# Towards Group-Based Configuration

**Alexander Felfernig**[1] and **Müslüm Atas**[1] and **Thi Ngoc Trang Tran**[1] and **Martin Stettinger**[1]

**Abstract.** Group-based configuration is a new configuration approach that supports scenarios in which a group of users is in charge of configuring a product or service. In this paper, we introduce a definition of a group-based configuration task and a corresponding solution. Furthermore, we show how inconsistent situations in group-based configuration can be resolved to achieve consensus within the group. We introduce these concepts on the basis of a working example from the domain of (group-based) software release planning.

## 1 Introduction

Configuration [1, 16] is considered as one of the most successful applications of Artificial Intelligence technologies. It is applied in many domains such as financial services [2], telecommunication [5], and the furniture industry [7]. Configuration environments are typically single-user oriented, i.e., the underlying assumption is that a specific user is in charge of completing the configuration task. However, considering configuration as a single user task can lead to suboptimal decisions [4]. For example, release planning is a task that typically requires the engagement of a group of stakeholders where the knowledge and preferences of all stakeholders should be taken into account in order to be able to achieve high-quality decisions [4, 6].

There are various scenarios where configuration decisions are not taken by a single person but by a group of users [3]. As mentioned, *Software Release Planning* [4] is a requirements engineering related task, where groups of users (stakeholders) are deciding about the ordering in which requirements should be implemented. In this scenario, stakeholders have different preferences and knowledge regarding the implementation alternatives. Consequently, requirements-related knowledge should be exchanged as much as possible and existing contradictions in preferences and evaluations have to be resolved. *Holiday Planning* [8] is another scenario where a group is in charge of identifying a configuration that is accepted by all group members – examples of related decisions are region to visit, hotel, and activities during the stay. *Product Line Scoping* [14] is related to the task of determining boundaries in a product line. This task is a specific type of requirements engineering task and related decisions are crucial for the success of a whole product line effort. *Investment Decisions* (e.g., project funding) [3] are often taken by a group of users who have to take into account constraints with regard to the overall amount of money that can be invested and the topics projects should deal with. The overall configuration task in this context is to identify a bundle of project proposals that takes into account the financial limits and includes high-quality proposals.

Existing configuration environments do not take into account the aspect of group configuration [3]. In contrast, for non-configurable items such as movies, restaurants, personnel decisions, and music, there already exist proposals how to support related group decision processes [11, 12, 15]. In this context, group recommendation heuristics [12] are applied to support groups in their decision making activities. In order to achieve consensus, different decision heuristics are applied which propose decisions acceptable for a group as a whole. For example, the *least misery* heuristic proposes alternatives which do not represent an absolute no-go for at least one of the group members. Besides decision heuristics, standard recommendation approaches [9] such as matrix factorization can be applied to predict recommendations acceptable for a group as whole. These approaches rely on existing group recommendations. Based on such information about group selection behavior, corresponding recommendations can be determined for similar groups.

In this paper, we focus on introducing a formal definition of a group configuration problem and show how inconsistencies in the preferences of group members can be resolved.[2] The remainder of this paper is organized as follows. In Section 2 we introduce a basic definition of a group-based configuration task and introduce a corresponding example configuration knowledge base. In Section 3 we discuss approaches that can help to resolve inconsistencies in the preferences of individual group members. In Section 4 we discuss further issues for future work. With Section 5 we conclude the paper.

## 2 Group-Based Configuration

In the following, we introduce definitions of a group configuration task and a corresponding solution. These definitions are based on a Constraint Satisfaction Problem (CSP) [17] which is frequently used for the definition of (single user) configuration tasks. The major characteristic of group-based configuration compared to other types of group decision tasks is that the alternatives are defined in terms of a knowledge base, i.e., the alternatives are not pre-specified. This requires new approaches to configuration and diagnosis search, and to represent the configuration task in a corresponding user interface.

*Definition 1: Group-based Configuration Task.* A group-based configuration task can be defined as a CSP ($V$,$D$,$C$) where $V$ is a set of variables, $D$ represents the corresponding domain definitions, and $C = PREF \cup CKB$ represents a set of constraints. In this context, $PREF = \bigcup PREF_i$ is the union of customer preferences $PREF_i$ and $CKB$ represents a configuration knowledge base.[3]

*Definition 2: Group-based Configuration.* A group-based configuration (solution) for a group-based configuration task is a complete set of assignments $CONF = \bigcup a_i : v_i = v_{ai}$ to the variables $v_i \in V$ such that $CONF \cup PREF \cup CKB$ is consistent.

[1] Graz University of Technology, Austria, email: {alexander.felfernig,muatas,ttrang,mstettinger}@ist.tugraz.at

[2] The work presented in this paper has been developed within the scope of the WEWANT project (Enabling Technologies for Group-based Configuration) which is funded by the Austrian Research Promotion Agency (850702).

[3] We denote *customer requirements* as *preferences* ($PREFS$) in order to distinguish these from *software requirements* in the working example.

*Example 1: Group-based Configuration Task*. For demonstration purposes, we introduce a simplified group-based configuration task from the domain of *software release planning*. The goal of software release planning is to assign to each software requirement a corresponding release. In this example, 9 requirements are represented in terms of variables $V = req_1, req_2, .., req_9$ and releases are represented as variable domains. If we assume that *three releases* have been planned for completing the whole software (i.e., implementing each individual requirement), each variable has a corresponding domain [1 .. 3], e.g., dom($r_1$) = [1 .. 3]. For the purpose of this example, we assume the existence of *three* stakeholders who are in charge of release planning – $PREF_i$ represents the preferences of stakeholder $i$.

The following is a complete specification of a group-based configuration task. In this task, the individual user requirements $PREF_i$ are consistent, i.e., a corresponding solution (software release plan) can be identified.[4] The configuration knowledge base $CKB$ includes additional constraints that describe dependencies between different software requirements $req_i$, for example, $req_1 > req_5$ denotes the fact that requirement $req_1$ must be implemented for $req_5$, i.e., there is a dependency between these requirements. Furthermore, the requirements $req_3$ and $req_4$ must not be implemented in the same release (e.g., due to resource constraints).

- $V$ = $\{req_1, .., req_9\}$
- $D$ = {dom($req_1$) = [1 .. 3], .., dom($req_9$ = [1 .. 3])}
- $PREF_1$ = $\{pref_{11} : req_1 = 1, pref_{12} : req_2 = 1, pref_{13} : req_3 = 1, pref_{14} : req_5 = 2, pref_{15} : req_8 = 3\}$
- $PREF_2$ = $\{pref_{21} : req_3 = 1, pref_{22} : req_4 = 2, pref_{23} : req_6 = 3, pref_{24} : req_7 = 3\}$
- $PREF_3$ = $\{pref_{31} : req_5 = 2, pref_{32} : req_6 = 3, pref_{33} : req_8 = 3, pref_{34} : req_9 = 2\}$
- $CKB$ = $\{c_1 : req_1 < req_5, c_2 : req_2 < req_8, c_3 : req_3 < req_6, c_4 : req_3 \neq req_4\}$

*Example 2: Group-based Configuration*. On the basis of the example group-based configuration task, a constraint solver could determine the following solution: $CONF$ = $\{a_1 : req_1 = 1, a_2 : req_2 = 1, a_3 : req_3 = 1, a_4 : req_4 = 2, a_5 : req_5 = 2, a_6 : req_6 = 3, a_7 : req_7 = 3, a_8 : req_8 = 3, a_9 : req_9 = 2\}$. For each requirement, the constraint solver proposes a corresponding release in the context of which the requirement should be implemented.

## 3 Resolving Inconsistencies in Group Preferences

In the example introduced in Section 2, the basic assumption is that the preferences of individual group members are *consistent*. However, in group-based configuration scenarios it happens quite often that the preferences of individual users differ. In the context of release planning scenarios, it is often the case that stakeholders have different preferences regarding the implementation of specific requirements. One requirement could be favored due to the fact that the corresponding functionalities are needed by the stakeholder. Another reason could be that a stakeholder has no preferences or simply does not understand the requirement in detail. Inconsistencies between preferences can be manually resolved by showing inconsistent preferences to stakeholders and let them decide which changes should be performed. In such scenarios, minimal conflict sets are determined [10] and conflict resolution is performed by users in a manual fashion.

[4] In Section 3 we discuss approaches to deal with inconsistencies.

Alternatively, conflicts between requirements can be resolved automatically by calculating minimal diagnoses (Definition 4) for minimal conflict sets (Definition 3).

*Definition 3: Conflict Set*. A conflict set $CS \subseteq \bigcup REQ_i$ is a minimal set of requirements such that inconsistent($CS$). CS is minimal if there does not exist a conflict set $CS'$ with $CS'$ is a conflict set and $CS' \subset CS$.

Minimal conflict sets can be exploited for determining the corresponding diagnoses [13]. Assuming that $\bigcup PREF_i \cup CKB$ is inconsistent, a minimal diagnosis (Definition 4) represents a minimal set of requirements that have to be deleted from $\bigcup PREF_i$ such that a solution can be found for the remaining constraints (see Definition 4).

*Definition 4: Group-based Configuration Diagnosis Task*. A group-based configuration diagnosis task is defined by a group-based configuration task ($V, D, C = PREF \cup CKB$) where $PREF \cup CKB$ is inconsistent.

*Definition 5: Group-based Configuration Diagnosis*. A diagnosis for a given group-based configuration task ($V, D, C = PREF \cup CKB$) is a set $\Delta$ such that $CKB \cup PREF - \Delta$ is consistent. $\Delta$ is minimal if $\neg\exists\ \Delta'$: $\Delta' \subseteq \Delta$.

*Example 3: Group-based Configuration Diagnosis Task*. An example group-based configuration task that includes inconsistencies between different user requirements is the following.

- $V$ = $\{req_1, .., req_9\}$
- $D$ = {dom($req_1$) = [1 .. 3], .., dom($req_9$) = [1 .. 3]}
- $PREF_1$ = $\{pref_{11} : req_1 = 2, pref_{12} : req_2 = 1, pref_{13} : req_3 = 1, pref_{14} : req_5 = 2, pref_{15} : req_8 = 3\}$
- $PREF_2$ = $\{pref_{21} : req_3 = 2, pref_{22} : req_4 = 3, pref_{23} : req_6 = 3, pref_{24} : req_7 = 3\}$
- $PREF_3$ = $\{pref_{31} : req_5 = 2, pref_{32} : req_6 = 3, pref_{33} : req_8 = 3, pref_{34} : req_9 = 2\}$
- $CKB$ = $\{c_1 : req_2 > req_1, c_2 : req_2 < req_8, c_3 : req_3 < req_6, c_4 : req_3 \neq req_4\}$

In this example, the requirements of the first stakeholder are inconsistent since the combination $req_1 = 2$ and $req_2 = 1$ is inconsistent with the underlying knowledge base ($req_2 > req_1$). Furthermore, there exists an inconsistency between the requirements $req_3 = 1$ (stakeholder 1) and $req_3 = 2$ (stakeholder 2).

The minimal conflict sets that can be derived from our working example are the following: $CS_1 = \{pref_{11}, pref_{12}\}$ and $CS_2 = \{pref_{13}, pref_{21}\}$. The corresponding set of alternative diagnoses (hitting sets) is the following: $\Delta_1 = \{pref_{11}, pref_{13}\}$, $\Delta_2 = \{pref_{11}, pref_{21}\}$, $\Delta_3 = \{pref_{12}, pref_{13}\}$, and $\Delta_4 = \{pref_{12}, pref_{21}\}$. A diagnosis is a minimal set of requirements from $\bigcup PREF_i$ such that $CKB \cup PREF - \Delta$ is consistent.

Diagnoses represent a set of consistency-preserving delete operations that can be applied to the set $\bigcup PREF_i$ in the case that $PREF \cup CKB$ is inconsistent. In many cases, there exist different diagnoses that can be recommended for preserving the consistency between user requirements and the configuration knowledge base ($CKB$). A ranking of alternative diagnoses in the context of group configuration scenarios can be achieved, for example, by determining a candidate set of minimal diagnoses that is then ranked on the basis of different types of group decision heuristics [12].

An example of the application of such group decision heuristics will be discussed in the following. Table 1 depicts a situation where individual user requirements are inconsistent. In order to resolve this inconsistency, the alternative diagnoses $\Delta_1, \Delta_2, \Delta_3$, and $\Delta_4$ can be applied. An open question in this context is which of the alternative

| stake-holder | $req_1$ | $req_2$ | $req_3$ | $req_4$ | $req_5$ | $req_6$ | $req_7$ | $req_8$ | $req_9$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | $pref_{11}: req_1 = 2$ | $pref_{12}: req_2 = 1$ | $pref_{13}: req_3 = 1$ | | $pref_{14}: req_5 = 2$ | | | $pref_{15}: req_8 = 3$ | |
| 2 | | | $pref_{21}: req_3 = 2$ | $pref_{22}: req_4 = 3$ | | $pref_{23}: req_6 = 3$ | $pref_{24}: req_7 = 3$ | | |
| 3 | | | | | $pref_{31}: req_5 = 2$ | $pref_{32}: req_6 = 3$ | | $pref_{33}: req_8 = 3$ | $pref_{34}: req_9 = 2$ |

**Table 1.** Tabular representation of constraints in an example group-based configuration task. Conflict set $CS_1 = \{pref_{11}, pref_{12}\}$ reflects inconsistent preferences of stakeholder 1 (the preferences are inconsistent with the configuration knowledge base) and conflict set $CS_2 = \{pref_{13}, pref_{21}\}$ reflects a conflict between the preferences of stakeholders 1 and 2.

diagnoses should be recommended first to the group of users – Table 2 summarizes the impact of the different diagnoses on the current preferences of stakeholders (users). For this purpose, different group decision heuristics can be applied that help to figure out alternatives acceptable for the whole group.

In the following, we exemplify three basic heuristics and show how these can influence the selection of a diagnosis. First, the *least misery* heuristic prefers alternatives (in our case diagnoses) that minimize the misery of individual users (see Formula 1 – $pref_\delta(s, \Delta)$ denotes the number of preferences that have to be changed by user $s$ in the context of diagnosis $\Delta$). In our scenario, least misery for a whole group would reflect the minimum of the maximum number of preferences part of a diagnosis, i.e., the lower the least misery value the better the corresponding diagnosis. For example, if diagnosis $\Delta_2$ is recommended, user 1 would have to adapt two of his/her requirements and user 2 would have to adapt zero. Diagnosis $\Delta_2$ has a lower misery value since the maximum number of requirements to be adapted is 1. Obviously, user 3 is in the situation of not being affected by any of the diagnosis candidates. Second, the *average* heuristic prefers alternatives with the lowest average deviation from the original preferences (see Formula 2). Finally, the *most pleasure* heuristic prefers alternatives with the best outcome for one user (see Formula 3). For example, in Table 1 the most pleasure value of all diagnoses $\Delta_i$ is 0.0 since for user 3 there does not exist a need to adapt his/her preferences in all of the diagnoses. For a detailed discussion of group decision heuristics we refer to [12].

$$leastmisery(\Delta) = argmax_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d \quad (1)$$

$$average(\Delta) = \frac{\Sigma_{s \in users} pref_\delta(s, \Delta)}{\#users} \quad (2)$$

$$mostpleasure(\Delta) = argmin_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d \quad (3)$$

| stakeholder | $\Delta_1 = \{r_{11}, r_{13}\}$ | $\Delta_2 = \{r_{11}, r_{21}\}$ | $\Delta_3 = \{r_{12}, r_{13}\}$ | $\Delta_4 = \{r_{12}, r_{21}\}$ |
|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 |

**Table 2.** Overview of the impact of the different diagnoses $\Delta_i$ on the current preferences of stakeholders, for example, stakeholder 1 has to change two of his/her requirements if diagnosis $\Delta_1$ gets selected.

| heuristic | $\Delta_1 = \{r_{11}, r_{13}\}$ | $\Delta_2 = \{r_{11}, r_{21}\}$ | $\Delta_3 = \{r_{12}, r_{13}\}$ | $\Delta_4 = \{r_{12}, r_{21}\}$ |
|---|---|---|---|---|
| least misery | 2.0 | 1.0 | 2.0 | 1.0 |
| average | 0.67 | 0.67 | 0.67 | 0.67 |
| most pleasure | 0.0 | 0.0 | 0.0 | 0.0 |

**Table 3.** Evaluation of the different diagnoses using the *least misery*, *average*, and the *most pleasure* heuristic. In all three heuristics the ranking criteria for the diagnoses is *less is better*.

## 4 Future Work

The major goal of this paper is to present our initial ideas related to the implementation of group-based configuration technologies. There are a couple of issues to be solved within the scope of future work - these issues will be discussed in the following paragraphs.

*Consensus in Group Decision Making*. Presenting diagnoses in situations where user preferences are inconsistent with the underlying configuration knowledge base and/or the preferences of other users is a basic means to trigger discussions and achieve consensus [4]. However, further aspects have to be taken into account in order to be able to accelerate the achievement of consensus in group decision making. Approaches that are promising in this context are, for example, the following. User interfaces have to be enriched in order to allow basic negotiation mechanisms between users. An example thereof is the following: stakeholder $A$ is interested in having implemented requirement $req_a$ as soon as possible. Furthermore, stakeholder $B$ is interested in having implemented requirement $req_b$ as soon as possible. Stakeholder $A$ would accept an earlier implementation of $req_b$ if stakeholder $B$ accepts an earlier implementation of requirement $req_a$. In this context, visualization concepts for the representation of the current decision situation will play a major role – alternative ways to represent decision situations are a focus of future work.

*Fairness in Group Decision Making*. An important issue in group decision making is fairness with regard to group members. Fairness is especially a topic within the scope of repeated decision processes where the same or similar groups are taking a decision. A related example is holiday decisions where a group of friends decides about a new travel destination and related activities. The preferences of users who were discriminated to some extent in previous year's travel arrangements should have a higher emphasis in the new holiday decision. Fairness also includes visualization aspects since the visualization of the current state of the decision process could help to increase fairness in group decision making, for example, by increasingly taking into account the preferences of other group members.

*Predictive Search*. Based on the information about already completed group decision processes, diagnosis and repair could be im-

proved by better predicting alternatives acceptable for the whole group. In this context, different types of personalization approaches should be included that help to take into account the preferences of the whole group when determining diagnoses and corresponding repair actions. Diagnosis prediction approaches for single users are already discussed in related work [1], however, in group decision scenarios further related aspects have to be taken into account. The prediction of a relevant diagnosis does not only have to take into account the selection behavior of users but also how users interacted with each other within the scope of a group decision process. Furthermore, the search for alternative configurations has to take into account group preferences, i.e., search heuristics must be learned on the basis of past group interactions.

*Negotiation Mechanisms*. The main challenge of negotiation mechanisms is to include these in a way that is easy to understand for users. Complex negotiation mechanisms will not be accepted by end-users, i.e., the major challenge is to propose decision and negotiation mechanisms that help to achieve high-quality decisions and consensus as soon as possible and to trigger inconsistency management only in situations where real disagreements exists. For example, if one stakeholder evaluates the risk level of a requirement with 7 (on a scale [1..10]) and the other stakeholder evaluates the same requirement with 8, there seems to be no real disagreement and the system may not have to point out an existing inconsistency.

*Intelligent User Interfaces*. Since group-based configuration tasks are solved in a distributed and asynchronous fashion, user interfaces should be able to take into account this situation. Figure 1 includes a screenshot of the CHOICLA group decision support environment [15].[5] In its current version, the system supports group decisions related to non-configurable products and services (e.g., party locations and type of dinner), i.e., decisions are taken with regard to a collected assortment of alternatives but are not taken with regard to certain attributes (variables) which are basic elements of a configuration task. In the current version of CHOICLA, the only possibility of taking decisions regarding configurable products is to enumerate a representative set of alternatives (e.g. new family car). In future versions of CHOICLA, we will support the integration of complete configuration tasks into decision processes. Variables will then be represented as alternatives and user preferences and inconsistencies will be represented on a corresponding graphical level.

## 5 Conclusions

In this paper, we introduced the concept of *group-based configuration*. We introduced a basic definition of a group-based configuration task (represented as a constraint satisfaction problem) and showed how to deal with inconsistent preferences of group members on the basis of the concepts of model-based diagnosis. In this context, we showed how to integrate different types of decision heuristics into diagnosis selection processes. Finally, we discussed different challenges for future work we want to tackle.

## REFERENCES


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

[2] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, 'The VITA Financial Services Sales Support Environment', in *AAAI/IAAI 2007*, pp. 1692–1699, Vancouver, Canada, (2007).


[5] www.choicla.com.

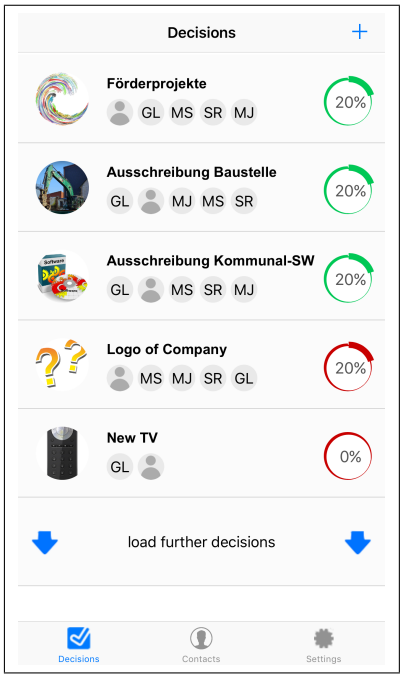


**Figure 1.** CHOICLA group decision support environment. Each entry represents a group decision task – the corresponding percentages indicate the share of users who already articulated their requirements. A red circle indicates the fact that the current user did not articulate his/her preferences.


[3] A. Felfernig, M. Stettinger, G. Ninaus, M. Jeran, S. Reiterer, A. Falkner, G. Leitner, and J. Tiihonen, 'Towards open configuration', in *16th Intl Workshop on Configuration*, pp. 89–94, Novi Sad, Serbia, (2014).

[4] A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maaleij, D. Pagano, L. Weninger, and F. Reinfrank, 'Group Decision Support for Requirements Negotiation', in *Advances in User Modeling, Springer Verlag*, volume 7138 of *LNCS*, pp. 105–116, (2012).

[5] Gerhard Fleischanderl, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner, 'Configuring large systems using generative constraint satisfaction', *IEEE Intelligent Systems*, **13**(4), 59–68, (1998).

[6] T. Greitemeyer and S. Schulz-Hardt, 'Preference-consistent evaluation of information in the hidden profile paradigm: Beyond group-level explanations for the dominance of shared information in group decisions.', *Jrnl of Personality & Soc Psychology 84(2)*, 332–339, (2003).

[7] A. Haag, 'Sales Configuration in Business Processes', *IEEE Intelligent Systems*, **13**(4), 78–85, (1998).

[8] A. Jameson, S. Baldes, and T. Kleinbauer, 'Two methods for enhancing mutual awareness in a group recommender system', in *ACM Intl. Working Conf. on Advanced Vis. Interf.*, pp. 48–54, Gallipoli, Italy, (2004).

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems – An Introduction*, Cambridge University Press, 2010.

[10] U. Junker, 'QuickXPlain: Preferred Explanations and Relaxations for Over-Constrained Problems', in *19th National Conference on AI (AAAI04)*, pp. 167–172, San Jose, CA, (2004).

[11] J. Masthoff, 'Group modeling: Selecting a sequence of television items to suit a group of viewers', *UMUAI*, **14**(1), 37–85, (2004).

[12] J. Masthoff, 'Group recommender systems', *Recommender Systems Handbook*, 677–702, (2011).

[13] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).

[14] K. Schmid, 'Scoping software product lines', in *Software Product Lines – Experience and Research Directions*, pp. 513–532, (2000).

[15] M. Stettinger, 'Choicla: Towards domain-independent decision support for groups of users', in *8th ACM Conference on Recommender Systems*, pp. 425–428, (2014).

[16] M. Stumptner, 'An overview of knowledge-based configuration', *AICOM*, **10**(2), 111–125, (1997).

[17] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.