Liquid Democracy in Group-based Configuration

Muesluem Atas and **Thi Ngoc Trang Tran** and **Ralph Samer**
and **Alexander Felfernig** and **Martin Stettinger**
Institute of software technology, Graz University of Technology, Graz, Austria,
email: {muesluem.atas, ttrang, rsamer, alexander.felfernig, mstettinger}@ist.tugraz.at

Davide Fucci
University of Hamburg, Hamburg, Germany
email: fucci@informatik.uni-hamburg.de

Abstract. Group-based configuration systems support scenarios where a group of users configures a product/service. In those group-based configuration scenarios where the knowledge of some group members regarding items is insufficient, an advice of experts is necessary in order to help members to evaluate products or services. This paper introduces a novel approach which takes advantage of the concept of *liquid democracy* that allows the delegation of group member votes to experts. Concerning the application of *liquid democracy*, we propose a new approach based on *Multi-attribute Utility theory (MAUT)-based evaluation* used to calculate the utility of configurable items. Compared to the traditional approach, the proposed MAUT-based evaluation focuses on the role of experts by assigning higher weights to them. Additionally, the respective expertise level of the experts is taken into account. Consequently, the main contribution of this paper consists in the improvement of group-based configuration by taking *liquid democracy* aspects into consideration.

1 Introduction

Configuration [5, 12] is an important application area of Artificial Intelligence that enables users to configure *complex items* described by many dimensions (attributes). Typical examples of such items include *release plans* [10], *tourism packages* [13], *furnitures* [6], and *financial services* [7, 11]. While most existing configuration systems focus on the support of single users, there also exist scenarios where items can be jointly configured by groups of users, e.g., requirements engineering scenarios where a group of stakeholders configures software release plans. In such scenarios, *group-based configuration systems* have been recognized as being useful tools that help to identify configurations which satisfy preferences of all group members [4]. When interacting with group-based configuration systems, each group member explicitly articulates his/her preference with respect to different item dimensions. Preferences declared by group members are then checked for consistency. As soon as all user preferences are consistent with each other as well as with the knowledge base, the constraint solver will be able to find items that satisfy the preferences of all group members. After this, utility values for each item can be calculated, for example, on the basis of *Multi-attribute Utility theory (MAUT)* [3]. Such an approach takes into account the preferences of group members with respect to the dimensions of items and the importance of dimensions from the users' point of view. The item achieving the highest utility value will then be recommended to the group.

In the context of group-based configuration, sometimes, some group members may be *unable* to evaluate the dimensions of a given set of items due to a knowledge gap. Hence, in order to precisely evaluate items, group members have to invest much effort in order to collect necessary information as well as to analyze items [14]. In such a situation, group members could ask for advice from people who are experts in the item domain of interest. The consultation of experts helps to precisely identify evaluations of items and thereby further facilitates the entire configuration process. The preference configuration of group members in this context can be interpreted and considered as a *liquid democracy* paradigm which provides an alternative decision making model to make better use of collective intelligence [14]. The *liquid democracy* concept empowers group members to either play an *active role* (i.e., *active users* who directly vote items) or a *passive role* (i.e., *passive users* who delegate their rating power to experts) in the voting process [2].

Recently, a variety of studies regarding liquid democracy have been conducted for the purpose of making better use of the so-called "*wisdom of the crowds*" [14]. For instance, Boldi et al. [2] propose a Facebook application that enables each user to select one of his/her friends as the expert of a music genre. The expert then helps him/her to select some pieces of music. Johann et al. [8] introduce the applicability of liquid democracy and e-democracy concepts to address challenges of a massive and continuous user participation in the context of requirements engineering. Zhang et al. [14] propose an efficient *statement voting* scheme that unifies two basic stages of liquid democracy, i.e., *delegation* and *voting*. During the voting/delegating phase, each voter can either vote for candidate(s) or delegate his/her voting power to another voter. Each voter is assigned with a *temporal ID* which is encrypted and distributed in such a way that guarantees the anonymity of the delegation/voting process. Although up to now, there exist many studies with regard to liquid democracy, to some extent it is still unclear how liquid democracy can be applied in the context of group-based configuration. Two emerging questions are: (i) "*How does the system recommend experts to a user who has not enough knowledge about items?*" and (ii) "*How to calculate the utility on the basis of emphasizing the importance of experts who were chosen by stakeholders?*". To the best of our knowledge, there does not exist any research

which provides an in-depth view of the correct application of liquid democracy in group-based configuration. In this paper, we present an insight of the application of liquid democracy in group-based configuration and propose a novel approach of a MAUT-based evaluation that takes preferences of group members/experts into account and thereby assigns a higher importance to the experts.

The remainder of the paper is organized as follows. In *Section 2*, we describe a group-based configuration scenario in requirements engineering which is used as a working example throughout the paper. In *Section 3*, we discuss how liquid democracy can be applied to group-based configuration in order to transfer the rating power from group members to experts. *Section 4* presents a new approach of MAUT-based evaluation to calculate the utility value of a requirement. *Section 5* discusses how requirements can be assigned to releases based on their utility value, their effort estimation, existing dependencies between requirements, and the capacity of releases. Finally, Section 6 draws a brief conclusion and provides some ideas for future work.

2 Working example

For demonstration purposes, we introduce a group configuration scenario occurring in a small requirements engineering example project where we configure a release plan. In this context, we define a set of requirements (R_1, R_2, R_3, and R_4) for developing a *sport watch*. These requirements are defined by a group of engineers with longstanding experience and practical knowledge in requirements engineering. Each requirement is described by an *id*, a *title*, and a textual *description* (see Table 1).

Id	Title	Description
R_1	Evaluation Software	To evaluate the collected training data, an evaluation software is required. The evaluation software requires the connection and the access to the clock's internal memory. The evaluation should contain measured information regarding the distance, the height, the average heart rate, and the calorie consumption.
R_2	Data-Storage Function	To evaluate the measured data, a storage service is required. The internal memory is used for saving the measured information, such as the distance, the height, the average heart rate, and the calorie consumption. The stored data will be used by the evaluation software.
R_3	GPS	To identify the position, a GPS sensor is used. Based on the measured position and time information, the speed and the distance can be measured.
R_4	Display lighting	The sport watch needs a display lighting to be operated at dusk.

Table 1. Example requirements for the development of a *sport watch*. Each requirement is described by an *id*, a *title*, and a textual *description*.

In this example, we assume a situation where a group of five stakeholders (i.e., users) reads requirements, evaluates them regarding different dimensions of requirements, and assigns them to different releases (i.e., release planning configuration). We defined two different releases which are shown in Table 2.

Given the sets of requirements and releases, we assume that each stakeholder evaluated requirements with regard to the following dimensions: *risk, effort*, and *profit*. The dimension of *risk* indicates "the estimated risk for developing a requirement", *effort* represents "the estimated total work done for developing a requirement", and *profit* corresponds to "the estimated profit of a requirement". These

Releases	Capacity (in hours)	Start date	End date
Release 1	260	2020-05-01	2020-07-01
Release 2	260	2020-07-15	2020-09-15

Table 2. Defined releases for the development of a *sport watch*. Each release is described by the *start date*, the *end date,* and the *capacity*. The *capacity* indicates the planned effort of a release in hours.

dimensions are evaluated using ratings. Thereby, the ratings can lie in the range between 1 and 5, where an evaluation of 5 indicates a requirement with *low risk, high profit,* and *low effort* and an evaluation of 1 represents a requirement with *high risk, low profit,* and *high effort*.

The evaluation of stakeholders with regard to different dimensions of requirements is shown in Table 3. In this table, some group members did not sufficiently evaluate dimensions of a requirement (i.e., some dimensions were not evaluated by stakeholders). For instance, the first stakeholder (S_1) did not evaluate the *profit* of the requirement R_2. In addition, there also exist some stakeholders who did not evaluate any dimension of a requirement. For instance, the fourth stakeholder (S_4) did not evaluate any dimension of the requirement R_1. A possible reason for the existence of such missing evaluations can be lacking expertise or knowledge of some stakeholders regarding the meaning of some requirements. In this scenario, those stakeholders who do not know much about the content of some requirements have to invest a lot of effort in order to acquire enough necessary knowledge and to analyze the requirements. This triggers a high cost of the requirement evaluation process. Alternatively, the stakeholder could ask for the advice of some experts to provide more precise evaluations with regard to dimensions of requirements. In other words, the stakeholder directly passes his/her evaluation power to experts by using liquid democracy (see Section 3). In the context of requirements engineering, the expert can be a requirement engineer who has longstanding experiences and practical knowledge of requirements. The consultation of experts helps to precisely evaluate the dimensions of items.

Alternatively, in some cases empty evaluations could be trigged by the reason that the stakeholder does not want to evaluate some dimensions of a requirement. Moreover, he/she also does not want to delegate the rating power to anyone else. In this scenario, group-based configuration systems will automatically check the quantity of complete evaluations of requirements and the configuration phase is only complete if this quantity is high enough. In this example, we assume that the quantity of complete evaluations should be *equal to or greater than 80%* of the total number of all evaluations. In other words, the configuration phase will not be finished until the quantity of available evaluations reaches 80%.

In addition to that, when evaluating a requirement, stakeholders can assign different *weights* to dimensions. Thereby, the weight is referred to as the importance of a dimension, i.e., the higher the importance of a dimension, the higher the weight. Different stakeholders could assign different weights to the same dimension. For instance, one stakeholder (e.g., developer) assumes that the *effort* of a requirement is the most important dimension, whereas another stakeholder (e.g., project manager) evaluates the *profit* to be the most important dimension of a requirement. In order to limit the scope of this paper, some simplifications have to be made. For the sake of simplicity, we assume that the importance (i.e., weight) of all dimensions for all stakeholders is equal and all dimensions have a weight of 1 from the

Stakeholders	Requirement 1 (R_1)			Requirement 2 (R_2)			Requirement 3 (R_3)			Requirement 4 (R_4)		
	Risk	Profit	Effort	Risk	Profit	Effort	Risk	Profit	Effort	Risk	Profit	Effort
S_1	5	3	4	2	-	4	4	4	4	2	-	-
S_2	3	3	-	2	-	3	2	5	-	2	5	4
S_3	3	4	3	5	-	4	2	3	3	4	2	-
S_4	-	-	-	4	-	2	2	4	4	1	3	-
S_5	3	3	4	-	-	4	2	-	4	4	3	4

Table 3. Evaluations of stakeholders with regard to the defined requirements in Table 1. Each requirement is represented by the three following dimensions: *risk, profit,* and *effort*. Each evaluation is in the range of 1 to 5, where the evaluation of 5 indicates a requirement with *low risk, high profit,* and *low effort* and the evaluation of 1 represents a requirement with *high risk, low profit,* and *high effort*. Evaluations which were not provided by stakeholders are represented as a dash symbol ("-").

stakeholders' point of view (i.e., $\forall$ s $\in$ stakeholders, *weight*(*s, risk*) = *weight*(*s, profit*) = *weight*(*s, effort*) = 1).

3 Application of Liquid Democracy

Liquid democracy is a hybrid voting model of participative democracy which combines *direct* and *representative democracy* approaches in order to empower electors [1, 9]. While *direct democracy* allows electors to directly vote for an item, *representative democracy* enables electors to select representatives (or experts) and empower them to vote for items. One of the major issues of direct democracy is the insufficient knowledge of the voter about some items. As a consequence, these voter may provide unprecise evaluations or may even not be able to assess them in a reasonable way. In sharp contrast to direct democracy, representative democracy allows a stakeholder to elect an expert who plays the role of a representative to vote for items. However, representative democracy is also known to show a weakness in terms of *representativeness*. In particular, this is true for scenarios where many voters delegate their voting power to only one expert. That means, the expert's opinion usually represents the idea of many voters and hence, it triggers a situation in which the evaluation of the expert partly reflects the opinion of a stakeholder. In this context, liquid democracy has been recognized as a mixed approach which takes advantage of the strength of direct and representative democracy. Liquid democracy enables voters to either directly vote items or delegate their voting rights to an expert. Consequently, this key benefit of liquid democracy serves as main motivation to apply this voting model.

In this paper, we use a liquid democracy approach in order to complete evaluations of dimensions which were not evaluated by stakeholders. As shown in Table 3, we can observe that stakeholders did not evaluate all dimensions of requirements. In this example, we assume that stakeholders S_2, S_4, and S_5 need experts to complete their evaluations. Expert selection can be done by one of two approaches. The first approach is to select *only one expert* for the three aforementioned stakeholders. The second approach is to allow each group member to select his/her own expert. In our example, we choose the second approach where each stakeholder chooses different experts for different requirements. For instance, regarding the requirement which is related to the user interface, the stakeholder can choose an expert who has experiences in user interface design. For the data storage-related requirement, the stakeholder can choose an expert who has knowledge of data management. In our approach, the expert selection process is done automatically by the recommender system. That means, experts on a specific topic are automatically identified and recommended to the stakeholder. Alternatively, each stakeholder is allowed to select experts who are not included in the recommended list. In our approach, the recommender system suggests experts based on the *expertise level*. In the context of requirements engineering, the expertise level of an expert can be calculated based on the following criteria: *working experience*, *skills*, *number of contributions* in requirements engineering projects, and *number of delegations* received in the requirements engineering domain. The expertise level is in range of 1 to 5. The expertise level of 5 indicates excellent topic-related knowledge, whereas the expertise level of 1 represents limited knowledge.

In this paper, we exemplify an expert recommendation process with 5 experts in the requirements engineering domain. Table 4 shows a recommended list of experts ranked in a descending order of the expertise level. In addition, stakeholders who want to delegate evaluations to other experts can select different experts for different requirements. As shown in Table 1, the development of the defined requirements requires deep knowledge with regard to different areas. Therefore, selecting an appropriate expert for each requirement helps to increase the overall quality of requirements engineering. The expert selection for stakeholders S_2, S_4, and S_5 are depicted in Table 5. In this table, we can observe that stakeholders select different experts for different requirements. For instance, stakeholder S_2 requires experts' evaluations regarding the dimensions of requirements R_1, R_2, and R_3. The stakeholder S_2 chooses $Expert_3$ for R_1, $Expert_2$ for R_2, and $Expert_4$ for R_3. Furthermore, we can observe that the stakeholder S_2 does not need any expert for R_4 and this is represented by a dash symbol ("-") in Table 5.

Experts	Expertise-Level (*sport watch* domain)
$Expert_2$	4.5
$Expert_5$	3.75
$Expert_4$	3.15
$Expert_1$	2.25
$Expert_3$	2.05

Table 4. The expertise level in the *sport watch* domain. The expertise level is in the range of 1 to 5, whereby 1 indicates limited knowledge and 5 indicates excellent knowledge.

Stakeholders	R_1	R_2	R_3	R_4
S_2	$Expert_3$	$Expert_2$	$Expert_4$	-
S_4	$Expert_2$	$Expert_2$	-	$Expert_2$
S_5	-	$Expert_5$	$Expert_2$	-

Table 5. Experts chosen by stakeholders regarding different requirements. The dash symbol "-" represents a situation in which a stakeholder does not need any advice of an expert.

After the selection of experts with respect to each requirement, these experts evaluate the remaining requirement dimensions which

Stakeholders	Requirement 1 (R_1)			Requirement 2 (R_2)			Requirement 3 (R_3)			Requirement 4 (R_4)		
	Risk	Profit	Effort	Risk	Profit	Effort	Risk	Profit	Effort	Risk	Profit	Effort
S_1	5	3	4	2	-	4	4	4	4	2	-	-
S_2	3	3	**3**	2	**3**	3	2	5	**3**	2	5	4
S_3	3	4	3	5	-	4	2	3	3	4	2	-
S_4	**2**	**3**	**3**	4	**4**	2	2	4	4	1	3	**5**
S_5	3	3	4	**3**	**2**	4	2	**4**	4	4	3	4

Table 6. Evaluations of stakeholders with regard to the defined requirements in Table 1. Each requirement is represented by the three following properties: *risk, profit,* and *effort*. The evaluation is in the range of 1 to 5, where the evaluation of 5 indicates a requirement with *low risk, high profit,* and *low effort* and the evaluation of 1 represents a requirement with *high risk, low profit,* and *high effort*. Evaluations which were not provided by stakeholders and experts are represented as a dash symbol ("-"). Evaluations provided by experts are represented in bold text.

were not evaluated by the stakeholders S_2, S_4, and S_5. The evaluations which were given by experts are shown (in bold) in Table 6. Next, the utility of each requirement has to be calculated and used as one of the important criteria to assign requirements to releases. The utility of each requirement is calculated based on *Multi-attribute Utility Theory* (*MAUT*) (see Section 4).

4 Application of Multi Attribute Utility Theory

As already mentioned before, configurable items are usually described by a set of dimensions. In this context, *Multi-attribute Utility Theory* (*MAUT*) [3] is applied. In this paper, we propose a new MAUT-based approach that calculates the utility of an item *i* according to the evaluations of stakeholders (*evaluation(s,d)*) with regard to dimensions d, the importance of these dimensions (*w(s,d)*) from the stakeholders' point of view, and the importance of stakeholders/experts (*w(s)*). The final result of the MAUT evaluation is then represented by the *weighted average* of all stakeholders' evaluations for the dimensions d. Formula 1 indicates that an experts' evaluation *evaluation(e,d)* for a dimension d is used in cases where a stakeholders' voting is delegated. Otherwise, a stakeholders' own evaluation will be taken into account for the MAUT calculation. In our approach, compared to a stakeholder, an expert has a higher impact on the overall utility of an item, i.e., the weight of an expert is twice the weight of a stakeholder (see Formula 2). In addition, the expertise level *el(e)* of an expert *e* is also considered in the weight calculation. The total MAUT value (i.e., the utility value) of a requirement R_i is then calculated by summing all dimension-specific MAUT values of the requirement R_i (see Formula 3).

$$eval(s,d) = \begin{cases} evaluation(e,d) & \text{if} \textit{evaluation(s,d) delegated} \\ evaluation(s,d) & \textit{otherwise} \end{cases} \tag{1}$$

$$w(s) = \begin{cases} weight(s) * 2 + el(e) & \text{if } \textit{evaluation(s,d) delegated} \\ weight(s) & \textit{otherwise} \end{cases} \tag{2}$$

$$Utility(R_i) = \frac{\sum_{s \in stakeholders} \frac{\sum_{d \in dims} eval(s,d) * w(s,d) * w(s)}{\sum_{d \in dims} w(s,d) * w(s)}}{|stakeholders|} \tag{3}$$

An example of the utility calculation of a requirement is presented in Formula (4). In this example, for simplicity, we assume that all stakeholders assign the same weight (i.e., the weight of 1) for all dimensions of requirements ($\forall s \in stakeholders, \forall d \in dimensions$, w$(s,d)$= 1). Additionally, we assume that each stakeholder has also the same importance ($\forall s \in stakeholders$ weight(s)= 1).

$$\begin{aligned} Utility(R_2) &= \frac{\sum_{s \in stakeholders} \frac{\sum_{d \in dims} eval(s,d) * w(s,d) * w(s)}{\sum_{d \in dims} w(s,d) * w(s)}}{|stakeholders|} \\ &= \frac{1}{5}\Big(\frac{2*1+4*1}{1+1} + \frac{2*1+3*(1*2+4.5)+3*1}{1+(1*2+4.5)+1} \\ &+ \frac{5*1+4*1}{1+1} + \frac{4*1+4*(1*2+4.5)+2*1}{1+(1*2+4.5)+1} \\ &+ \frac{3*(1*2+3.75)+2*(1*2+3.75)+4*1}{(1*2+3.75)+(1*2+3.75)+1}\Big) \\ &= \frac{1}{5}\Big(\frac{6}{2} + \frac{24.5}{8.5} + \frac{9}{2} + \frac{32}{8.5} + \frac{32.75}{12.5}\Big) = 3.354 \end{aligned} \tag{4}$$

Similarly, MAUT values of other requirements are also calculated by using Formulae 1 - 3. The MAUT values of requirements R_1, R_2, R_3, and R_4 are the following: $MAUT(R_1) = 3.266$, $MAUT(R_2) = 3.354$, $MAUT(R_3) = 3.380$ and $MAUT(R_4) = 3.326$. After the calculation of requirement utilities, requirements will be assigned to defined releases (see Section 5).

5 Release Planning

In Section 4, we showed how the utility value of a requirement can be calculated based on *Multi-attribute Utility Theory* (*MAUT*). The higher the MAUT value, the sooner the requirement will be implemented. In the context of requirements engineering, making a recommendation of requirements is referred to as release planning, i.e., to clarify which requirement should be implemented in which release. In release planning, stakeholders have to estimate the effort investing for each requirement. In our working example, the *effort* is referred to as the invested time (in hour) to implement a requirement. The higher the evaluation of effort, the lower the invested time. We assume that an evaluation of 5 corresponds to an effort of 50 hours and an evaluation of 1 corresponds to an effort of 250 hours. In order to calculate the effort of a requirement, we first calculate the average of evaluations with regard to the effort of the requirement given by all stakeholders. After that, the effort (in hour) is calculated using the Formula 5, where $effort(R_i, s)$ is the evaluation of the stakeholder s about the effort of the requirement R_i.

$$effort(R_i) = \Big(5 - \frac{\sum_{s \in stakeholders} effort(R_i, s)}{|stakeholders|} + 1\Big) * 50 \tag{5}$$

We exemplify the calculation of the effort of the requirement R_1 as shown in Formula 6. The effort values of other requirements are calculated in a similar way and presented in Table 7.

$$effort(R_1) = \left(5 - \frac{4+3+3+3+4}{5} + 1\right) * 50 = 130 \quad (6)$$

Requirements	Average effort (effort in hours)	Assigned release
R_1	3.4 (130)	Release 2
R_2	3.4 (130)	Release 1
R_3	3.6 (120)	Release 1
R_4	4.33 (83.33)	Release 2

Table 7. The assignment of requirements to releases based on the effort of requirements, dependencies between requirements, their utility values, and capacity of releases. The effort of each requirement in represented in the second column of the table.

In our example, release planning is done based on four criteria: the *utility* (MAUT) value, *effort* (measured in hours), the *dependency* between requirements, and the *capacity* of releases. Given the fact that requirement R_3 achieves the highest utility (i.e., $MAUT(R_3) = 3.380$) and its estimated time effort of 120 hours, R_3 turns out to be the best candidate to be assigned to *Release 1*. Furthermore, it is reasonable that requirement R_2 should follow R_3 and hence also be assigned to *Release 1*, as R_2 shows the second highest utility and there is still some remaining capacity left for *Release 1* to cover R_2 in this release (i.e., capacity(*Release 1*) = 260 hours). Next, requirement R_4 has to be assigned to a release. The assignment of requirement R_4 to the first release is not possible, because of the limited capacity (remaining capacity (*Release 1*) = 10 hours and effort(R_4) = 83.33 hours). Therefore, requirement R_4 is assigned to the second release. Finally, requirement R_1 is assigned to the second release. Based on the description of requirements shown in Table 1, we can observe that there is a dependency between R_1 and R_2 which is indicated as follows: "*The evaluation software requires the access to the clock's internal memory*". In the context of requirements engineering, this means that the first requirement R_1 (i.e, evaluation software) can not be implemented before R_2 (i.e., data storage function) has been completed. However, in this scenario, the identified dependency will not trigger any changes in the release planning since the release plan in Table 7 shows that requirement R_2 which is assigned to the first release (development period: from 2020-05-01 to 2020-07-01) will be implemented before all the requirements assigned to the second release (development period: from 2020-07-15 to 2020-09-15). With this final step, all requirements are assigned to releases and the requirements engineering process is complete.

6 Conclusion and Future Work

In this paper, we introduced utility analysis concepts which focus on *liquid democracy*. These concepts allow the manual delegation of a stakeholder's voting right to a domain expert. First, we described a scenario for the development of a sport watch which is used as a working example throughout this paper. Based on the working example, we applied liquid democracy in order to receive consultations from experts in situations where stakeholders do not have enough knowledge with regard to certain requirements. Afterwards, we proposed a novel approach of a MAUT-based evaluation which takes into account users' and experts' evaluations and assigns higher importance to expert consultations (i.e., evaluations). Finally, we proposed a group-based configuration for release planning where requirements were assigned to releases based on derived utility values, effort estimations, existing dependencies, and release capacities.

Within the scope of future work, we plan to integrate the proposed approach in a requirements engineering tool named INNOSENSR[1]. INNOSENSR is a modern innovative release planning tool which makes use of intelligent techniques in order to facilitate the complete requirements engineering process. In the current version of INNOSENSR, stakeholders have to evaluate requirements without getting any support from domain experts. In the future, we will integrate our approach into INNOSENSR in order to increase the requirements engineering quality.

Acknowledgment

The work presented in this paper has been conducted within the scope of the Horizon 2020 project OPENREQ (732463).

REFERENCES

[1] Christian Blum and Christina Isabel Zuber, 'Liquid democracy : Potentials, problems, and perspectives', *The Journal of Political Philosophy*, **24**, 162–182, (2016).

[2] Paolo Boldi, Corrado Monti, Massimo Santini, and Sebastiano Vigna, 'Liquid FM: recommending music through viscous democracy', in *Proceedings of the 6th Italian Information Retrieval Workshop, Cagliari, Italy, May 25-26, 2015.*, (2015).

[3] J. S. Dyer, *Maut - Multi-attribute Utility Theory*, 265–292, Springer New York, New York, NY, 2005.

[4] Alexander Felfernig, M Atas, TNT Tran, and Martin Stettinger, 'Towards group-based configuration', in *International Workshop on Configuration 2016 (ConfWS16)*, pp. 69–72, (2016).

[5] Alexander Felfernig, Lothar Hotz, Claire Bagley, and Juha Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1 edn., 2014.

[6] A. Haag, 'Sales configuration in business processes', *IEEE Intelligent Systems and their Applications*, **13**(4), 78–85, (Jul 1998).

[7] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich, *Recommender Systems: An Introduction*, Cambridge University Press, New York, NY, USA, 1st edn., 2010.

[8] Timo Johann and Walid Maalej, 'Democratic mass participation of users in requirements engineering?', in *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, pp. 256–261, (2015).

[9] Anna Litvinenko, 'Social media and perspectives of liquid democracy on the example of political communication of pirate party in germany', in *The Proceedings of the 12th European Conference on e-Government in Barcelona*, pp. 403–408, (2012).

[10] Gerald Ninaus, Alexander Felfernig, Martin Stettinger, Stefan Reiterer, Gerhard Leitner, Leopold Weninger, and Walter Schanil, 'Intellireq: Intelligent techniques for software requirements engineering', in *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, pp. 1161–1166, Amsterdam, The Netherlands, The Netherlands, (2014). IOS Press.

[11] Markus Stolze, Simon Field, and Pascal Kleijer, 'Combining configuration and evaluation mechanisms to support the selection of modular insurance products', in *Proceedings of the 8th European Conference on Information Systems, Trends in Information and Communication Systems for the 21st Century, ECIS 2000, Vienna, Austria, July 3-5, 2000*, pp. 858–865, (2000).

[12] Markus Stumptner, 'An overview of knowledge-based configuration', *Artificial Intelligence Community*, **10**(2), 111–125, (April 1997).

[13] TNT Tran, Müslüm Atas, Martin Stettinger, and Alexander Felfernig, 'An extension of choicla user interfaces for configurable products', *RSBDA'16*.

[14] Bingsheng Zhang and Hong-sheng Zhou, 'Brief announcement: Statement voting and liquid democracy', in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pp. 359–361, New York, NY, USA, (2017). ACM.

[1] `http://innosensr.com`