Towards recommending configurable offerings

J.Tiihonen*

Faculty of Information and Natural Sciences,
Department of Computer Science and Engineering,
Aalto University School of Science and Technology,
P.O. Box 19210, 00076 Aalto, Finland
Fax: +358-50-551-5224 E-mail: Juha.Tiihonen@tkk.fi
*Corresponding author

A. Felfernig

Faculty of Computer Science,
Institute of Software Technology,
Graz University of Technology,
Inffeldgasse 16b, A-8010 Graz, Austria
Fax: +43-(316)-873-5706 E-mail: Alexander.Felfernig@ist.tugraz.at

Abstract: Configuration technologies provide a solid basis for the implementation of a mass customisation 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 provide an overview of existing approaches to the integration of configuration and recommendation technologies, propose extensions and indicate directions of future work.

Keywords: knowledge-based configuration; recommender technologies; recommending configurable products.

Reference to this paper should be made as follows: Tiihonen, J. and Felfernig, A. (2010) 'Towards recommending configurable offerings', *Int. J. Mass Customisation*, Vol. 3, No. 4, pp.389–406.

Biographical notes: Juha Tiihonen is a Researcher and Project Manager at the Department of Computer Science and Engineering at the Aalto University School of Science and Technology (previously Helsinki University of Technology). He received his MSc (Tech.) and Lic.Sc. (Tech.) in Computer Science from Helsinki University of Technology. His main interest is product and service configuration in its various forms, including modeling, configurators, operations management aspects of business processes based on product and service configuration, and design for configuration. The most recent work includes recommendation support for configurable offerings. He has led several configuration related research projects.

Alexander Felfernig is a Professor at the Graz University of Technology. His research concentrates on different aspects of intelligent systems and business informatics such as knowledge-based configuration of complex products and services, knowledge-based recommenders in e-commerce, and model-based diagnosis with a special focus on knowledge acquisition and maintenance for complex products and services. In these research areas, he coordinates industrial and research projects, organises scientific events and has published more than 100 articles in international journals and conferences. Furthermore, he is the managing director of ConfigWorks, a company developing interactive recommendation technologies.

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 (Emde et al., 1996) – this phenomenon is well-known as mass confusion (Huffman and Kahn, 1998). On the other hand, customers do not know their preferences beforehand since preferences are typically constructed (Haeubl and Murray, 2003) 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 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 mechanisms effectively supporting the configuration and selection of interesting product and service alternatives. With a few exceptions (Coester et al., 2002; Geneste and Ruet, 2001; Tseng et al., 2005) existing recommender technologies (Felfernig et al., 2007a) 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 commercial configuration environments dealing with complex products and services. The major goals of this paper are threefold. First, we aim to provide an overview of applicability of existing approaches to the integration of configuration and recommendation technologies. Second, we discuss scenarios in which recommendation technologies can be exploited in configuration sessions. Third, we present promising existing approaches in detail with a working example and propose extensions. In addition, we point out issues for future work. This work has been conducted within the scope of the COSMOS project.[1]

1.1 Recommendation technologies

Collaborative filtering (see, e.g., Adomavicius and Tuzhilin, 2005) 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 neighbours*, which are users with similar preferences. 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 specific settings supporting, for example, the *collaborative development of product innovations* (Franke et al., 2008). This issue will not be discussed in this paper.

Content-based filtering (see, e.g., Pazzani and Billsus, 2006) 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 neighbours (Burke, 2002) 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. A major challenge of applying this technique to configurable offerings is that – beside the availability of accurate 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.

Utility-based recommendation estimates the relative satisfaction or desirability of consumption of an item, i.e., *utility* 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. In the context of configurable offerings, e.g., individual attribute or components settings can be recommended based on their utilities. An example for the application of utility-based recommendation approaches in the financial services domain is given in Felfernig et al. (2007b). A further discussion of utility-based approaches in the configuration context can be found, for example, in Ardissono et al. (2003).

Knowledge-based recommenders exploit explicit information about items and user requirements and how those can be satisfied (Burke, 2002; Felfernig et al., 2007b). *Constraint-based recommendation* (Felfernig and Burke, 2008) 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 (Felfernig et al., 2007b). Constraint-based approaches exploit the same technologies as many knowledge-based configuration environments and are additionally combined with, for example, utility-based recommendation supporting the ranking of candidate configurations.

Case-based recommendation (Burke, 2000) is another type of knowledge-based recommendation. In contrast to content-based filtering and collaborative filtering, elementary properties of items (e.g., PC price) in previous configurations are taken into account rather than extracted keywords or categories. It exploits similarity functions and Bayes predictors on previous configurations to determine interesting items and feature settings fitting to the wishes and needs of users. *Bayes predictors* allow the prediction of interesting items on the basis of their probability of being selected given the existing user preferences. Naïve Bayes predictors assume that variables are independent, which makes them computationally most feasible but potentially less accurate. Naïve Bayes

predictors have been applied for content-based filtering (Pazzani and Billsus, 2006) and case-based recommendation (Coester et al., 2002). *In this paper, we will apply and extend Naïve Bayes predictors for the identification of interesting alternatives and (sub)configurations* following the case-based recommendation approach.

1.2 Recommendation scenarios

It is possible to identify different scenarios for the recommendation of configurable offerings. In those scenarios, recommendation functionalities could 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).

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 recommendation into existing configurators. We analyse existing approaches in the field (Coester et al., 2002; Geneste and Ruet, 2001; Tseng et al., 2005) and introduce potential improvements.

The remainder of the paper is organised as follows. In the following section, we introduce a working example from the domain of configurable computers. In Section 3, we provide an overview of relevant recommendation algorithms for configurable products and services and propose extensions to be taken into account in future developments. With Section 4, we conclude 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. Table 5 exhibits five previous valid configurations (interaction outcomes), and an incomplete configuration of the active user, for whom suitable feature values will be recommended.

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 ($IntGr = yes$).

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

pr	*Socket*	*CScr*	*mb*	*Socket*	*IntGr*	*GScr*
as	a	1,250	a1	a	yes	300
i4	i	2,858	a2	a	no	0
i9	i	4,537	i1	i	yes	200
			i2	i	no	0

Separate graphics cards, Table 2(a), 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 an integrated graphics card specify their graphics performance with $GScr$. A system must always have a way to produce graphics. Thus the constraint $(mb.IntGr = no) \Rightarrow (gc \neq none)$ is introduced. Note that $none$ is a special feature value specifying that an empty assignment is made for an optional feature, allowed also in complete configurations. $pc.GScr$ refers to graphics performance of the PC, determined as the maximum $GScr$ provided by the graphics card or the motherboard.

Table 2 Graphics cards (a, left) and hard disks (b, right)

gc	*GScr*	*hd*	*capacity*
g2	2,800	h2	250
g8	2,200	h5	500
g9	5,500	h9	1,000

Hard disks (hd) are available in different capacities (GB), see Table 2(b). All optical drives, see Table 3, read CD and DVD. Some write DVD or DVD + Blu-ray.

Table 3 Properties of optical drives of the working example

od	*Write DVD and 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 4 Intended usage features of the working example

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)

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 4.

The following domain knowledge expressed as constraints is available:

$ph \neq no \Rightarrow od.dw = yes$: to archive photos
$ph = adv \Rightarrow hd.capacity \geq 500$: disk space for advanced photo processing
$ph = adv \Rightarrow pr.CScr \geq 2{,}500$: CPU for advanced photo processing
$ph = adv \Rightarrow me \geq 2$: RAM for advanced photo processing
$vi = sd \Rightarrow pr.CScr \geq 2{,}700$: CPU for SD video editing
$vi = hd \Rightarrow pr.CScr \geq 4{,}500$: CPU for HD video editing
$vi = sd \Rightarrow od.dw = yes$: burn DVD videos for SD video editing
$vi = hd \Rightarrow od.bw = yes$: burn Blu-ray videos for HD video editing
$ga = 3d \Rightarrow pr.CScr \geq 1{,}500$: CPU for 3D gaming
$ga = 3d \Rightarrow pc.GScr \geq 1{,}500$: graphics for 3D gaming
$ga = adv \Rightarrow pr.CScr \geq 2{,}800$: CPU for advanced gaming
$ga = adv \Rightarrow pc.GScr \geq 5{,}000$: graphics for advanced gaming

Some algorithms discussed in this paper apply *feature importance weights*. We apply the following distribution: 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%. These weights could stem from direct customer specifications, representative preferences from statistical samples, or the application of utility constraints as documented in Felfernig et al. (2007b).

Notation

We base the discussion of recommendation algorithms on the following conventions. Relation $conf$ holds previous K complete and consistent configurations, each specifying consistent values for all the existing N features $f_1, \ldots, f_N$. The value of feature f_i in configuration k is referred to as $f_{i,k}$. For example $f_{vi,3} = sd$, see Table 5. The kth configuration is referred to as $conf_k$. Classification (discussed in Section 3.1 in the context of distance metrics) of configuration k is referred to as cl_k. When referring to the profile of the active user, we use index u, $u \notin \{1, 2, \ldots, K\}$, e.g., $f_{i,u}$ refers to the value of feature f_i for the active user. A recommended value for feature f_j for the active user is denoted by $r_{f_j,u}$. The set of specified features in the active user profile is F_u, in our example $\{f_{vi}, f_{ph}, f_{ga}\}$, see the last row of Table 5. $F_u = \{f_j | f_{j,u} \neq noval\}$, and the set of features for which the active user profile does not have a value is $\bar{F}_u$, in our example $\{f_{pr}, f_{mb}, f_{me}, f_{hd}, f_{gc}, f_{od}\}$. Note that $noval$ specifies that there is no assignment to a feature value, illustrated as an empty cell in Table 5. $noval$ is not allowed in complete configurations. Furthermore, a projection $\pi_{F_u}(conf)$ over previous configurations $conf$ onto the set of features F_u for which the active user has set a value is referred to as $conf_{F_u}$. The corresponding projection onto features that the active user does not have a value is $conf_{\bar{F}_u}$. To avoid the unintended removal of duplicate tuples, the index of a configuration k is considered to be included in $conf_k$ and projections, see Table 5. Finally, $dom(f_j)$ returns the domain of feature f_j.

Table 5 Configurations from previous configuration sessions and the active user profile

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*	*cl*
1	no	no	2d	as	a1	1	h2	none	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							

3 Recommendation algorithms

In this section, we provide an overview of existing algorithms supporting personalised configuration processes and indicate possible ways to extend those to be more applicable in real-world settings. The algorithms can be used for two different purposes: on the one hand for the prediction of individual feature values and on the other hand for recommendation of all missing feature values to complete a configuration. The case-based recommendation approach is applied – the idea is to investigate existing (similar) configurations in order to predict interesting feature settings and complete solutions. Note that our cases describe user preferences as well as technical features.

Four recommendation algorithms will be discussed and illustrated with the working example. Some of the algorithms rely on distance metrics discussed in Subsection 3.1 to determine similarity or dissimilarity of feature values. Two of the algorithms, *nearest neighbour* (3.2) and *most popular choice* (3.5) provide recommendations for entire remaining configurations. Recommendation of individual feature values is supported by *weighted majority voter* (3.3) and *Naïve Bayes voter* (3.4). For each approach, we present its idea and extensions that can further improve it with regard to dimensions such as prediction quality. A discussion of further approaches (3.6) concludes this section.

3.1 Distance metrics

The recommendation 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. For example, many 500 GB hard disks from different manufacturers are very similar, and slightly larger 640 GB hard disks could often be used interchangeably.

We apply ideas of the *heterogeneous value difference metric* (*HVDM*) (Wilson and Martinez, 1997) to cope with symbolic (nominal) and numeric features in a relatively simple manner. On the feature level, the distance is defined as follows: the function $d_{f_i}(x, y)$, formula (1), returns the distance between values x and y of feature f_i, using a different sub-function for different types of features: distances between symbolic feature values are computed by the function $vdm_{f_i}(x, y)$ (2), and those between numeric values by the function $diff_{f_i}(x, y)$ (3). Based on experiments of Wilson and Martinez (2005), these functions provide similar influence on the overall distance measurements. Distance values returned by $d_{f_i}(x, y)$ are *normalised* to usually be in range 0 to 1.

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

The function $vdm_{f_i}(x,y)$ learns the similarity of symbolic values in a domain automatically. This is done by examining the probability that individual feature values contribute to the *classification* of the samples – 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 classifications [basic (ba), standard (st) or advanced (ad)] that represent the sophistication level of the configuration. This classification is used as the classifier for *HVDM* (see column 'cl' in Table 5); this is calculated as follows – see formula (2):

$$vdm_{f_i}(x,y) = \sqrt{\sum_{cl \in C} \left| \frac{N_{f_i,x,cl}}{N_{f_i,x}} - \frac{N_{f_i,y,cl}}{N_{f_i,y}} \right|^2} = \sqrt{\sum_{cl \in C} |P_{f_i,x,cl} - P_{f_i,y,cl}|^2} \tag{2}$$

In the function $vdm_{f_i}(x,y)$, $N_{f_i,x}$ is the number of instances (configurations) in $conf$ that have value x for feature f_i; $N_{f_i,x,cl}$ is the number of instances in $conf$ that have value x for feature f_i and output class cl; C is the set of output classes in the problem domain (in our case $\{ba, st, ad\}$). $P_{f_i,x,cl}$ is the conditional probability of output class cl given that feature f_i has the value x, i.e., $P(cl|f_i = x)$, determined as $\frac{N_{f_i,x,cl}}{N_{f_i,x}}$. When $N_{f_i,x} = 0$, $P(cl|f_i = x)$ is considered 0.

In our example population $conf$, $N_{pr,as} = 2$, $N_{pr,i4} = 2$, and $N_{pr,i9} = 1$. The classification frequencies for processor $N_{pr,x,cl}$ are presented on left half of Table 6, 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 6.

Table 6 Classification frequencies of pr (left) and corresponding distance matrix (right)

cl, x	*as*	*i4*	*i9*		*as*	*i4*	*i9*
ba	1	0	0	as	0	0.707	1.225
std	1	1	0	i4	0.707	0	0.707
adv	0	1	1	i9	1.225	0.707	0

Distances between numeric values x and y of feature f_i (in $conf$) are determined by the function $diff_{f_i}(x,y)$ (3). Since 95% of the values in a normal distribution fall within two standard deviations of the mean, the difference between numeric values 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 Wilson and Martinez (1997), normalisation through standard deviation is often preferable to normalisation through division by the range of that feature.

$$diff_{f_i}(x,y) = \frac{|x-y|}{4\sigma_{f_i}} \tag{3}$$

The distance metric $d_{f_i}(x, y)$ usually returns a value from 0 to 1. The similarity $sim_{f_i}(x, y)$ (4) between two feature values x and y of feature f_i can then be defined as:

$$sim_{f_i}(x, y) = 1 - d_{f_i}(x, y) \tag{4}$$

In the following subsections, presenting the four algorithms and their extensions, we will show how these metrics can be applied to the recommendation of feature values as well as to the recommendation of complete configurations.

3.2 Nearest neighbour

The idea of a *nearest neighbour* is simple: determine a neighbour configuration, which is closest to the known parts of active user's profile, and *recommend feature values of this nearest neighbour* for remaining features. The distance of the configuration $conf_u$ of the active user and neighbour configuration $conf_a$ is defined as the sum of distances between corresponding feature values, weighted by feature importance weights. Only features that have a value in active users configuration are taken into account ($f_i \in F_u$):

$$dist(conf_u, conf_a) = \sum_{f_i \in F_u} d_{f_i}(f_{i,u}, f_{i,a}) * w(f_i) \tag{5}$$

A nearest neighbour configuration c with smallest distance is identified among those configurations that complete the active user's configuration $conf_u$ in a consistent manner. Recommendations are feature values of this the nearest neighbour configuration c.

$$r_{f_j,u} = f_{j,c} \text{ for features } f_j \in \bar{F}_u \tag{6}$$

In our example, the nearest neighbour 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 neighbour formula to configurations $conf_2 \ldots conf_5$ 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). Consequently, feature values of the nearest consistent neighbour $conf_5$ are recommended: $r_{pr,u} = i4$, $r_{mb,u} = i1$, $r_{me,u} = 2$, $r_{hd,u} = h9$, $r_{gc,u} = g8$ and $r_{od,u} = dw$. Recommending feature values of the nearest consistent neigbor is straightforward. However, in the case of non-existence of consistent nearest neighbours, repair actions have to be taken into account. Further details regarding the calculation of repair actions can be found in Felfernig et al. (2004).

3.3 Weighted majority voter

The *weighted majority voter* (Coester et al., 2002) *recommends individual feature values* based on each neighbour configuration in $conf$ 'voting' for its feature values. The weight of each neighbour vote (*neighboir weight*) is determined by the number of feature values equal to already determined values of the active user's profile (F_u). For example, the weight of $conf_1$ for the active user u is 2, because $f_{vi,1} = f_{vi,u} = no$, and

$f_{ph,1} = f_{ph,u} = no$. Thus, $conf_1$ would contribute by recommending its feature values by giving 2 votes to its feature value settings of the yet unspecified features $f_{pr} = as$, $f_{mb} = a1$, $f_{me} = 1$, $f_{hd} = h2$, $f_{gc} = none$, and $f_{od} = dr$. As a sum of all neighbour configurations voting for their feature values with their neighbour weight, the following feature values get most votes: $f_{pr} = as$ (3 votes), $f_{mb} = a1$ (2), $f_{me} = 1$ (3), $f_{hd} = h2$ (2), $f_{gc} = none$ (2), and $f_{od} = dr$ (2).

The consistency of a potential recommendation is checked by adding the proposed value to the known values in the user profile. After the user selects a feature value, recommendations will be recalculated to reflect the new situation. Due to consistency checks, $pr = i4$ (one vote) and $gc = g2$ (one vote) replace those with most votes, assuming the selection of the first feature value in a domain in case of a tie in votes. Thus, the recommendations are $r_{pr,u} = i4$, $r_{mb,u} = a1$, $r_{me,u} = 1$, $r_{hd,u} = h2$, $r_{gc,u} = g2$ and $r_{od,u} = dr$.

Next, we discuss the formula proposed in Coester et al. (2002) that formalises the approach outlined above, and provide our extensions. First, we define the equality function eq to return 1 when two feature values are equal, otherwise 0.

$$eq(x, y) = \begin{cases} 1 & \text{if x=y} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The neighbour weight $w(conf_x, conf_u)$ of a neighbour 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):

$$w(conf_x, conf_u) = \sum_{i \in F_u} eq(f_{i,x}, f_{i,u}) \tag{8}$$

Neighbours voting for their feature values establishes a specific amount of support for each feature value $v \in dom(f_j)$. This support for user's configuration $conf_u$ to have value v for feature f_j is expressed as a *prediction score* $pr(conf_u, f_j, v)$. It is achieved as the sum of neighbour weights (interpreted as votes) having value v for feature f_j:

$$pr(conf_u, f_j, v) = \sum_{i=1}^{K} eq(f_{j,i}, v) * w(conf_i, conf_u) \tag{9}$$

A consistent feature value v with maximum prediction score $pr(conf_u, f_j, v)$ is the recommendation $r_{f_j,u}$ for feature f_j in configuration $conf_u$.

One possible extension to the approach of Coester et al. (2002) is an alternative way for determining the neighbour weights: Neighbour weights can be determined by the similarity of neighbour and user profile feature values instead of equality. This would improve prediction quality in presence of similar feature values, because similar values compared to user's existing selections would also contribute to the weight of a neighbour. Second, the importance of individual features for a user (feature weights) should be taken into account. Thus, the weight $w(conf_x, conf_u)$ of a neighbour configuration $conf_x$ with respect to configuration $conf_u$ is defined as follows [instead of formula (8)]:

$$w(conf_x, conf_u) = \sum_{i \in F_u} sim_{f_i}(f_{i,x}, f_{i,u}) * w(f_i) \tag{10}$$

The weights of example neighbours 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 potential recommendations are $r_{pr,u} = as$ (0.191), $r_{mb,u} = a1$ (0.126), $r_{me,u} = 1$ (0.191), $r_{hd,u} = h2$ (0.126), $r_{gc,u} = none$ (0.126) and $r_{od,u} = dr$ (0.126). To provide (locally) consistent recommendations, pr and gc should be substituted with closest consistent feature values $r_{pr,u} = i4$ (0.060) and $r_{gc,u} = g8$ (0.093).

3.4 Naïve Bayes voter

'Naïve Bayes voter' Coester et al. (2002) *recommends individual feature values*. To determine a recommendation for feature f_j, a probability predictor is determined for each feature value v in the domain of f_j. A feature value with the highest predictor will be recommended. The Naïve Bayes voter [formula (12)] is such a predictor that applies the idea of the Bayes theorem – see formula (11):

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{11}$$

Here A and B are Boolean-valued random variables representing occurrence of corresponding events, and $P(A)$ and $P(B)$ are probabilities of these events. $P(B|A)$ denotes the conditional probability of event B given that event A has taken place, and $P(A|B)$ the conditional probability of event A given event B.

In the case of Naïve Bayes voter, event B represents the fact that feature f_j has value v. Event A represents the fact that a configuration has the combination of feature values already specified by the active user in F_u. Thus, $P(A|B)$ is the conditional probability of the active user's current value combination for already specified features, given that feature f_j has value v. Finally, $P(B|A)$ is the probability for the feature f_j to have value v, given the partial configuration of the active user. Applying this idea, the predictor formula (12) consists of two parts: a basic probability $P(B)$ and a conditional probability $P(A|B)$. Note that the divisor $P(A)$ of formula (11) is omitted – hence, the predictor value is directly proportional to the probability $P(B|A)$ but does not directly represent a probability. This safely simplifies calculations because the predictor value is effectively used only for ranking feature values - not for determining a probability, and $P(A)$ would be the same for all feature values that will be compared. The parts of formula (12) will be described below, necessary utility functions will be defined, and illustrating examples provided. Finally, potential extensions for providing higher prediction quality will be introduced.

$$pr(f_j, v) = \overbrace{Pr_{basic}(f_j, v)}^{\text{basic probability } P(B)} * \underbrace{\prod_{f_i \in F_u} m_{est}(count_{match}(f_j, v, f_i, u), count(f_j, v), 1/K, K)}_{\text{conditional probability } P(A|B)} \tag{12}$$

The basic probability $P(B)$ for value v of feature f_j is simply the proportion of configurations having that feature value, see formula (13). Formula (13) applies utility function $count(f_j, v)$ (14) that returns the number of neighbours in $conf$ having value v for feature f_j. For example, applying formula (13) for feature optical drive f_{od} for value $v = dw$ gives $Pr_{basic}(f_{od}, dw) = 0.600$ because three of five neighbour configurations have value dw for f_{od}.

$$Pr_{basic}(f_j, v) = \frac{count(f_j, v)}{K} \tag{13}$$

$$count(f_j, v) = \sum_{k=1}^{K} eq(f_{j,k}, v) \tag{14}$$

The conditional probability part $P(A|B)$ of formula (12) determines a probability of the active user's current value combination (features values of F_u), given those neighbour configurations that have feature $f_j = v$. For each feature $f_i \in F_u$, a conditional probability estimate $P(f_{i,u}|f_j = v)$ for the current value of the active user $(f_{i,u})$ is calculated. The conditional probability for the combination of values in F_u is the product of individual feature value probability estimates, utilising the independence assumption. When determining the probability estimate $P(f_{i,u}|f_j = v)$ for feature f_i, three aspects are considered. First, the estimation takes into account exactly those N configurations in $conf$ that have value v for feature f_j, i.e., $f_{j,k} = v$, these configurations are referred to as $conf_v$. N_c is the number of neighbours within $conf_v$ that have equal feature value as the user profile for feature f_i, that is, $f_{i,k} = f_{i,u}$ and $f_{j,k} = v$. Probability estimate $p_c = \frac{N_c}{N}$ with a small N_c or N would be subject to large variations. Therefore the third aspect, applying an *m-estimate* (Bratko et al., 1996) to stabilise probability calculations even in case of (too) few samples is adopted. The idea of an m-estimate can be summarised as follows. Let N_c be the number of samples in a class c whose probability p_c is being estimated, and N the total number of observed samples. The m-estimate adds m virtual samples with probability p that augment the estimation of probability. When N grows sufficiently, real samples outweigh the virtual samples. Formula (15) represents this idea. N is determined with function $count$ (14). N_c is determined with function $count_{match}$ (16). Formula (12) applies the same m-estimate virtual sample parameters as Coester et al. (2002): $m = K$, the number of neighbours and $p = 1/K$.

$$m_{est}(N_c, N, p, m) = \frac{N_c + mp}{N + m} \tag{15}$$

The utility function $count_{match}(f_j, v, f_i, u)$ returns the number of neighbours in $conf$ having value v for feature f_j and an equal value as the user profile u for feature f_i.

$$count_{match}(f_j, v, f_i, u) = \sum_{k=1}^{K} eq(f_{j,k}, v) * eq(f_{i,u}, f_{i,k}) \tag{16}$$

Continuing the working example, the conditional probability $P(A|B)$ given event B: $f_{od} = dw$ is determined. Out of 3 neighbours with $f_{od} = dw$, 1 ($conf_2$) has the same video specification as our active user profile ($f_{vi} = no$), which results

in an m-estimate $m_{est}(1, 3, 0.2, 5) = 0.250$. Calculating the remaining m-estimates for the other two already specified features in context of $f_{od} = dw$ produces the following results. Photo f_{ph} : $m_{est}(0, 3, 0.2, 5) = 0.125$ because 0 neighbours out of 3 with $f_{od} = dw$ has the same f_{ph} value as active user. Gaming f_{ga}: $m_{est}(1, 3, 0.2, 5) = 0.250$ (1 neighbour out of 3 has the same f_{ga} value). Therefore the predictor $pr(f_{od,u}, dw) = 0.600 * (0.250 * 0.125 * 0.250) = 0.0046875$. Predictors for other possible values of f_{od} are smaller. Because the predictor for $f_{od} = dw$ is highest and consistent with existing feature values of the active user, recommendation $r_{od,u} = dw$ is determined. The potential recommendations for feature values with highest predictors are $r_{mb,u} = a1$, $r_{me,u} = 1$, $r_{hd,u} = h2$, and $od = dw$. Due to consistency constraints, $r_{pr,u} = i4$ instead of as that has the highest predictor, and $r_{gc,u} = g2$ instead of $none$. Note that recommendations $r_{pr,u} = i4$ and $r_{mb,u} = a1$ are mutually incompatible. As previously, we follow the approach that recommendations for individual features will be recalculated after selecting a feature value, which alleviates the problem of mutually inconsistent individual feature value recommendations. A potential extension to improve the prediction accuracy of the Naïve Bayes voter in the presence of similar feature values is to take into account similar feature values in neighbour configurations instead of requiring them to be equal. A straightforward method is to modify the determination of the basic probability $P(B)$. We give each feature value 'support' when neighbour configurations have feature values within maximum distance Δ, $d_{f_j}(f_{j,u}, f_{j,a}) \leq \Delta$, instead of requiring equal feature values as formula (13) does. 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 $s_{f_j}(x, y)$:

$$s_{f_j}(x, y) = \begin{cases} (1 - d_{f_j}(x, y))^2, & \text{if } d_{f_j}(x, y) \leq \Delta \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Support for an individual feature value is divided by the sum of supports given to all values in domain of f_j. The extended basic probability is thus:

$$Pr_{basic}(f_j, v) = \frac{\sum_{k=1}^{K} s_{f_j}(v, f_{j,k})}{\sum_{w \in dom(f_j)} \sum_{k=1}^{K} s_{f_j}(w, f_{j,k})} \tag{18}$$

Applying the similarity-based basic probability formula (18) with a (large) $\Delta = 0.8$ yields alternative basic probabilities, e.g., $Pr_{basic}(f_{od}, dw) = 0.578$. In this example, same recommendations result as with the original formula.

3.5 Most popular choice

The most popular choice algorithm (Coester et al., 2002) *recommends feature values for the set of features that the user does not have a value*, typically to complete a configuration. A top-down view of the algorithm is is as follows: The most popular choice algorithm applies the idea of the Bayes theorem (11) and the Naïve Bayes assumption of independent features. A probability estimate for each a configuration $c \in conf$ is calculated by applying formula (19). A configuration c with highest probability estimate is identified among those configurations that complete the active user's configuration $conf_u$ in a consistent manner. Recommendations are feature values of this configuration c.

In this case, event B represents the fact that a configuration has the feature value combination of configuration $conf_c$ for the set of features that the user does not have a value, $\bar{F}_u$. Event A represents the fact that a configuration has the combination of feature values already specified in F_u. Thus, $P(A|B)$ is the conditional probability of the active user's value combination for already specified features, given configurations with feature value combination of configuration $conf_c$ for the set of unknown features $\bar{F}_u$. Finally, $P(B|A)$ is the probability of the feature value combination of $conf_c$ for the yet unspecified features, given the partial configuration of the active user. Applying this idea, the predictor formula (19) consists of two parts: a basic probability $P(B)$ and a conditional probability $P(A|B)$. As with Naïve Bayes Voter, the divisor $P(A)$ of formula (11) is omitted, and the predictor only represents a value directly proportional to the probability $P(B|A)$. Next, the parts of formula (19) will be detailed, necessary utility functions introduced, and illustrating examples provided. Finally, potential extensions for providing higher prediction quality will be introduced.

The *basic probability* $P(B)$ part, $Pr_{basic}(c, \bar{F}_u)$, represents the probability of the feature value combination of configuration c for the set of features that the user does not have a value, $\bar{F}_u$. This basic probability is directly determined by relative popularity of feature values of $conf_c$ for the set of features $\bar{F}_u$, formula (20). The *basic probability for an individual feature* f_j having value v that configuration c has ($f_{j,c} = v$) is simply the proportion of configurations of $conf$ having value v for feature f_j. The *basic probability of configuration c* is determined by multiplying the basic probabilities for individual feature values of the configuration ($f_j \in \bar{F}_u$). These aspects are formalised in formula (20). Formula (20) applies function $count(f_j, v)$ (14) to determine the number of neighbours in $conf$ with value v for feature f_j.

$$Pr(c, u, F_u) = \overbrace{Pr_{basic}(c, \bar{F}_u)}^{\text{basic probability } P(B)} * \underbrace{\prod_{f_j \in F_u} m_{est}(eqcfgs_{match}(c, \bar{F}_u, f_j, f_{j,u}), eqcfgs(c, \bar{F}_u), 1/K, K)}_{\text{conditional probability } P(A|B)} \quad (19)$$

$$Pr_{basic}(c, \bar{F}_u) = \prod_{j \in \bar{F}_u} \frac{count(f_j, f_{j,c})}{K} \quad (20)$$

In our example, the basic probability of feature value combination of $conf_5$, $Pr_{basic}(conf_5, \bar{F}_u)$ is determined with the formula (20) as follows. The basic probabilities of the individual unknown features are calculated. For example, feature $f_{pr,5} = i4$. Two out of five configurations of $conf$ has value $i4$ for feature f_{pr}. Therefore, the basic probability is $2/5 = 0.4$. For other features of $conf_5$, the basic probabilities are 0.2 (f_{mb}: 1 * $i1$), 0.2 (f_{me}: 1 * 2), 0.4 (f_{hd}: 2 * $h9$), 0.2 (f_{gc}: 1 * $g8$) and 0.6 (f_{od}: 3 * dw). Therefore, the basic probability of $conf_5$ is 0.4 * 0.2 * 0.2 * 0.4 * 0.2 * 0.6 = 0.000768.

The conditional probability $P(A|B)$ part of formula (19) provides the probability of value combination in F_u, given configurations that have the same configuration as $conf_c$ with respect to those features that the user profile does not have a value ($\bar{F}_u$). This conditional probability estimate is based on four aspects. First, the estimation takes into

account exactly those configurations in $conf$ that have the same configuration as $conf_c$ with respect to those features that the user profile does not have a value ($\bar{F}_u$). These N configurations are referred to as $conf_C$. Second, each feature $f_j \in F_u$ that the active user already has a value is analysed individually to determine $P(f_j = f_{j,u}|conf_C)$. N_c is the number of configurations within $conf_C$ that have the same feature value for feature f_j as the active user's profile, $f_{j,u} = f_{j,c}$. Third, the *m-estimate* approach is applied for determining the probability estimate for feature f_j, using formula (15). We apply the m-estimate parameters of Coester et al. (2002): The number of 'in-samples' N_c and the number of all samples N were described above. To calculate N and N_c, utility functions defined in formulas (21), (22) and (23) are applied. The virtual sample parameters for the m-estimate are $p = 1/K$ and $m = K$. Fourth, the conditional probability given the value combination of $conf_c$ is achieved by multiplying the individual feature probability estimates (m-estimates), based on the assumption of independent features.

Formula $eqcfg(i, j, F)$ (21) tests if 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.

$$eqcfg(i, j, F) = \begin{cases} 1 & \text{if } \forall f_f \in F : eq(f_{f,i}, f_{f,j}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Formula $eqcfgs(c, F)$ (22) returns the number of configurations that are equal to configuration c with respect to a set of features F.

$$eqcfgs(c, F) = \sum_{k=1}^{K} eqcfg(c, k, F) \tag{22}$$

Finally, formula $eqcfgs_{match}(c, F, f_j, v)$ (23) returns the number of configurations that are equal to configuration c with respect to a set of features F, and additionally have value v for feature f_j.

$$eqcfgs_{match}(c, F, f_j, v) = \sum_{k=1}^{K} eqcfg(c, k, F) * eq(f_{j,k}, v) \tag{23}$$

The conditional probability part of the predictor of formula (19) is applied in the example as follows. Related to $conf_5$, the set $conf_C$ consists only of $conf_5$ because there are no other equal configurations to $conf_5$ with respect to features absent from user profile. Thus $N = 1$. The product of formula (19) expands as $m_{est}(0, 1, 0.2, 5) * m_{est}(0, 1, 0.2, 5) * m_{est}(1, 1, 0.2, 5) = 0.167 * 0.167 * 0.167 = 0.00926$. For example, in the first term $N_c = 0$, because configuration $conf_5$ has $f_{vi,5} = sd$ that is not equal to no in the user profile. The third term for f_{ga} has $N_c = 1$, because $f_{ga,5} = 3d$ which equals $f_{ga,u}$ of the user profile.

Combining the basic probability and the conditional probability parts of formula (19), $conf_5$ has the highest predictor $0.000768 * 0.00926 = 0.00000711$. Completing active user's configuration with feature values of $conf_5$ is consistent, therefore they are recommended: $r_{pr,u} = i4$, $r_{mb,u} = i1$, $r_{me,u} = 2$, $r_{hd,u} = h9$, $r_{gc,u} = g8$ and $r_{od,u} = dw$.

As with the Naïve Bayes voter above, the concept of basic probability can be extended to give each feature value support when neighbour configurations have feature values within maximum distance Δ, ($d_{f_j}(f_{j,u}, f_{j,a}) \leq \Delta$), defined in formula (17). To calculate the modified basic probability, 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. Recommendations would not be affected by eliminating the divisor, which would simplify calculations with the trade-off of sacrificing numeric compatibility with the original formula. Therefore, we replace formula (20) with formula (24):

$$Pr_{basic}(c, \bar{F}_u) = \prod_{f_j \in \bar{F}_u} \frac{\sum_{k=1}^{K} s_{f_j}(f_{j,c}, f_{j,k})}{\sum_{w \in dom(f_j)} \sum_{k=1}^{K} s_{f_j}(w, f_{j,k})} \tag{24}$$

With our modified formula (24) and (a large) Δ= 0.8 the individual feature value probabilities are: 0.403 (*pr*), 0.286 (*mb*), 0.280(*me*), 0.444 (*hd*), 0.286 (*gc*) 0.578 (*od*). Therefore, the basic probability of $conf_5$ = 0.00237.

With the extended algorithm $conf_5$ has the predictor $0.00237 * 0.00926 = 0.0000219$, slightly higher than 0.0000204 of $conf_3$. As above, the values of $conf_5$ are recommended.

3.6 Further approaches

The algorithms discussed in this paper can be seen as representative extensions for existing commercial configuration environments. Similar to the algorithms analysed are those discussed in Tseng et al. (2005) that as well represent an approach to the application of case-based reasoning in product configuration. 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. In this context, Tseng et al. (2005) do not take into account probabilities which provide an indication of the most promising similar cases. Geneste and Ruet (2008) presents an approach to integrate case-based reasoning with constraint solving with the goal to adapt identified nearest neighbours to the new configuration problem. The used algorithm (Geneste and Ruet, 2008) for calculating nearest neighbours takes into account component structures but does not take into account probabilities of selection. The authors introduce an approach to the calculation of adaptations for the identified nearest neighbours in order to conform with the new customer preferences. No details are provided regarding the minimality of changes or how the adaption affects 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. There exist a couple of approaches similar to Geneste and Ruet (2008), where cased-based reasoning (Kolodner, 1993) is integrated with constraint solving. On the one hand, the idea is to make constraint solving more efficient by the means of starting with already existing cases. On the other hand, the feasibility checking of case-adaptations requires the integration of corresponding constraint technologies (Purvis, 1997). In case-based reasoning, one of the major challenges in this context is to develop criteria for the adaptability of cases. Such criteria are taken into account, for example, in Purvis (1997) and Smyth and Keane (1996). 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 (Felfernig et al., 2004; Friedrich, 2004) have to be provided.

4 Conclusions and future work

In this paper, we provide an overview of approaches to integrate case-based recommendation with configuration technologies. This integration allows for the derivation of individualised and personalised 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. We proposed extensions to already existing recommendation algorithms (in terms of importance weights, similarity metrics with a new classification approach, and consistency criteria for recommendations) with the goal to indicate potential improvements in future system developments. Future work in this area will be to evaluate the proposed extensions within the scope of a series of user studies and to apply our approach in other application scenarios such as reconfiguration. For example, in insurance and financial domains the situation or needs of individuals or customer organisations 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. In our view, recommender-supported configuration systems have significant potential to support the mass customisation strategy, reduce mass confusion and to make complex products and services accessible to larger audiences even in self-service e-commerce scenarios.

References

Adomavicius, G. and Tuzhilin, A. (2005) '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*, Vol. 17, No. 6, pp.734–749.

Ardissono, L., Felfernig, A., Friedrich, G., Jannach, D., Petrone, G., Schäfer, R. and Zanker, M. (2003) 'A framework for the development of personalized, distributed web-based configuration systems', *AI Magazine*, Vol. 24, No. 3, pp.93–108.

Bratko, I., Cestnik, B. and Kononenko, I. (1996) 'Attribute-based learning', *AI Communications*, Vol. 9, No. 1, pp.27–32.

Burke, R. (2000) 'Knowledge-based recommender systems', *Encyclopedia of Library and Information Systems*, Vol. 69, No. 32.

Burke, R. (2002) 'Hybrid recommender systems: survey and experiments', *User Modeling and User-Adapted Interaction*, Vol. 12, No. 4, pp.331–370.

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

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

Felfernig, A. and Burke, R. (2008) 'Constraint-based recommender systems: technologies and research issues', in *ACM, editor, ACM International Conference on Electronic Commerce*, Innsbruck, Austria, pp.17–26.

Felfernig, A., Friedrich, G., Jannach, D. and Stumptner, M. (2004) 'Consistency-based diagnosis of configuration knowledge bases', *Artificial Intelligence*, Vol. 2, No. 152, pp.213–234.

Felfernig, A., Friedrich, G. and Schmidt-Thieme, L. (2007a) 'Recommender systems', *IEEE Intelligent Systems-Special Issue on Recommender Systems*, Vol. 22, No. 3.

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

Franke, N., Keinz, P. and Schreier, M. (2008) 'Complementing mass customization toolkits with user communities', *Journal of Product Innovation Management*, Vol. 25, No. 6, pp.546–559.

Friedrich, G. (2004) 'Elimination of spurious explanations', in G. M"uller and K. Lin (Eds.): *16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, pp.813–817.

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

Haeubl, G. and Murray, K.B. (2003) 'Preference construction and persistence in digital marketplaces: the role of electronic recommendation agents', *Journal of Consumer Psychology*, Vol. 13, No. 1, pp.75–91.

Huffman, C. and Kahn, B.E. (1998) 'Variety for sale: mass customization or mass confusion?', *Journal of Retailing*, Vol. 74, No. 4, pp.491–513.

Kolodner, J. (1993) *Case-based Reasoning*, Morgan Kaufmann Publisher.

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

Purvis, L. (1997) 'Dynamic constraint satisfaction using case-based reasoning techniques', in *Constraint Programming (CP07) Workshop on Dynamic Constraint Satisfaction*.

Smyth, B. and Keane, M. (1996) 'Using adaptation knowledge to retrieve and adapt design cases', *Journal of Knowledge-based Systems*, Vol. 9, No. 2, pp.127–135.

Tseng, H., Chang, C. and Chang, S. (2005) 'Applying case-based reasoning for product configuration in mass customization environments', *Expert Sys. with Applic.*, Vol. 29, No. 4, pp.913–925.

Wilson, D. and Martinez, T. (1997) 'Improved heterogeneous distance functions', *Journal of Artificial Intelligence Research*, Vol. 6, pp.1–34.

Notes

1 Customer-oriented systematically managed service offerings (COSMOS) is a project supported by the Finnish Funding Agency for Technology and Innovation.