Personalized Product Configuration

Alexander Felfernig[1], Monika Mandl[1], Juha Tiihonen[2] , Monika Schubert[1]

[1]Institute of Applied Informatics,
Graz University of Technology

[2]Software Business and Engineering Institute,
Helsinki University of Technology

1 Introduction

Configuration systems have a long tradition as a successful application area of Artificial Intelligence, see, for example, Barker et al. (1989, p. 298-318); Fleischhanderl et al. (1998, p. 59-68); Mittal and Frayman (1990, p. 1395-1401); Stumptner (1997, p. 111-126). On an informal level, configuration can be inter-preted as a special case of design activity where the artifact being configured is assembled from instances of a fixed set of well-defined component types which can be composed conforming to a set of constraints (Sabin and Weigel 1998, p. 42-49). Constraints can represent technical restrictions, rules regarding production processes, or restrictions that are related to economic factors. Example domains where product configurators are applied are computers, cars, financial services, railway stations, and complex telecommunication switches.

Although configuration has many advantages such as a significantly lower amount of incorrect quotations and orders, shorter product delivery cycles, and higher productivity of sales representatives (Fleischhanderl et al. 1998, p. 59-68), customers (users) in many cases have the problem of not understanding the set of offered options in detail and are often overwhelmed by the complexity of those options. This phenomenon is well known as mass confusion (Huffman and Kahn 1998, p. 491-513). The other problem is that users typically do not know exactly which products or components they would like to have. This phenomenon is described by the theory of preference construction (Bettman et al. 1998, p. 187-217) which follows from the fact that users do not know their preferences beforehand but rather construct and adapt their preferences within the scope of (in our case) a configuration process. In such a situation it makes sense to support users with recommendations that are, for example, derived from preferences articulated by similar users (Tiihonen and Felfernig 2009, to appear).

In this paper we present an environment that supports personalized configura-tion of mobile phones and corresponding subscriptions (RecoMobile). Our major contribution is the integration of recommendation technologies with knowledge-based configuration (a functionality not available in commercial systems) and to show the applicability of the developed concepts through an empirical study.

The remainder of the paper is organized as follows. In the next section we introduce major functionalities of RecoMobile by showing a simple application scenario. We then present the recommendation algorithms integrated in the RecoMo-bile configurator (third section). In the following we discuss the results of a user study conducted with the RecoMobile system with the goal to point out the major improvements that can be achieved by applying recommendation technologies in the context of knowledge-based configuration scenarios (fourth section). Thereaf-ter we discuss related work and conclude the paper.

2 The RecoMobile Configurator

RecoMobile is a knowledge-based configurator for mobile phones & services enriched with recommendation functionalities that help to predict useful feature settings for the user. The entry screen of RecoMobile is depicted in Figure 1. First, the user has to answer a few questions concerning some general attributes of the configuration domain (e.g., *the preferred phone style* or *preferences regarding the internet access*). For the following questions regarding mobile subscription details, privacy settings, and the phone, the recommender proposes feature settings (default pro-posals) that are determined on the basis of user interactions of past configuration sessions. After the specification of a set of requirements, the configuration system checks whether a solution (configuration) exists. In the case that no solution could be found, the system activates a diagnosis & repair component (Felfernig et al. 2004, p. 213-234; Felfernig et al. 2008, p. 218-226; Felfernig et al. 2009, p. 791-796; Ritov and Baron 1992, 49-62). If the user selects a repair proposal, for example, *change the desired phone style to slider*, the system is able to find a solution.

The layout of the RecoMobile phone selection page is depicted in Figure 2. The phone selection page enlists the set of phones that fulfill the given set of customer requirements. This set is ranked on the basis of similarity metrics (for details see the next section). For each mobile phone the user can activate a product fact sheet that is implemented as a direct link to the supplier's web page. Finally, it is possible to select the preferred mobile phone and to finish the session.

3 Configuring, Recommending, Ordering

In this section we will provide technical details of how RecoMobile supports configuration tasks, how the system determines repair alternatives in situations where

no solution could be found, how recommendations for features are determined, and how phones are ranked taking into account user preferences.

Supporting configuration tasks. The task of identifying a configuration for a given set of specified customer requirements can be defined as follows:

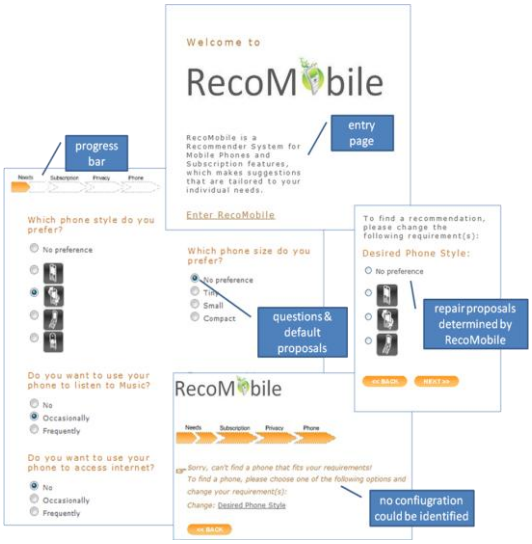

Figure 1: RecoMobile–specifying customer require ments.

Figure 2: RecoMobile–selecting a phone.

Definition 1 (configuration task): a configuration task can be defined as a constraint satisfaction problem (V, D, C). $V=\{x_0, x_1, \ldots, x_n\}$ represents a set of finite domain variables and $D=\{dom_0, dom_1, \ldots, dom_n\}$ represents a set of do-mains dom_i where dom_i is assigned to the variable x_i. Finally, $C = C_{KB} \cup C_R$ where $C_{KB} = \{c_0, c_1, \ldots, c_m\}$ represents a set of domain constraints (the configuration knowledge base) that restrict the possible combinations of values assigned to the variables in V and $C_R = \{r_0, r_1, \ldots, r_q\}$ represents a set of customer requirements.

A simple example for a configuration task is V={*styleReq*, *webUse*, *GPSReq*, *pModel*, *pStyle*, *pHSDPA*, *pGPS*} where *styleReq* expresses the user's preferred phone style, *webUse* specifies how often the user intends to access internet with the phone, and *GPSReq* specifies whether the user wants to use GPS navigation functionality.

Table 1 specifies the existing phone models (*pModel*), their styles a.k.a. form fac-tor (*pStyle*), whether the phone supports fast internet access (*pHSDPA*), and whether the phone supports GPS navigation (*pGPS*). The respective domains are D={{*any*, *bar*, *clam*}, {*no*, *occasional*, *often*}, {*false*, *true*}, {*p1*, *p2*, *p3*}, {*bar*, *clam*}, {*true*, *false*}, {*true*, *false*}}.

Table 1: Available phone models in working example.

pModel	pStyle	pHSDPA	pGPS
p1	*bar*	*false*	*false*
p2	*clam*	*true*	*true*
p3	*clam*	*true*	*false*

Furthermore, we introduce a set of domain constraints $C_{KB} = \{c_0, c_1, c_2, c_3\}$. Table 1 can be interpreted as a constraint in disjunctive normal form, which yields c_0. The remaining constraints represent the following domain properties:

- c_1: (*webUse = often*) → (*pHSDPA=true*) */* frequent web use requires a fast internet connection */*
- c_2: (*styleReq=any*) OR (*styleReq=pStyle*) */* the phone should support the user's preferred style */*
- c_3: (*GPSReq = true*) → (*pGPS = true*) */* if GPS navigation is required, it must be supported */*

Finally, an example for customer requirements is C_R = {r_0:*styleReq=clam*, r_1:*webUse=often*, r_2:*GPSReq=false*}.

On the basis of this definition of a configuration task we can now introduce the definition of a solution for a configuration task (also denoted as *configuration*).

Definition 2 (configuration): a solution (configuration) for a given configuration task (V, D, C) is represented by an instantiation I = {$x_0 = v_0$, $x_1 = v_1$, …, $x_n = v_n$}, where $v_i \in dom_i$. A configuration is *consistent* if the assignments in I are consistent with the constraints in C. Furthermore, a configuration is *complete* if all the variables in V have a concrete value. Finally, a configuration is *valid*, if it is both consistent and complete. An example for a valid configuration is the following: {*styleReq=clam*, *webUse=often*, *GPSReq=false*, *pModel=p3*, *pStyle=clam*, *pHDSPA=true*, *pGPS=false*}.

Diagnosing inconsistent requirements. In situations where no configuration can be found for a given set of requirements, we have to activate a diagnosis functionality (Felfernig et al. 2004, p. 213-234; Felfernig et al. 2008, p. 218-226; Felfernig et al. 2009, p. 791-796; Ritov and Baron 1992, 49-62). Let us assume the following set of customer requirements C_R = {r_1:*styleReq = bar*, r_2:*webUse = often*, r_3:*GPSReq = true*}. The setting in C_R does not allow the calculation of a solution; consequently, we have to identify a minimal set of requirements that has to be changed in order to be able to restore consistency. We are interested in minimal changes since we want to keep the original set of requirements the same as much as possible. The calculation of a minimal set of requirements that has to be changed is based on the determination of conflict sets (Junker 2004, p. 167-172; Reiter 1987, p. 57-95).

Recommending feature values. RecoMobile supports the recommendation of fea-ture values and the calculation of user-individual rankings for phones. To calculate recommendations for feature values, valid configurations of previous sessions are stored in a database. On the basis of these configurations RecoMobile supports two basic algorithms: *nearest neighbors* (Tiihonen and Felfernig 2009, to appear) and *Naïve Bayes voter* (Coester et al. 2002; Tiihonen and Felfernig 2009, to appear). In our evaluation of RecoMobile in the next section we focus on a comparison of near-

est neighbor based feature value recommendation with corresponding non-personalized configurator versions. An empirical evaluation of the performance of different feature recommendation algorithms is within the scope of future work.

Nearest neighbor based feature value recommendation. The idea of a nearest neighbor algorithm is to determine the neighbor configuration $conf_i$, which is closest to the active user's already specified requirements, and to recommend feature values from this nearest neighbor. The distance between the already specified user requirements and a neighbor configuration $conf_i$ is defined as the sum of individual distances (McSherry 2003, p. 291-305) between corresponding feature values, weighted by feature importance weights[1]. An example set of valid configurations from already completed configuration sessions is shown in Table 2.

To calculate distances between feature values, RecoMobile applies the *Heterogeneous Value Difference Metric* (*HVDM*) from Wilson and Martinez (1997, p. 1-34) to cope with both symbolic and numeric features. The distance values are normalized to usually be in range 0 to 1. The similarity of symbolic values in a domain is learned 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. The higher the probability of a pair of feature values to be present in identically classified configurations, the more similar these feature values are considered (Wilson, D. and Martinez 1997, p. 1-34).

Table 2: Example: valid configurations from previous sessions.

feature/configuration	$conf_1$	$conf_2$	$conf_3$
styleReq	*bar*	*clam*	*clam*
webUse	*often*	*often*	*occasional*
GPSReq	*false*	*true*	*false*
pModel	*p1*	*p2*	*p3*
pStyle	*bar*	*clam*	*clam*
pHSDPA	*false*	*true*	*true*
pGPS	*false*	*true*	*false*

A simple example for the application of the *nearest neighbor* based approach to the recommendation of feature values is the following. Table 2 contains three valid configurations $\{conf_1, conf_2, conf_3\}$ from previous configuration sessions. Let us assume that the current user has already specified the requirements $C_R = \{r_0$:*styleReq* = *clam*, r_1:*webUse*= *often*$\}$. Intuitively, the nearest neighbor for this combination of requirements is $conf_2$ since the feature values of *styleReq* and *webUse* are identical with the values specified in C_R. If we want to predict a value for the feature *GPSReq*, we would simply use the value specified in the configuration $conf_2$, i.e., *GPSReq* = *true*.

Maintaining the consistency of recommendations. Note that recommendations for feature values must be consistent with the already specified set of customer require-

[1] The weights for the RecoMobile features have been determined in a user study where participants had to estimate the importance of different phone features.

ments, i.e., if the user accepts a recommended feature value, this selection should not trigger an inconsistency and the activation of the diagnosis & repair component. RecoMobile checks the consistency of a potential recommendation before showing it to the user. In cases where none of the candidate nearest neighbors is able to provide a value that can be recommended, RecoMobile omits to recommend a feature value.

Similarity-based ranking of phones. For the ranking of phones to be presented to the user we follow a similarity-based approach. We determine the distance of each previous configuration to the user's current configuration, so that phones from nearest configurations are shown first. Phones that are compatible with user requirements are presented to the user. For example, if the customer requirements are C_R={r_0:*styleReq* = *clam*, r_1:*webUse* = *occasional*, r_2:*GPSReq* = *false*}, phones of configurations $conf_2$ and $conf_3$ meet the customer requirements. Since $conf_3$ is most similar to the user configuration the corresponding phone p3 is ranked first, before the phone of $conf_2$, p_2. p_1 cannot be recommended because its style does not match customer requirements.

After this introduction to the basic technologies integrated in the RecoMobile configurator, we will now focus on the discussion of the results of our empirical study. This study has been conducted with the goal to show the advantages of applying recommendation technologies in knowledge-based configuration settings.

4 Empirical Evaluation

We differentiate between four basic versions that differ in the way phones are ordered (*price-based* vs. *similarity-based ordering*) on the phones selection page and in the way in which dialog elements are presented (*with or without recommended feature settings*) (see Table 3). In our evaluation we compared configurator versions with recommendation support (version B and D) with configurator versions without recommendation support for feature settings (version A and C). To compare those versions we defined the hypotheses shown in Table 4. The calculation of recommendations for feature settings in version B and D was based on the data collected by configurator versions A and C.

Table 3: RecoMobile–configurator versions in user study.

	no recommended feature settings	**recommended feature settings**
price-based phone ranking	version A	version B
similarity-based phone ranking	version C	version D

We evaluated these hypotheses on the basis of a dataset collected within an on-line survey conducted at two Austrian universities (Graz and Klagenfurt), at the University of Bolzano, and the Helsinki University of Technology. N=546 subjects

participated in the study. In the scenario of the study the participants had to decide which mobile phone (including the corresponding services) they would select. Each participant was assigned to one of the four configurator versions (see Table 3) with the task to identify a mobile phone solution and to place a fictitious order.

Table 4: Overview of hypotheses (H_1..H_{10}).

id	Hypothesis
H_1	*personalized configurators increase the* ***confidence*** *of a user in his or her product decision*
H_2	*users of a personalized configurator are more* ***satisfied*** *with the quality of the configuration process*
H_3	*personalized configurators increase a user's* ***trust*** *in the presented configuration solution*
H_4	*personalized configurators* ***better support*** *users in finding the best options*
H_5	*the* ***probability of reusing*** *the configurator is higher with personalized versions*
H_6	*the* ***probability of recommending*** *the configurator to other users is higher with personalized versions*
H_7	*a user's* ***expectations regarding the solution*** *are better fulfilled with the personalized versions*
H_8	*personalized configurators trigger a higher* ***purchase probability*** *than non-personalized ones*
H_9	*the average* ***interaction time*** *per page is lower with personalized versions*
H_{10}	*defaults trigger a* ***decision bias*** *– the selected feature values differ significantly between personalized and non-personalized versions*

After interacting with the configurator, the participants had to fill out a questionnaire (answers provided on a 11-point Likert scale) that helped us to evaluate the hypotheses {H_1, H_2,…, H_8}. Hypotheses H_9 and H_{10} were evaluated directly on the basis of session interaction data (selected feature settings and session length). The major results of evaluating the hypotheses are summarized in Table 5. We can observe a tendency ($p<0.2$) that personalized configurators are able to increase the *confidence of a user in her or his product decision (H_1)*. The confidence in a user's own decision plays a major role in the final phase of the decision making process in terms of the willingness to buy a product (Felfernig et al. 2006, p. 1-22). An explanation model for this increase of confidence is the *status-quo bias* (Samuel-son and Zeckhauser 1988, p. 370-392) which argues that users are reluctant to change the current state. The current state in our case is the set of recommended feature settings. In addition, accepting the proposed settings reduces the risk of configuration inconsistent states and also the risk of malfunctioning configurations (Mandl et al. 2009, p. 69-80).

Table 5: Evaluation results for hypotheses H_1..H_9.

id	non-pers.	pers.	significance
H_1	5.33(2.94)	5.73(2.65)	$p<0.2$
H_2	5.57(2.0)	6.31(2.19)	**$p<0.05$**
H_3	4.83(2.56)	5.20(2.45)	$p<0.2$
H_4	5.05(2.68)	5.74(2.29)	**$p<0.05$**
H_5	4.43(3.11)	5.07(2.88)	$p<0.09$
H_6	4.38(2.99)	5.05(2.90)	$p<0.08$
H_7	4.67(2.33)	5.46(2.65)	**$p<0.05$**
H_8	4.24(3.44)	4.73(3.02)	$p<0.17$
H_9	3.0 min. (1.05)	3.27 min.(1.67)	$p<0.16$

Regarding the overall *quality of the configuration process* (H_2) users are significantly ($p<0.05$) more satisfied with versions that support recommendations for features (personalized versions). Such recommendations can provide support in situations where users are not sure about which value to select. In addition, they actively help users to keep partial configurations consistent, i.e., the probability of being confronted with repair situations (see, e.g., Figure 1) is reduced.

There is a tendency that users have more *trust in the presented recommendation (H_3)* when they are applying the personalized configurator version ($p < 0.2$). This result is very important since the level of trust directly correlates with a customer's willingness to buy an item (Chen and Pu 2005, p. 135-145). An explanation for this effect is that due to the personalized recommendation of feature values users are enabled to retrieve better solutions that fit their wishes and needs.

Within versions that support the automated recommendation of feature values we can observe a higher *precision* (*position of selected configuration/number of items in the result set*) of version D (mean precision value of 1.87) compared to version B (mean precision value of 2.08). A similar effect can be observed if we compare version C (1.84) with version A (2.09). Consequently, similarity-based configuration ranking outperforms price-based ranking in all cases. A study focusing on the comparison of different similarity and diversity metrics is in the focus of future work.

Personalized configurators significantly better *support* users in finding the best options ($p<0.05$, H_4). This effect can be explained by active user support provided by the automated generation of recommendations for feature settings. In addition, there is a strong evidence that a user's *willingness to reuse the configurator* is higher (H_5) if he or she used the personalized version ($p<0.09$) and users of personalized configurators have a higher probability of *recommending the configurator application to other users* ($p<0.08$, H_6). Personalized configurators are significantly better fulfilling a user's *expectations regarding the presented result* ($p<0.05$, H_7) which indicates higher-quality configurations that have been found due to more active user support. Furthermore, there exists a tendency that personalized configurators are triggering a higher *purchase probability* ($p < 0.17$, H_8) – this result is consistent with the fact that personalized configurators increase a user's trust in the configuration result. Contrary to the original assumption, there is a tendency that *configuration sessions with personalized configurators take longer* ($p<0.16$, H_9). This effect can be explained in the sense that users invest more time in evaluating the proposed feature values. Although the overall amount of cognitive efforts is slightly higher with personalized versions, the participants of the study preferred personalized configurators.

Finally, we wanted to investigate the impact of personalized feature recommendations on a user's feature selection behavior (H_{10}). The interesting result is that for many of the features (presented as questions within a configuration session) we could observe significant differences in the selection behavior depending on the configurator version users were interacting with. As an example, the evaluation results regarding the feature *fixed costs per month concerning the selected monthly minutes package* are depicted in Figure 3. The selection behavior of users interacting

with a personalized version differs significantly from the selection behavior of users interacting with a non-personalized version.

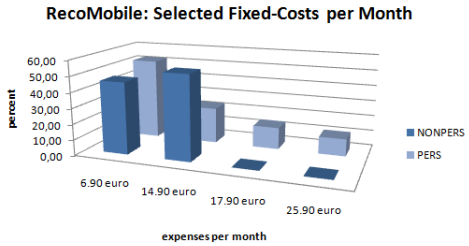

Figure 3: Selected *fixed costs per month* in personalized and non-personalized versions – the underlying distributions differ significantly ($\chi 2=67.47$).

Consequently, the recommendation of feature settings can trigger a change of selection behavior – we now want to analyze reasons for this effect. An explana-tion model for this phenomenon is that users often tend to favor the status quo compared to alternatives often of equal attractiveness (Bettman et al. 1998, p. 187-217). The tendency towards decision maintenance and reluctance to change a current state is also known as *status-quo bias* (Samuelson and Zeckhauser 1988, p. 370–392). Changes to the current state are related either to losses or to gains and typically humans are loss-averse (Bettman et al. 1998, p. 187-217; Tversky and Kahneman 1984, p. 341–350). In the case that feature recommendations are included in the presentation of options, users are then reluctant to change the current state (the recommended feature setting). The mentioned losses in the context of configurator applications could be suspected inconsistent states of the configurator application or missing product features that could be essential for a functioning product.

An overview of example features with significant different selection behavior triggered by different configurator versions is given in Table 6.

Table 6: Evaluation result for hypothesis H_{10} – example features with significant different value selections depending on the configurator version.

feature	$\chi 2$	significance
fixed costs per month	67.47	p<0.0001
phone style	111.55	p<0.0001
phone size	192.8	p<0.0001
sms support	14.86	p=0.0019

The features *fixed costs per month*, *phone style*, *phone size*, and *sms support* have significantly different selection distributions depending on the version of the configuration system (personalized or non-personalized).

5 Related Work

Main-stream recommender applications are based on collaborative filtering (Konstan et al. 1997, p. 77-87) and content-based filtering (Pazzani 1999, p. 393-408) approaches. These approaches are predominantly applied to quality and taste products – a very well known example is amazon.com (Linden et al. 2003, p. 76-80). The application of pure collaborative or content-based recommendation is the exception of the rule – in many cases only hybrid approaches can solve problems such as the ramp-up problem (e.g., for a new user the recommender system does not dispose of rating data which makes the calculation of initial recommendations a challenging task). A discussion of this and further issues regarding the deployment of recommenders can be found in Burke (2002, p. 331-370).

Configuration systems have a long and successful history in the area of Artificial Intelligence (Barker et al. 1989, p. 298–318; Fleischanderl et al. 1998, p. 59–68; Mittal and Frayman 1990, p. 1395–1401; Sabin and Weigel 1998, p. 42–49). Although these systems support interactive decision processes with the goal to determine configurations that are useful for the customer, the integration of personalization technologies has been ignored with only a few exceptions – see, for example, Coester et al. (2002); Geneste, L. and Ruet (2001, p. 4-10). The RecoMobile configurator presented in this paper has been developed and evaluated on the basis of the configuration concepts introduced from Tiihonen and Felfernig (2009, to appear). The goal of the work presented here was to implement and evaluate a system that integrates recommendation technologies that actively support users in a configuration process.

The integration of recommendation technologies with knowledge-based configuration is still in a very early stage. Most of the existing commercial configuration environments are lacking of recommendation functionalities – the study presented in this paper points out potentials for improvements. There exist some contributions that take into account the application of personalization technologies in the configuration context. Geneste and Ruet (2001, p. 4-10) introduce an approach to the integration of case-based reasoning methods (Kolodner 1993; McSherry 2003, p. 291-305; Smyth and Keane 1996, p. 127-135) with constraint solving (Junker 2004, p. 167-172) with the goal to adapt nearest neighbors identified for the current problem. There exist a couple of approaches that are similar to Geneste and Ruet (2001, p. 4-10) - see, for example, Coester et al. (2002). All of those approaches do not provide a clear concept for enabling minimal changes and handling inconsistent feature value recommendations.

In the context of our empirical study we could observe decision biases (Ritov and Baron 1992, p. 49–61) triggered by automated feature value recommendations. In the psychological literature there exist a couple of theories that explain the existence of different types of decision biases. An overview of such biases and their impact for interactive selling applications is given, for example, in Mandl et al. (2009, p. 69-80). A detailed analysis of the impact of knowledge-based recom-

mender applications regarding dimensions such as trust or intention to return to the web page has been performed from Chen and Pu (2005, p. 135-145). Among the most important results is the fact that *explanations for recommendation results can be a highly effective means to achieve a user's trust*. Similar effects have been reported from Felfernig et al. (2006, p. 1-22).

6 Conclusions

In this paper we have presented an approach to the recommendation of feature values in the context of interactive configuration dialogs. We have introduced the RecoMobile configuration environment that supports the configuring of mobile phone packages (phone and corresponding subscription features). This application has been evaluated within the scope of an empirical study. The results of this study show that our personalization approach allows significant improvements in the quality of configuration dialogs. The study as well indicates the existence of decision biases that are triggered by the personalized recommendation of feature values.

References

Barker, V., O'Connor, D., Bachant, J., and Soloway, E. (1989) Expert systems for configuration at Digital: XCON and beyond, Communications of the ACM, 32, 3, 298–318.

Bettman, J., Luce, M., and Payne, J. (1998) Constructive Consumer Choice Processes, Journal of Consumer Research 25, 3, 187-217.

Burke, R. (2002) Hybrid Recommender Systems: Survey and Experiments, UMUAI, 12(4):331_370.

Chen, L., and Pu, P. (2005) Trust Building in Recommender Agents, 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI'05), Reading, UK, pp. 135-145.

Coester, C., Gustavsson, A., Olsson, R., and Rudstroem, A. (2002) Enhancing web-based configuration with recommendations and cluster-based help, AH'02 Workshop on Recommendation and Personalized in e-Commerce, Malaga, Spain.

Felfernig, A., Friedrich, G., Jannach, D., and Stumptner, M. (2004) Consistency-based diagnosis of configuration knowledge bases, Artificial Intelligence, 2, 152, 213–234.

Felfernig, A., Gula, B., and Teppan, E. (2006) Knowledge-based Recommender Technologies for Marketing and Sales, Special issue of Personalization Techniques for Recommender Systems and Intelligent User Interfaces for the International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 21, 2, 1-22.

Felfernig, A., Friedrich, G., Teppan, E., and Isak, K. (2008) Intelligent Debugging and Repair of Utility Constraint Sets in Knowledge-based Recommender Applications, 13th ACM Intl. Conf. on Intelligent User Interfaces (IUI'08), Canary Islands, Spain, 218-226.

Felfernig, A., Friedrich, G., Schubert, M., Mandl, M., Mairitsch, M., and Teppan, E. (2009) Plausible Repairs for Inconsistent Requirements, 21st International Joint Conference on Artificial Intelligence (IJCAI'09), Pasadena, California, USA, pp. 791-796.

Fleischanderl, G., Friedrich, G., Haselboeck, A., Schreiner, H., and Stumptner, M. (1998) Configuring Large Systems Using Generative Constraint Satisfaction, IEEE Intelligent Systems, 13, 4, 59–68.

Geneste, L. and Ruet, M. (2001) Experience-based Configuration, 17th International Conference on Artificial Intelligence, Workshop on Configuration, Seattle, WA, USA, pp. 4-10.

Huffman, C. and Kahn, B. (1998) Variety for Sale: Mass Customization or Mass Confusion, Journal of Retailing, 74, pp. 491–513.

Junker, U. (2004) QuickXPlain: Preferred Explanations and Relaxations for Over-Constrained Problems. 19th National Conference on Artificial Intelligence (AAAI'04), San Jose, AAAI Press, pp. 167–172.

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

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. and Riedl, J. (1997) GroupLens: applying collaborative filtering to Usenet news Full text. Communications of the ACM, 40,3,77-87.

Linden, G., Smith, B., and York, J. (2003) Amazon.com recommendations: Item_to_Item Collaborative Filtering, IEEE Internet Computing, 7(1):76_80.

Mandl, M., Felfernig, A., and Schubert, M. (2009) Consumer Decision Making in Knowledge-based Recommendation, 5th International Conference on Active Media Technology, p. 69-80, LNCS, Beijing, China.

Mittal, S. and Frayman, F. (1990) Towards a Generic Model of Configuration Tasks, 11th International Joint Conference on Artificial Intelligence, Detroit, MI, pp. 1395–1401.

Pazzani, M. (1999) A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 13(5-6):393–408.

Reiter, R. (1987) A theory of diagnosis from first principles. AI Journal, 23(1):57–95.

Ritov, I. and Baron, J. (1992) Status-quo and omission biases, Journal of Risk and Uncertainty 5,2, 49–61.

Sabin, D. and Weigel, R. (1998) Product Configuration Frameworks – A Survey, IEEE Intelligent Systems, 13, 4, pp. 42–49.

Samuelson, W. and Zeckhauser, R. (1988) Status quo bias in decision making, Journal of Risk and Uncertainty 108, 2, 370–392.

McSherry, D. (2003) Similarity and Compromise. Intl. Conference on Case-based Reasoning (ICCBR'03), pages 291-305, Trondheim, Norway.

Smyth, B., and Keane, M. (1996) Using Adaptation Knowledge to Retrieve and Adapt Design Cases, Journal of Knowledge-based Systems, 9, 2, 127-135.

Stumptner, M. (1997) An overview of knowledge-based configuration, AICOM, 10, 2, 111–126.

Tiihonen, J. and Felfernig, A. (2009) Towards Recommending Configurable Offerings, International Journal of Mass Customization, to appear.

Tversky, A. and Kahneman, D. (1984) Choices, values, and frames, American Psychologist 39, 341–350.

Wilson, D. and Martinez, T. (1997) Improved Heterogenous Distance Functions, Journal of Artificial Intelligence Research, 6, 1-34.