Publication IV

Felfernig, A. Mandl, M., Tiihonen, J. Schubert, M., and Leitner G. (2010). Personalized user interfaces for product configuration. 15th international conference on intelligent user interfaces (IUI '10), Hong Kong, China, 317-320. doi 10.1145/1719970.1720020.

© 2010 ACM.
Reprinted with permission

Personalized User Interfaces for Product Configuration

Alexander Felfernig
Graz University of Technology
Applied Software Engineering
Inffeldgasse 16b, A-8010 Graz
alexander.felfernig@ist.tugraz.at

Monika Mandl
Graz University of Technology
Applied Software Engineering
Inffeldgasse 16b, A-8010 Graz
monika.mandl@ist.tugraz.at

Juha Tiihonen
Helsinki University of Technology
Computer Science
and Engineering
02015 TKK, Finland
juha.tiihonen@tkk.fi

Monika Schubert
Graz University of Technology
Applied Software Engineering
Inffeldgasse 16b,
A-8010 Graz
monika.schubert@ist.tugraz.at

Gerhard Leitner
University of Klagenfurt
Interactive Systems
Universitätsstrasse 65-67,
A-9020 Klagenfurt
gerhard.leitner@uni-klu.ac.at

ABSTRACT
Configuration technologies are well established as a foundation of mass customization which is a production paradigm that supports the manufacturing of highly-variant products under pricing conditions similar to mass production. A side-effect of the high diversity of products offered by a configurator is that the complexity of the alternatives may outstrip a user's capability to explore them and make a buying decision. In order to improve the quality of configuration processes, we combine knowledge-based configuration with collaborative and content-based recommendation algorithms. In this paper we present configuration techniques that recommend personalized default values to users. Results of an empirical study show improvements in terms of, for example, user satisfaction or the quality of the configuration process.

Author Keywords
Configuration systems, recommender systems, model-based diagnosis.

ACM Classification Keywords
I.2.5.Expert system tools and techniques.

General Terms
Human Factors, Design, Algorithms

INTRODUCTION
Configuration systems have a long tradition as a successful application area of Artificial Intelligence, see, for example, [1,9,15,18,22]. On an informal level, configuration can be interpreted 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* [18]. 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 [1], 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. 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 [2] 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 [23].

In this paper we present functionalities that support personalized configuration of mobile phones and corresponding subscriptions. Our major contribution is the integration of recommendation technologies with knowledge-based configuration (a functionality that is not available in commercial systems).

The remainder of the paper is organized as follows. In the next section we present the recommendation approaches useful for supporting personalized configuration. Thereafter we shortly discuss results of empirical evaluations. Finally, we discuss related work and conclude the paper.

CONFIGURING, RECOMMENDING, ORDERING
In this section we will provide technical details that help to understand how our prototype implementation determines

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IUI'10, February 7–10, 2010, Hong Kong, China.
Copyright 2010 ACM 978-1-60558-515-4/10/02...$10.00.

repair alternatives in situations where no solution can 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:

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 domains 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 factor (*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*}}.

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

Table 1: Available phone models in working example.

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*) $\rightarrow$ (*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 phone style* */
- c_3: (*GPSReq = true*) $\rightarrow$ (*pGPS = true*) /* *if GPS navigation is required, the phone must support it* */

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 $I = \{x_0 = v_0, x_1 = v_1, \ldots, x_n = v_n\}$, 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 [6,7,8,17]. 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 (see the following definition) [11,17].

Definition 3 (conflict set): a conflict set is a set $CS \subseteq C_R$ s.t. $C_{KB} \cup CS$ does not allow the calculation of a solution. Furthermore, CS is said to be minimal if there does not exist a set CS' with CS' $\subset$ CS.

In our working example we can identify the two minimal conflict sets $CS_1=\{r_1, r_2\}$ and $CS_2=\{r_1, r_3\}$. Both are conflict sets since $\{r_1, r_2\} \cup C_{KB}$ as well as $\{r_1, r_3\} \cup C_{KB}$ is inconsistent. Furthermore, both conflict sets are minimal since for both there does not exist a proper subset with the conflict set property (see Definition 3). In order to restore consistency, we have to resolve each of the identified minimal conflict sets. A systematic way to this is to apply the concept of model-based diagnosis [17]. A *customer requirements* (*CR*) *diagnosis problem* and a corresponding *CR diagnosis* can be defined as follows.

Definition 4 (CR diagnosis problem and CR diagnosis): a *CR diagnosis problem* is defined as a tuple (C_{KB}, C_R) where C_R is a set of requirements and C_{KB} represents the constraints of the configuration knowledge base. A *diagnosis* for (C_{KB}, C_R) is a set $d \subseteq C_R$, s.t. $C_{KB} \cup (C_R - d)$ is consistent. A diagnosis is *minimal* if there does not exist a diagnosis $d' \subset d$, s.t. $C_{KB} \cup (C_R - d')$ is consistent.

For determining the complete set of minimal diagnoses we can apply the algorithm proposed by [17]. The core of the concept presented in [17] is the Hitting Set Directed Acyclic Graph (HSDAG) algorithm that is complete in the sense that all the existing diagnoses are found.

Recommending feature values. Beside the calculation of diagnoses in the case that no solution could be found, the recommendation of feature values and the calculation of user-individual rankings for phones are important functionalities. To calculate recommendations for feature values, valid configurations of previous sessions are stored

in a database. On the basis of these configurations two basic algorithms are supported in our prototype environment: *nearest neighbors* (see [23]) and *Naïve Bayes voter* [5,23]. The *Naïve Bayes voter* is discussed in detail in [5,23] and is taken into account in this paper. An empirical evaluation of the performance of different feature recommendation algorithms is a goal 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 [20] between corresponding feature values, weighted by feature importance weights.

To calculate distances between feature values, *Heterogeneous Value Difference Metric* (*HVDM*) [25] can be applied which help to cope with both symbolic and numeric features. The individual similarity metric to be used for calculating the similarity between two feature values is chosen depending on the basic characteristic of the feature (*less is better*, *more is better*, or *nearer is better* – for details see, for example, [20]). The distance values are normalized to usually be in range 0 to 1. The similarity of symbolic values in a domain is learned automatically [25]. 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 [25].

Maintaining the consistency of recommendations. Note that recommendations for feature values must be consistent with the already specified set of customer requirements, 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. In cases where none of the candidate nearest neighbors is able to provide a value that can be recommended, feature recommendation can be omitted.

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.

FIRST EMPIRICAL RESULTS

We have evaluated our prototype configuration environment within the scope of an empirical study with n>500 participants.[1] This study showed significant improvements in terms of qualitative measures such as *trust in a configuration* or the *willingness to buy* [4] as well as in terms of measures such as *prediction quality* of the used similarity measures (precision). In all those dimensions personalized configurator versions outperformed non-personalized versions as they are still in use in commercial environments. One of the major results of this initial study is a clear observation of *decision biases* where selection probabilities significantly changed depending on the configurator version.

RELATED AND FUTURE WORK

Main-stream recommender applications are based on collaborative filtering [13] and content-based filtering [16] approaches. These approaches are predominantly applied to quality and taste products – a very well known example is amazon.com [14]. 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 [3].

Configuration systems have a long and successful history in the area of Artificial Intelligence [1,9,15,18,22]. 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, [5,10]. 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. The authors of [10] introduce an approach to the integration of case-based reasoning methods [12,21] with constraint solving [11] with the goal to adapt nearest neighbors identified for the current problem. There exist a couple of approaches that are similar to [10] – see, for example, [5]. All of those approaches do not provide a clear concept for enabling minimal changes and handling inconsistent feature value recommendations.

[1] A detailed discussion of these results has been omitted due to space limitations and will be provided in an extended version of this paper.

CONCLUSIONS

In this paper we provided an overview of basic recommendation techniques that can be used in the context of configuring complex products and services. These techniques show to be useful in terms of improving the user acceptance of the configurator interface.

ACKNOWLEDGEMENTS

The work presented in this paper has been conducted within the research projects WECARE (Austrian Research Promotion Agency) and COSMOS (TEKES Finland).

REFERENCES

1. Barker, V., O'Connor, D., Bachant, J., and Soloway, E. Expert systems for configuration at Digital: XCON and beyond, Communications of the ACM, 32, 3 (1989), 298–318.
2. Bettman, J., Luce, M., and Payne, J. Constructive Consumer Choice Processes, Journal of Consumer Research 25, 3 (1998), 187-217.
3. Burke, R. Hybrid Recommender Systems: Survey and Experiments, Journal of User Modeling and User-Adapted Interaction (UMUAI), 12(4):331□370, 2002.
4. Chen, L., and Pu, P. Trust Building in Recommender Agents, 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI'05), Reading, UK, 2005, pp. 135-145.
5. Coester, C., Gustavsson, A., Olsson, R., and Rudstroem, A. Enhancing web-based configuration with recommendations and cluster-based help, AH'02 Workshop on Recommendation and Personalized in e-Commerce, 2002, Malaga, Spain.
6. Felfernig, A., Friedrich, G., Jannach, D., and Stumptner, M. Consistency-based diagnosis of configuration knowledge bases, Artificial Intelligence, 2, 152 (2004), 213–234.
7. Felfernig, A., Friedrich, G., Teppan, E., and Isak, K. Intelligent Debugging and Repair of Utility Constraint Sets in Knowledge-based Recommender Applications, 13th ACM International Conference on Intelligent User Interfaces (IUI'08), 2008, Canary Islands, Spain, 218-226.
8. Felfernig, A., Friedrich, G., Schubert, M., Mandl, M., Mairitsch, M., and Teppan, E. Plausible Repairs for Inconsistent Requirements, 21st International Joint Conference on Artificial Intelligence (IJCAI'09), Pasadena, California, USA, 2009, pp. 791-796.
9. Fleischanderl, G., Friedrich, G., Haselboeck, A., Schreiner, H., and Stumptner, M. Configuring Large Systems Using Generative Constraint Satisfaction, IEEE Intelligent Systems, 13, 4 (1998), 59–68.
10. Geneste, L. and Ruet, M. Experience-based Configuration, 17th International Conference on Artificial Intelligence, Workshop on Configuration, Seattle, WA, USA, 2001, pp. 4-10.
11. Junker, U. QuickXPlain: Preferred Explanations and Relaxations for Over-Constrained Problems. 19th National Conference on Artificial Intelligence (AAAI'04), San Jose, AAAI Press, 2004, pp. 167–172.
12. Kolodner, J. Case-based Reasoning, Morgan Kaufmann Publishers, 1993.
13. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. and Riedl, J. GroupLens: applying collaborative filtering to Usenet news Full text. Communications of the ACM, 40,3 (1997),77-87.
14. Linden, G., Smith, B., and York, J. Amazon.com recommendations: Item□to□Item Collaborative Filtering, IEEE Internet Computing, 7(1):76□80, 2003.
15. Mittal, S. and Frayman, F. Towards a Generic Model of Configuration Tasks, 11th International Joint Conference on Artificial Intelligence, Detroit, MI, 1990, pp. 1395–1401.
16. Pazzani, M. A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 1999, 13(5-6):393–408.
17. Reiter, R. A theory of diagnosis from first principles. AI Journal, 23(1):57–95, 1987.
18. Sabin, D. and Weigel, R. Product Configuration Frameworks – A Survey, IEEE Intelligent Systems, 13, 4 (1998), pp. 42–49.
19. Samuelson, W. and Zeckhauser, R. Status quo bias in decision making, Journal of Risk and Uncertainty 108, 2 (1988), 370–392.
20. McSherry, D. Similarity and Compromise. Intl. Conference on Case-based Reasoning (ICCBR'03), pages 291-305, 2003, Trondheim, Norway.
21. Smyth, B., and Keane, M. Using Adaptation Knowledge to Retrieve and Adapt Design Cases, Journal of Knowledge-based Systems, 9, 2 (1996), 127-135.
22. Stumptner, M. An overview of knowledge-based configuration, AI Communications (AICOM), 10, 2 (1997), 111–126.
23. Tiihonen, J. and Felfernig, A. Towards Recommending Configurable Offerings, International Journal of Mass Customization, to appear, 2009.
24. Tversky, A. and Kahneman, D. Choices, values, and frames, American Psychologist 39 (1984), 341–350.
25. Wilson, D. and Martinez, T. Improved Heterogenous Distance Functions, Journal of Artificial Intelligence Research, 6 (1997), 1-34.