtering those similarity functions compare elementary properties of items (e.g., PC price) rather than extracted keywords or categories. Our major goal in this paper is to apply and extend different concepts of case-based and content-based recommendation in order to make them applicable in configuration settings.

Recommendation Scenarios. Recommendation of configurable offerings can focus on

- selecting a suitable base product line to configure (such as a car model)
- recommending a complete configuration (such as a complete PC for gaming or a tractor for peat harvesting including suitable wheels, air-intake filters, and other equipment)
- recommending how to complete a configuration (e.g., to propose still unspecified details of a PC)
- recommending a subconfiguration (e.g., a storage subsystem suitable for a particular type of use such as a PC storage subsystem for full-HD video-editing and authoring)
- recommending individual attribute or component settings (e.g., a mobile data connection for a business person)

Consequently, a high diversity of usage and integration scenarios for recommendation technologies in the configuration context can be envisioned. In this paper our major focus is the integration of case-based and content-based recommendation into existing configurators. We analyze existing approaches in the field [5, 12, 16] and show how to extend and improve those approaches by developing a more sophisticated notion of *similarity between item features* and *explicitly taking into account customer importance weights* for features.

The remainder of the paper is organized as follows. In the following section we introduce a working example from the domain of configurable computers. In Section 3 we discuss relevant case-based and content-based algorithms for configurable products and services and introduce our extensions to those algorithms. A discussion of related and future work is presented in Section 4. Section 5 concludes the paper.

2 Working Example

In this paper we consider (for reasons of simplicity) only "flat" configuration models consisting of features (no variation of structure or connections), each having a finite domain of possible values. Our example product is a PC, that has as features a motherboard (mb), a hard disk (hd), an optical drive (od), a processor (pr), and optionally a graphics card (gc). The amount of memory (me) is specified in gigabytes (1, 2, 3, or 4). A *complete configuration* specifies a value for each feature. Furthermore, a *valid configuration* is complete and consistent with a defined set of constraints.

We represent some non-configurable attribute values related to features to specify constraints more intuitively. Processors are introduced in Table 1(a). Processor performance is approximated with an industry standard benchmark, specified by $CScr$. *Socket* determines a processor's connection to a motherboard. Motherboards (see Table 1(b)) are designed to be compatible with either manufacturer A's or I's processors, socket 'a' or 'i', respectively. Thus, a specific constraint specifies that a processor must fit the motherboard: $pr.socket = mb.socket$. In addition, some motherboards provide an integrated graphics card ($IntGr = yes$).

pr	socket	CScr
as	a	1250
i4	i	2858
i9	i	4537

mb	socket	IntGr	GScr
a1	a	yes	300
a2	a	no	0
i1	i	yes	200
i2	i	no	0

Table 1. Processors (a, left) and motherboards (b, right)

Separate graphics cards provide higher performance than those integrated to motherboards. Graphics performance is approximated with an industry standard benchmark, represented by graphics performance score ($GScr$). Similarly, motherboards with integrated graphics card specify their graphics performance with $GScr$. Graphics cards $g2$, $g8$, and $g9$ have graphics performance scores 2800, 2200, and 5500, respectively. A system must always have a way to produce graphics. Thus the constraint $(mb.IntGr = no) \implies (gc \neq no)$ is introduced. pc.GScr refers to graphics performance of the PC, determined as the maximum $GScr$ provided by the graphics card or the motherboard.

Hard disks (hd) are available in different capacities (GB) ($h2.capacity = 250$, $h5.capacity = 500$, and $h9.capacity = 1000$). All optical drives (see Table 2) read CD and DVD. Some write DVD or DVD + Blu-ray.

od	Write DVD & CD (dw)	Read Blu-ray (br)	Write Blu-ray (bw)
dr	no	no	no
dw	yes	no	no
br	no	yes	no
bw	yes	yes	yes

Table 2. Properties of optical drives of the running example

Three additional features, namely video editing (vi), photos (ph), and gaming (ga) are included in the configuration model to describe intended use of the PC being configured. Details are specified in Table 3.

The following domain knowledge is available:
$ph \neq no \implies od.dw = yes$: to archive photos
$ph = adv \implies hd.capacity \geq 500$: disk space for photos
$ph = adv \implies pr.CScr \geq 2500$: CPU for advanced photo

Feature	Values		
Video editing (**vi**)	no (*no*)	standard definition (*sd*)	high-definition (*hd*)
Photos (**ph**)	no (*no*)	normal home use (*std*)	advanced amateur or professional (*adv*)
Gaming (**ga**)	no or 2D games (*2d*)	3d games (*3d*)	enthusiast performance 3d games, HD resolutions (*adv*)

Table 3. Intended usage features of the running example

$ph = adv \implies me \geq 2$: RAM for advanced image processing

$vi = sd \implies pr.CScr \geq 2700$: CPU for SD video editing

$vi = hd \implies pr.CScr \geq 4500$: CPU for HD video editing

$vi = sd \implies od.dw = yes$: burn DVD videos

$vi = hd \implies od.bw = yes$: burn Blu-ray videos

$ga = 3d \implies pr.CScr \geq 1500$: CPU for 3D gaming

$ga = 3d \implies pc.GScr \geq 1500$: graphics for 3D gaming

$ga = adv \implies pr.CScr \geq 2800$: CPU for advanced gaming

$ga = adv \implies pc.GScr \geq 5000$: graphics for advanced gaming

For the formulas discussed in this paper, we assume the following distribution of *feature importance weights*: video editing $w(vi) = 5\%$, photos $w(ph) = 5\%$, gaming $w(ga) = 9\%$, processor $w(pr) = 18\%$, motherboard $w(mb) = 5\%$, amount of memory $w(me) = 15\%$, hard disk $w(hd) = 16\%$, graphics card $w(gc) = 17\%$, optical drive $w(od)$=10%. Those features could stem from direct customer specifications, representative preferences from statistical samples, or the application of utility constraints as documented in [9].

Notation. We base the discussion of recommendation algorithms on the following conventions. Relation $Conf$ holds previous K configurations, each having values for the existing N features $f_1, .., f_N$. The value of feature f_i in configuration k is referred to as $f_{i,k}$. The k^{th}configuration is referred to as $Conf_k$. Classification (discussed in Section 3.1) of configuration k is referred to as c_k. When referring to the profile of the active user, we use index u, $u \notin K$, e.g., $f_{i,u}$ refers to the value of feature i for the active user. The set of specified features in the active user profile is F_u. $F_u = \{f_j | f_{j,u} \neq noval\}$, and the set of features for which the active user profile does not have a value is $\bar{F}_u$. Furthermore, a projection $\pi_{F_u}(Conf)$ of previous configurations for which the profile has (does not have) values is referred to as $Conf_{F_u}$ ($Conf_{\bar{F}_u}$). The index of a configuration (k) is considered to be included in $Conf_k$ and projections, see Table 4, to avoid unintended removal of duplicate tuples. Finally $dom(f_j)$ returns the domain of f_j.

k	f_1 **vi**	f_2 **ph**	f_3 **ga**	f_4 **pr**	f_5 **mb**	f_6 **me**	f_7 **hd**	f_8 **gc**	f_9 **od**	c
1	no	no	2d	as	a1	1	h2	no	dr	ba
2	no	std	2d	as	a2	1	h5	g2	dw	st
3	sd	std	adv	i4	i2	3	h5	g9	dw	ad
4	hd	adv	adv	i9	i2	4	h9	g9	bw	ad
5	sd	adv	3d	i4	i1	2	h9	g8	dw	st
u	no	no	3d							

Table 4. Previous configurations and active user profile

3 Recommendation Algorithms

3.1 Distance Metrics

The algorithms discussed in this paper use distance functions to determine similarity or dissimilarity of individual feature values, and ultimately that of configurations. The motivation for using distance functions instead of equality when comparing feature values is that equality may be too strict for a measure - close values or configurations could remain ignored.

We decided to include ideas of the *Heterogeneous Value Difference Metric* (*HVDM*) [17] to cope with symbolic (nominal) and numeric features in a relatively simple manner. On the feature level the distance is defined as follows where the function $d_{f_i}(x, y)$ returns the distance between values x and y of feature a, using a different sub-function for different types of features: distances between symbolic feature values are computed by function $vdm_{f_i}(x, y)$, and those between linear values by function $diff_{f_i}(x, y)$. Based on experiments of [17], these functions provide similar influence on the overall distance measurements. Distance values returned by $d_{f_i}(x, y)$ are *normalized* to usually (but not always) be in range 0 to 1.

$$d_{f_i}(x,y) = \begin{cases} 1 & \text{if x or y is unknown; otherwise} \\ vdm_{f_i}(x,y), & \text{if } f_i \text{ is symbolic} \\ diff_{f_i}(x,y), & \text{if } f_i \text{ is linear} \end{cases}$$

The function $vdm_{f_i}(x, y)$ learns the similarity of symbolic values in a domain automatically. This takes place by examining the probability that individual feature values contribute to the classification of the sample vector - in our case classification of configurations. Slightly oversimplifying, the closer the probability of a pair of feature values to be present in identically classified configurations, the more similar these feature values are considered. In this paper we take a simplistic view, and consider a configuration to belong to one of three clusters (basic (*ba*), standard (*st*), or advanced (*ad*)), which is used as the classifier for *HVDM* (see column "c" in Table 4).

$$\begin{aligned} vdm_{f_i}(x,y) &= \sqrt{\sum_{c=1}^{C} \left| \frac{N_{f_i,x,c}}{N_{f_i,x}} - \frac{N_{f_i,y,c}}{N_{f_i,y}} \right|^2} \\ &= \sqrt{\sum_{c=1}^{C} |P_{f_i,x,c} - P_{f_i,y,c}|^2} \end{aligned}$$

In the function $vdm_{f_i}(x, y)$, $N_{f_i,x}$ is the number of instances (configurations) in the training set T that have value x for feature f_i; $N_{f_i,x,c}$ is the number of instances in T that have value x for feature f_i and output class c; C is the number of output classes in the problem domain (in our case 3 - see Table 4). $P_{f_i,x,c}$ is the conditional probability of output class c given that feature f_i has the value x, i.e., $P(c|f_i = x)$. When $N_{f_i,x} = 0$, $P(c|f_i = x)$ is considered 0.

For example, in our example population $Conf$, $N_{pr,as} = 2$, $N_{pr,i4} = 2$, and $N_{pr,i9} = 1$. The classification frequencies for processor $N_{pr,x,c}$ are presented on left half of Table 5, e.g., feature value *as* for classification basic (*ba*) occurs exactly once. The resulting distance matrix for processors is presented on the right half of Table 5.

Distances between linear values x and y of feature f_i (in $Conf$) are determined by function $diff_{f_i}(x, y)$. Since 95% of the values in a normal distribution fall within two standard deviations of the mean, the difference between numeric values

c,x	as	i4	i9
ba	1	0	0
std	1	1	0
adv	0	1	1

	as	i4	i9
as	0	0.707	1.225
i4	0.707	0	0.707
i9	1.225	0.707	0

Table 5. Classification frequencies of pr (left), and corresponding distance matrix (right)

is divided by 4 standard deviations to scale each value into a range that is usually (95% of cases) of width 1. As motivated in [17] "distances are often normalized by dividing the distance for each variable by the range of that attribute, so that the distance for each input variable is in the range 0..1. However, dividing by the range allows outliers (extreme values) to have a profound effect on the contribution of an attribute. For example, if a variable has values which are in the range 0..10 in almost every case but with one exceptional (and possibly erroneous) value of 50, then dividing by the range would almost always result in a value less than 0.2."

$$diff_{f_i}(x,y) = \frac{|x-y|}{4\sigma_{f_i}}$$

The distance metric $d_{f_i}(x,y)$ usually returns a value from 0 to 1. The similarity $sim_{f_i}(x,y)$ between two feature values x and y of feature f_i can then be defined as $sim_{f_i}(x,y) = 1 - d_{f_i}(x,y)$. In the following subsections we show how these metrics can be applied to the recommendation of feature values as well as to the recommendation of complete configurations. In the sense of case-based configuration we investigate existing (and similar) configurations in order to predict interesting feature settings and complete solutions. Note that our cases describe user preferences as well as technical features.

Following this approach, we extend existing algorithms [5] to take into account *similarity* of feature values, not just direct equality. Furthermore, we extend previous approaches to take into account the *weights* of different features.

3.2 Nearest Neighbor

The idea of a *nearest neighbor* is simple: determine a neighbor configuration, which is closest to the known parts of active user's profile, and recommend feature values of this nearest neighbor. The nearest neighbor is determined as follows (distance between two configurations):

$$dist(Conf_u, Conf_a) = \sum_{i \in F_u} d_{f_i}(f_{i,u}, f_{i,a}) * w(f_i)$$

For nearest neighbor c the following has to hold: $\nexists c', 1 \leq c' \leq K : dist(Conf_u, Conf_{c'}) < dist(Conf_u, Conf_c)$.

In our example, the nearest neighbor relative to our user profile is $Conf_1$: $d_{vi}(no,no)$= 0.000, $w(vi)$=0.050; $d_{ph}(no,no)$ = 0.000, $w(ph)$=0.050; $d_{ga}(3d,2d)$ = 0.707, $w(ga)$=0.090. Total weighted distance $dist(Conf_u, Conf_1)$ = 0.064. Applying the nearest neighbor formula to configurations $Conf_2$ .. $Conf_5$ provides distances 0.125, 0.224, 0.250, and 0.097. Unfortunately, the combination of known feature values of the user profile and $Conf_1$ is not consistent (graphics and cpu performances are not sufficient for 3d gaming). Feature values of the nearest consistent neighbor $Conf_5$ are recommended: $pr = i4$, $mb = i1$, $me = 2$, $hd = h9$, $gc = g8$, $od = dw$.

3.3 Weighted Majority Voter

The *weighted majority voter* [5] recommends individual feature values based on each neighbor configuration in $Conf$ "voting" for its feature values. The weight of each neighbor vote is determined by the number of equal feature values to the user profile. For example, the weight of $Conf_1$ for user u is 2, because $f_{vi,1} = f_{vi,u} = no$, and $f_{ph,1} = f_{ph,u} = no$. Thus, $Conf_1$ would give 2 votes for $pr = as$, $mb = a1$, $me = 1$, $hd = h2$, $gc = no$, and $od = dr$. The following feature values get most votes: $pr = as$ (3 votes), $mb = a1$ (2), $me = 1$ (3), $hd = h2$ (2), $gc = no$ (2), and $od = dr$ (2).

The consistency of a potential recommendation is checked by adding the proposed value to the known values in the user profile. After user selects a feature value, recommendations will be recalculated to reflect the new situation. Due to consistency checks, $pr = i4$ (1 vote) and $gc = g2$ (1 vote) replace those with most votes, assuming the selection of the first feature value in a domain in case of a tie in votes.

Next, we rewrite the formula of [5] and propose the corresponding extensions. First, we define the equality function eq to return 1 when two values are equal, otherwise 0.

$$eq(x,y) = \begin{cases} 1 & \text{if x=y} \\ 0 & \text{otherwise} \end{cases}$$

The weight $w(conf_x, conf_u)$ of a neighbor configuration $conf_x$ with respect to configuration $conf_u$ (a user's partial configuration) is the number of equal feature values in features for which $conf_u$ has a value (F_u):

$$w(conf_x, conf_u) = \sum_{i \in F_u} eq(f_{i,x}, f_{i,u})$$

The prediction score $pr(conf_u, f_j, v)$ for (user's configuration) $conf_u$ having value v for feature f_j is thus sum of weights (votes) of neighbors having value v for feature f_j:

$$pr(conf_u, f_j, v) = \sum_{i=1}^{K} eq(f_{j,i}, v) * w(conf_i, conf_u)$$

A value v with maximum prediction score $pr(conf_u, f_j, v)$ is the recommendation $r(f_j, conf_u)$ for feature f_j in configuration $conf_u$:

$$r(f_j, conf_u) = v \text{ such that}$$
$$\nexists v', v' \in dom(f_j) : pr(conf_u, f_j, v') > pr(conf_u, f_j, v)$$

We propose a modified algorithm to derive neighbor weights. Neighbor weights are determined by the similarity of neighbor and user profile feature values instead of equality used by [5]. Thus, similar values compared to user's existing selections contribute to the weight of a neighbor. Further, we take into account the importance of individual features for a user (feature weights). Both of those aspects are extremely important for interactive settings. Thus, the weight $w(conf_x, conf_u)$ of a neighbor configuration $conf_x$ with respect to configuration $conf_u$ is defined as follows:

$$w(conf_x, conf_u) = \sum_{i \in F_u} sim_{f_i}(f_{i,x}, f_{i,u}) * w(f_i)$$

The weights of example neighbors are $w(conf_1, conf_u) = 0.126$, $w(conf_2, conf_u) = 0,065$, $w(conf_3, conf_u) = -0.034$, $w(conf_4, conf_u) = -0.060$, and $w(conf_5, conf_u) = 0.093$. Using these weights, the first potential recommendations are $pr = as$ (0.191), $mb = a1$ (0.126), $me = 1$ (0.191), $hd = h2$ (0.126), $gc = no$ (0.126), and $od = dr$ (0.126). To provide (locally) consistent recommendations, pr and gc must be substituted with $pr = i4$ (0.060), and $gc = g8$ (0.093).

3.4 Most Popular Choice

The most popular choice algorithm [5] recommends entire remaining configurations, typically to complete a configuration. The probability estimate for a configuration $c \in Conf$ with respect to known features in user's profile F_u (and corresponding unknown features $\bar{F}_u$) is calculated as:

$$Pr(c, u, F_u) = Pr_{basic}(c, \bar{F}_u) * \prod_{j \in F_u} Pr(f_{j,u} = f_{j,u}|Conf)$$

In the original formula, basic probability $Pr_{basic}(c, \bar{F}_u)$ of a neighbor configuration c is based on the popularity of it's feature values on features for which the user profile does not have a value, $\bar{F}_u$. The basic probability for feature f_j having the value that configuration c has ($f_{j,c}$) is simply the proportion of neighbors having that value for feature f_j. Basic probability of a configuration is determined by multiplying the basic probabilities for its feature values. We apply function $count(f_j, v)$ that returns the number of neighbors in $Conf$ having value v for feature f_j: $count(f_j, v) = \sum_{k=1}^{K} eq(f_{j,k}, v)$.

$$Pr_{basic}(c, \bar{F}_u) = \prod_{j \in \bar{F}_u} \frac{count(f_j, f_{j,c})}{K}$$

We extend the concept of basic probability by giving each feature value support when neighbour configurations have feature values within maximum distance Δ, ($d_{f_j}(f_{j,u}, f_{j,a}) \leq \Delta$). The support is defined as term $(1 - d_{f_j}(f_{j,u}, f_{j,a}))^2$ to quickly lessen its significance when the distance increases. We define support $s_{f_j}(x, y)$

$$s_{f_j}(x, y) = \begin{cases} (1 - d_{f_j}(x, y))^2, & \text{if } d_{f_j}(x, y) \leq \Delta \\ 0, & \text{otherwise} \end{cases}$$

Maintaining these as probabilities requires that the sum of probabilities of existing values is 1. Thus, the sum of supports for feature f_j having the value that configuration c has ($f_{j,c}$) is divided by the sum of supports given to all values in domain of f_j that exist in at least one neighbor configuration. Therefore

$$Pr_{basic}(c, \bar{F}_u) = \prod_{j \in \bar{F}_u} \frac{\sum_{k=1}^{K} s_{f_j}(f_{j,c}, f_{j,k})}{\sum_{v \in dom(f_j)} \sum_{k=1}^{K} s_{f_j}(v, f_{j,k}) * min(1, count(f_j, v))}$$

The basic probability is weighted with a Bayesian predictor for the user profile u to have the values already selected, given the existing neighbors, $\prod_{j \in F_u} P(f_{j,u} = f_{j,u}|Conf)$. $P(f_{j,u} = f_{j,u}|Conf)$, is defined to be an m-estimate [2] to stabilize probability calculations even in case of (too) few samples. The m-estimate assumes m virtual samples with initial probability p that augment the estimation of probability. $m_{est}(N_c, N, p, m) = \frac{N_c + mp}{N + m}$. We apply the following m-estimate parameters: 1) the number of "in-samples" N_c is the number of such neighbor configurations that have equal value (f_j, u) as the user profile for feature f_j being inspected, and that have the same configuration with respect to features ($\bar{F}_u$) that the user profile does not have a value; 2) the number of all samples N is the number of such neighbor configurations that have the same configuration with respect to those features ($\bar{F}_u$) that the user profile does not have a value; and 3) m-estimate virtual sample parameters are $p = 1/K$, and $m = K$.

$$\prod_{f_j \in F} P(f_{j,u} = f_{j,u}|Conf) =$$

$$\prod_{f_j \in F} m_{est}(eqcfgs_m(c, F \cup f_j, f_j, f_{j,u}), eqcfgs(c, F), 1/K, K)$$

$eqcfg(i, j, F)$ tests if neighbor configurations i and j are equal with respect to a set of features F. It returns 1 iff profiles i and j have equal feature values for all features $f \in F$. Otherwise it returns 0.

$$eqcfg(i, j, F) = \begin{cases} 1 & \text{if } \forall f \in F : eq(f_{f,i}, f_{f,j}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$eqcfgs(c, F)$ returns the number of neighbor configurations that are equal to configuration c with respect to a set of features F. $eqcfgs(c, F) = \sum_{k=1}^{K} eqcfg(c, k, F)$.

$eqcfgs_m(c, F, f_j, v)$ returns the number of neighbor configurations that are equal to configuration c with respect to a set of features F, and which have value v for feature f_j. $eqcfgs_m(c, F, f_j, v) = \sum_{k=1}^{K} eqcfg(c, k, F) * eq(f_{j,k}, v)$.

In our example, the basic probability of $conf_5$ with the original formula [5] is 0.003072 = 0.4 * 0.2 * 0.2 * 0.4 * 0.2 * 0.6, because $f_{pr} = i4$ two times, other terms for basic probability are 0.2 (mb), 0.2 (me), 0.4 (hd), 0.2 (gc), and 0.6 (od). With our modified formula and (a large) Δ= 0.8 they are: 0.403 (pr), 0.286 (mb), 0.280(me), 0.444 (hd), 0.286 (gc) 0,600 (od) = 0,00246

$\prod_{j \in F_u} P(f_{j,u} = f_{j,u}|Conf) = m_{est}(1, 1, 0.2, 5) * m_{est}(0, 1, 0.2, 5) * m_{est}(0, 1, 0.2, 5)$. For example, for the first term $N_c = 1$, because configuration $conf_5$ has $vi = sd$ just as the user profile, and there are no other equal configurations to $conf_5$ with respect to features absent from user profile. The second term for ph has $N_c = 0$, because $conf_5$ has different value for ph than the user profile u. $conf_5$ becomes the configuration of choice (with estimate 1.12E-07, thus its value are recommended: $pr = i4$, $mb = i1$, $me = 2$, $hd = h9$, $gc = g8$, and $od = dw$.

4 Related and Future Work

The paper of [5] presents a recommender implementation for on-line PC configuration and underlying recommendation algorithms. We extended the recommendation algorithms of [5]

in order to be able to take into account the aspects of importance weights and similarity (which substitutes the notion of equality). Furthermore, we take into account the notion of consistency, i.e., we only allow recommendations which are consistent with the given set of customer requirements. Admittedly, the evaluation of our approach has not been completed up to now but will be a strong focus of future work.

The contribution of [12] presents an approach to integrate case-based reasoning with constraint solving with the goal to adapt identified nearest neighbors to the new configuration problem. The used algorithm for calculating nearest neighbors takes into account component structures but does not take into account probabilities of selection. The authors then discuss an approach to the calculation of adaptations for the identified nearest neighbors in order to conform with the new customer preferences. No details are provided regarding the minimality of changes or how the adaption effects the given customer requirements. One of our major goals for future research is to integrate mechanisms which allow the calculation of minimal adaptations in the case of recommendations partially inconsistent with the given customer requirements.

[16] present an approach to the application of case-based reasoning for product configuration tasks. The authors develop their approach on the basis of a high-level product structure where instance similarities are determined on the basis of simple equality relations. Compared to the approach presented in this paper, the authors do not take into account propabilities which provide an indication of the most promising similar cases.

Management of consistency of recommendations with respect to an existing (partial) configuration is an important topic. In cases where there exist interesting configurations for customers but those are incompatible with the initial set of requirements, corresponding explanations have to be provided. Vice versa, such explanations should take into account minimal distances to existing nearest neighbours. The developments in this area will rely on existing work related to the determination of explanations [11, 7].

We proposed to apply a similarity metric that automatically determines the similarity of feature values based on classification outcomes of configurations. This approach has potential to improve the quality of identified cases and feature settings. However, efforts have to be invested to evaluate the proposed metrics within the scope of user studies. An interesting open question in this context is whether classifiers should be based on the whole configurable product, or should, e.g., sub-system-specific classifiers be applied to provide more accurate similarity metrics.

Reconfiguration of products and services could benefit from personalized recommendation support. For example, in insurance and financial domains situation or needs of individuals or customer organizations change within long relationships of customership. It should be possible to update the configured solution correspondingly while avoiding solutions that introduce sub-optimal switching costs or weakening of current contractual terms. These are open questions which are within the focus of the COSMOS project.

5 Conclusions

In this paper we have shown the potential benefitss of integrating case-based/content-based recommendation with configuration technologies. This integration allows for the derivation of individualized and personalized product and service offerings. Those technologies show great potential for reducing the so-called mass confusion phenomenon which prevents users from identifying products and services fitting their wishes and needs. The recommendation approach presented in this paper is a first but important step towards personalized configuration systems which more actively support users in preference construction processes.

References

[1] G. Adomavicius and A. Tuzhilin, 'Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions', *Knowledge and Data Engineering, IEEE Transactions on*, **17**(6), 734–749, (2005).

[2] I. Bratko, B. Cestnik, and I. Kononenko, 'Attribute-based learning', *AI Communications*, **9**(1), 27–32, (1996).

[3] R. Burke, 'Knowledge-based Recommender Systems', *Encyclopedia of Library and Information Systems*, **69**(32), (2000).

[4] R. Burke, 'Hybrid Recommender Systems: Survey and Experiments', *User Modeling and User-Adapted Interaction*, **12**(4), 331–370, (2002).

[5] R. Coester, A. Gustavsson, R. Olsson, and A. Rudstroem, 'Enhancing web-based configuration with recommendations and cluster-based help', in *AH'02 Worksh. on Recommendation and Personalization in EComm.*, Malaga, Spain, (2002).

[6] W. Emde, C. Beilken, J. Boerding, W. Orth, U. Ptersen, J. Rahmer, M. SPenke, A. Voss, and S. Wrobel, 'Configuration of Telecommunication Systems in KIKon', in *Workshop on Configuration*, pp. 105–110, Stanford, California, (1996).

[7] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based Diagnosis of Configuration Knowledge Bases', *Artificial Intelligence*, **2**(152), 213–234, (2004).

[8] A. Felfernig, G. Friedrich, and L. Schmidt-Thieme, 'Recommender systems', *IEEE Intelligent Systems-Special Issue on Recommender Systems*, **22**(3), (2007).

[9] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, 'The vita financial services sales support environment', in *AAAI*, pp. 1692–1699, (2007).

[10] N. Franke, P. Keinz, and M. Schreier, 'Complementing mass customization toolkits with user communities', *Journal of Product Innovation Management*, forthcoming, (2008).

[11] G. Friedrich, 'Elimination of Spurious Explanations', in *16th European Conference on Artificial Intelligence (ECAI 2004)*, eds., G. Müller and K. Lin, pp. 813–817, Valencia, Spain, (2004).

[12] L. Geneste and M. Ruet, 'Experience based configuration', in *17th International Conference on Artificial Intelligence*, volume 1, pp. 4–10. IJCAI, (2001).

[13] G. Haeubl and K.B. Murray, 'Preference Construction and Persistence in Digital Marketplaces: The Role of Electronic Recommendation Agents', *Journal of Consumer Psychology*, **13**(1), 75–91, (2003).

[14] C. Huffman and B. E. Kahn, 'Variety for sale: Mass customization or mass confusion?', *Journal of Retailing*, **74**(4), 491–513, (1998).

[15] M.J. Pazzani and D. Billsus, 'Content-based recommendation systems', *The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science*, **4321**, (2006).

[16] H. Tseng, C. Chang, and S. Chang, 'Applying case-based reasoning for product configuration in mass customization environments', *Expert Sys. with Applic.*, **29**(4), 913–925, (2005).

[17] D. Wilson and T. Martinez, 'Improved Heterogeneous Distance Functions', *Journal of Artificial Intelligence Research*, **6**, 1–34, (1997).