

Dissertation

**Recommendation Technologies for
Group Decision Making**

Dipl.-Ing. Martin Stettinger, BSc

Graz, March 2016

*Institute for Software Technology
Graz University of Technology*



Supervisor/First reviewer: Univ.-Prof. Dipl.-Ing. Dr. techn. Alexander Felfernig

Second reviewer: Prof. Dr. Walid Maalej

Abstract (English)

Many everyday life decisions are made in the context of groups. Such decisions range from crucial investment decisions, for example, in the public sector, to the selection of the next restaurant for a dinner with friends. The field of group decision support by means of recommenders represents a new and upcoming area of research. Besides known challenges such as the cold start problem, group recommender systems have to deal with several additional risk factors which can negatively affect the decision outcome. A major type of risk which can significantly decrease the quality of decision outcomes in group decision scenarios are decision biases. *Decision biases* describe situations where humans tend to decide in certain (simplified) ways.

Examples of decision biases are *anchoring effects* and *serial position (primacy/recency) effects*. Anchoring effects occur in situations where the final group decision is influenced by the first person in the group who articulates his/her preferences. Primacy/recency effects explain situations where individual preferences are affected by the sequence in which information units (e.g., description statements of solution alternatives) are presented.

In this thesis we present CHOICLA, a novel group decision support environment which shows the ability to counteract decision biases and thus has the potential to significantly increase the quality of decision outcomes. CHOICLA advances the state-of-the-art by providing decision support for groups of users in a domain-independent fashion. Existing technologies only provide support for specific domains (e.g., coordinating appointments, selecting tourist destinations, music, ...) and are not open in the sense that no functions for a domain-independent configuration of new decision tasks are provided.

Additionally, we present the results of several empirical studies that have been conducted with the aim of getting feedback on the usability of CHOICLA as well as the acceptance and prediction quality of our introduced group recommendation heuristics. Furthermore, the studies provide evidence that the concepts introduced in CHOICLA have the potential to counteract decision biases such as anchoring effects and serial position effects.

Finally, we present first insights on how specific recommendation heuristics can influence the communication behaviour among group members during a decision process. Due to the fact that a higher amount of information interchange can significantly increase the quality of a decision, we discuss recommendation heuristics which have the potential to increase communication frequency.

Abstract (German)

Eine Vielzahl an Alltagsentscheidungen betreffen eher Gruppen als Einzelpersonen. Beispielhafte Gruppenentscheidungen reichen von heiklen Investitionsentscheidungen (beispielsweise in der öffentlichen Verwaltung) bis hin zu Restaurant-Entscheidungen für ein gemeinsames Abendessen mit Freunden. Empfehlungstechnologien zur Unterstützung von Gruppenentscheidungen stellen einen neuen aufstrebenden Forschungsbereich dar. Zusätzlichen Risikofaktoren, welche die Entscheidungen negativ beeinflussen können, müssen neben den bekannten Herausforderungen (z.B.: "cold start problem") berücksichtigt werden. Sogenannte "decision biases" stellen dabei häufig auftretende Beeinflussungen dar. *Decision biases* beschreiben Situationen, in welchen Menschen nach bestimmten (vereinfachten) Mustern handeln.

Anchoring Effekte und Serial Position (Primacy/Recency) Effekte repräsentieren häufig auftretende decision biases. Anchoring Effekte triggern Entscheidungen, die von der ersten Person, welche innerhalb der Gruppe Präferenzen artikuliert, beeinflusst werden. Primacy/Recency Effekte beschreiben Situationen, in welchen die Reihenfolge der präsentierten Informationseinheiten (z.B.: Beschreibungstext einer Alternative) einen wesentlichen Einfluss auf die persönlichen Präferenzen ausübt.

In dieser Arbeit wird eine neue Umgebung zur Unterstützung von Gruppenentscheidungen präsentiert. Das vorgestellte CHOICLA System zeigt die Fähigkeit, decision biases entgegenzuwirken und somit die Entscheidungsqualität signifikant zu verbessern. CHOICLA erweitert den state-of-the-art, indem es Unterstützung für Gruppenentscheidungen unabhängig von der zugrunde liegenden Domäne anbietet. Existierende Technologien bieten lediglich Unterstützung für spezielle Domänen (z.B.: Terminkoordinations, Tourismusdestinationen, Musik, ...) und bieten keine Möglichkeit, Entscheidungsaufgaben domänenunabhängig zu konfigurieren.

Zusätzlich präsentiert diese Arbeit Resultate zahlreicher Benutzerstudien, welche Feedback sowohl zur Benutzerfreundlichkeit des Systems als auch zur Akzeptanz und Empfehlungsqualität der entwickelten Empfehlungsfunktionen liefern. Des Weiteren liefern die Ergebnisse einen Beweis dafür, dass die in CHOICLA integrierten Technologien das Potential haben, decision biases wie Anchoring Effekte und Primacy/Recency Effekte entgegenzuwirken.

Abschließend werden Ergebnisse bezüglich der Auswirkung von speziellen Empfehlungsfunktionen auf das Kommunikationsverhalten innerhalb einer Gruppe während eines Entscheidungsprozesses

diskutiert. Da die Entscheidungsqualität durch ein hohes Kommunikationsaufkommen signifikant gesteigert werden kann, werden Empfehlungsfunktionen erörtert, welche den Informationsaustausch innerhalb der Gruppe signifikant erhöhen können.

Acknowledgement

At this point I would like to thank all the people who have supported me throughout my entire studies.

Special thanks go to the following people:

- Mr. Univ.-Prof. Dipl.-Ing. Dr.techn. Alexander Felfernig who offered me support in all respects at any time and thus allowed me a frictionless and pleasant way of completing this PhD thesis.
- My family which supported me in all respects throughout my whole studies.
- My colleagues Dipl.-Ing. Stefan Reiterer, Michael Jeran, Dipl.-Ing. Gerald Ninaus, Dipl.-Ing. Florian Reinfrank, Dipl.-Ing. Klaus Isak, and Assoc.Prof. Mag. Dr. Gerhard Leitner.

Martin Stettinger
Graz, 2016

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Graz,

Place, Date

Signature

EIDESSTÄTTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Graz, am

Ort, Datum

Unterschrift

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Research Objectives	4
1.3. Contributions	7
1.4. Thesis Outline	11
2. Basic Approaches in Recommender Systems	15
2.1. Abstract	15
2.2. Introduction	15
2.3. Collaborative Filtering	16
2.4. Content-based Filtering	20
2.5. Knowledge-based Recommendation	22
2.6. Hybrid Recommendations	27
2.7. Hints for Practitioners	29
2.7.1. Usage of Algorithms	29
2.7.2. Recommendation Environments	30
2.8. Further Algorithmic Approaches	31
2.8.1. Critiquing-based recommendation	31
2.8.2. Group recommendation	32
2.9. Conclusions	33
3. Human Computation Based Acquisition Of Financial Service Advisory Practices	35
3.1. Abstract	35
3.2. Introduction	35
3.3. Developing PEOPLEVIEWS Recommenders	37
3.4. User Interface	41
3.4.1. PEOPLEVIEWS	41

3.4.2. STUDYBATTLE	43
3.5. Preliminary Evaluation Results	47
3.6. Future Work	48
3.7. Conclusions	50
4. Towards Open Configuration	53
4.1. Abstract	53
4.2. Introduction	53
4.3. Community-based Knowledge Engineering	55
4.4. Group-based Configuration	58
4.5. Flexible Product Enhancement	61
4.6. Related and Future Work	62
4.7. Conclusions	63
5. Configuring Decision Tasks	65
5.1. Abstract	65
5.2. Introduction	65
5.3. Configuring a decision task	66
5.4. Dependencies among features	71
5.5. Configuring decision tasks in CHOICLA	72
5.6. Related and Future Work	73
5.7. Conclusions	74
6. Choicla: Intelligent Decision Support for Groups of Users in the Context of Personnel Decisions	75
6.1. Abstract	75
6.2. Introduction	75
6.3. Choicla Decision Support	76
6.3.1. Design of Decision Apps	76
6.3.2. Choicla Decision Apps	80
6.4. Choicla Personnel Decisions	80
6.4.1. Users View	80
6.4.2. Candidates View	82
6.5. User Study	83
6.6. Related & Future Work	83
6.7. Conclusions	85
7. Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment	87
7.1. Abstract	87

7.2. Introduction	87
7.3. Choicla Environment	90
7.3.1. Configurability of Decision Apps	90
7.3.2. Recommendation Support	91
7.3.3. Decision Alternatives	94
7.3.4. Evaluation of Decision Alternatives	94
7.3.5. Explanations	95
7.3.6. Preference Visibility	96
7.4. Choicla Decision Apps	96
7.5. User Study	97
7.6. Conclusions and Future Work	101
8. Counteracting Anchoring Effects in Group Decision Making	103
8.1. Abstract	103
8.2. Introduction	103
8.3. The CHOICLA Environment	106
8.4. User Study	107
8.5. Conclusions and Future Work	113
9. Conclusions & Future Work	115
9.1. Conclusions	115
9.2. Future Work	118
List of Figures	123
List of Tables	127
Bibliography	131

Introduction

1.1. Motivation

Recommendation technologies for the support of single user decisions are well studied but the *support of group decision tasks by means of recommenders* is a new and upcoming field of research (see, e.g., Masthoff (2011)). There are a few applications with basic group recommendation approaches included but these are restricted to very specific domains such as, software requirements engineering (Felfernig et al. (2012b)), ambient intelligence (Perez et al. (2010)), interactive television (Masthoff (2004)), and e-tourism (Jameson (2004); McCarthy et al. (2006)).

Due to the fact that group decision tasks differ in terms of their process design, a variety of configurable features is needed to support the design of decision tasks (Stettinger et al. (2014)). Such features can be, for example, specific heuristics that support the recommendation of decisions, restriction of participants who are allowed to introduce additional decision alternatives in the decision process, or special preference visibilities during the decision process. The underlying domain of a group decision task has a major impact on the needed feature combination. Also the assessment criteria used by recommender systems depend on the underlying domain. For example, personnel decisions need assessment criteria different from decisions regarding a restaurant location or a cinema movie.

Recommendation technologies can be separated into collaborative filtering, content-based filtering, knowledge-based recommendation, and group recommendation (Jannach et al. (2010)). *Collaborative filtering (CF)* (Konstan et al. (1997)) calculates recommendations for single users on the basis of other users with similar tastes. The first step is to define a measure for the similarity of users inside the system. On the basis of similar users, behavioural patterns are used to recommend items of interest to the current user. Recommendations based on *Content-based Filtering* (Pazzani and Billsus (2007)) present new items to the user which are similar to the ones the user has already rated. Also for content-based filtering a measure of similarity is needed to find items which are similar to each other. Both

collaborative filtering and content-based filtering show a cold start problem where the system can not generate a "useful" recommendation if new users or new items are added. In scenarios where collaborative filtering and content-based filtering can not be applied, knowledge-based recommendation technologies can be used. *Knowledge-based recommendation* (Burke (2000); Felfernig and Burke (2008)) is based on explicit knowledge about products (items) and the user preferences as well as knowledge about the context-dependent item recommendation. The need to define the recommendation knowledge in an explicit fashion represents the drawback of knowledge-based recommendation and triggers a *knowledge acquisition bottleneck* due to communication overloads between domain experts and knowledge engineers. The major advantage of knowledge-based recommendation is that due to the explicit recommendation knowledge no cold start problems occur (Jannach et al. (2010)). Compared to recommendation scenarios for single users, *group recommendation* scenarios have to cope with several new challenges due to the fact that the system has to generate recommendations that take the individual preferences of all group members into account. Thus the recommended item(s) for a group of users represent those which best fit the actual group preference.

Many everyday life decisions affect groups of users, for instance, decisions regarding the location of the next skiing trip with friends or a decision regarding a new company name or logo. Some decision tasks such as selecting a restaurant for a dinner with colleagues occur regularly in the same group. Very often biases occur in the context of group decision scenarios which can lead to suboptimal decision outcomes. Cosley et al. (2003) and Adomavicius et al. (2013) describe decision biases as undesirable for recommendation systems for different reasons. One major disadvantage can be, for instance, that biases provide opportunities for manipulation of the recommender system. Furthermore, biases significantly reduce the quality of recommendations because they deteriorate the input quality of the recommender system (Cosley et al. (2003); Adomavicius et al. (2013)). Another negative consequence of decision biases is that the user's view of items relevance as well as the recommender's view of user preferences can be contorted (Cosley et al. (2003); Adomavicius et al. (2013)). For a more detailed overview of decision biases we refer to Felfernig (2014).

Anchoring effects, for example, are responsible for decisions which are biased by the first preference articulating person in a group (Adomavicius et al. (2011); Jacowitz and Kahneman (1995)). If a recommender system does not disclose the preferences of other group members in early stages of a decision process, the overall information exchange between the group members can be increased. Due to this increase of information/knowledge exchange, the quality of the decision outcome can be improved (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). The occurrence of anchoring effects in recommendation contexts is confirmed by several authors. Felfernig et al. (2012b) analyzed bias-induced preference shifts in the context of requirements engineering and confirmed the existence of anchoring effects. Similar to group decision scenarios also in collaborative filtering scenarios anchoring effects are triggered by unfolding the ratings of similar users (Adomavi-

cius et al. (2011, 2014)). Results regarding relationships between decision quality and degree of preference disclosure are confirmed by social-psychological studies (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)).

Primacy/recency effects (serial position effects) describe situations where items at the beginning and the end of an item list are recalled and evaluated more often than those in the middle of the list (Felfernig et al. (2007a); Murphy et al. (2012)). In this context the increased evaluation probability represents the behavioural aspect whereas the recall itself describes the cognitive aspect (Felfernig et al. (2007a); Murphy et al. (2012)). The mentioned items can be, for instance, lists of links (Murphy et al. (2012)), a menu plan in a restaurant (Bar-Hillel (2015)), products and their attribute descriptions (Felfernig et al. (2007a)) as well as argumentations in product descriptions (Stettinger et al. (2015b)). The popularity of an item or attribute has no influence on the occurrence of this effect – for example, item properties presented at the beginning and the end of recommendation dialogs are recalled significantly more often independent of their relevance for the current user (Felfernig et al. (2007a)).

Serial position effects are among the most robust effects in psychology (Bar-Hillel (2015)). They are known for decades, backed by hundreds of studies, and can be explained by the fact that humans dislike evaluating large lists of items to identify those that fit their preferences best (Bar-Hillel (2015)). One possible approach to counteract such effects in the context of group decision scenarios is the adaptation of the preference acquisition interface to motivate the participants to analyze the item descriptions in more detail. To achieve this, the preference acquisition interface can be changed, for example, from a star-based rating scale to a utility-based rating scale where items are evaluated on the basis of a defined set of interest dimensions (Stettinger et al. (2015b)).

Often the decision outcome is not explained and in some cases even not all group members are informed about the final decision. Explanations can be related to the group decision itself as well as to individual items. Explanations for group decisions play an important role since they can have a positive impact on the overall acceptance of group decisions (Stettinger et al. (2015a)). Item explanations (e.g., argumentations as to why an item is recommended to a group) can have a significant influence on the decision outcome since they have an impact on the way items are perceived and also evaluated (Tintarev and Masthoff (2007); Gkika and Lekakos (2014)). Finally, Stettinger et al. (2015b) showed that even the ordering of the arguments used in an item explanation can have a significant impact on the decision outcome. The lack of explanations in recommender systems can lead to a lower level of trust in recommender systems (Felfernig et al. (2006); Pu and Chen (2007)).

The work presented in this thesis focuses on group recommender systems and also presents a decision support system called CHOICLA¹ where the gained knowledge of all user studies is integrated in one tool. CHOICLA is a self-explanatory group decision support system with a special focus on

¹<http://www.choicla.com/>

the system's usability. CHOICLA also takes into account the above mentioned risk factors and moves group decision making one step further by providing:

1. Explanations in all phases of a group decision task (e.g., argumentations as to why an item is recommended to a group). Explanations ensure that the decisions are transparent and the participants are more likely to trust the system. Explanations as well as transparency have a positive influence on the users trust in the system (Felfernig et al. (2006); Pu and Chen (2007)).
2. Configuration technology. Due to the configuration functionalities provided during the creation process of a group decision task, the CHOICLA environment is open in the sense that one can create group decision tasks independently from the underlying domain (Stettinger et al. (2014)).
3. Domain-dependent recommendation heuristics. Group recommendations in CHOICLA are calculated on the basis of the individual preferences of the group members. Due to the fact that there is no group recommendation heuristic that fits each and every group decision task, several different group recommendation heuristics are available in the system (Masthoff (2011)).
4. Advanced level of usability. One major focus during the development of CHOICLA was to assure usability. The system is now at a stage where it is easy understandable and easy to use by people without information technology background. To achieve the largest degree of usability, the user interfaces are dynamic and show only items and actions which should be accessible to the user in the current state of the group decision process (Stettinger et al. (2014)).
5. Variable preference elicitation schemes. To achieve the highest level of flexibility of the system, the creator of a group decision task can choose among different preference elicitation schemes. These schemes reach from a simple five-star evaluation of the decision alternatives to utility-based preference elicitation where the participants can articulate their preferences on basis of interest dimensions which can be weighted. Utility-based preference elicitation interfaces show the ability to counteract serial position effects (Felfernig et al. (2007a); Murphy et al. (2012)).
6. Intelligent preference visibility options. CHOICLA offers different preference transparency settings. If preferences should be transparent during the decision task, users can – depending on the decision task – select among different representation schemes which reach from a summary of the individual preferences to full preference transparency. Intelligent preference visibility settings help counteracting anchoring effects (Adomavicius et al. (2011); Jacowitz and Kahneman (1995); Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)).

1.2. Research Objectives

On the basis of the previous discussion we identified the following research issues:

1. Support groups of users in decision scenarios

Several everyday decisions rather arise in the context of groups than single individuals. For

this reason we have to think further and advance the state of the art in recommender systems by offering recommendation technologies for groups of users. *Recommendation technologies in the context of group decision scenarios* is a new and upcoming field of research (Masthoff (2011)). Existing technologies focus on specific domains such as interactive television (Masthoff (2004)), ambient intelligence (Perez et al. (2010)), news access or e-tourism (Jameson (2004); McCarthy et al. (2006)) but are not able to cover different kinds of decision scenarios. Group decisions can be negatively influenced by various risk factors. For example, knowledge about the preferences of other group members in early stages of the decision process can have a negative impact on decision quality since such decisions are in most cases biased by the first preference-articulating person(s) (Anchoring effect - see Jameson (2004); Nunamaker et al. (1991); Adomavicius et al. (2011); Jacowitz and Kahneman (1995); Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). The non-inclusion of explanations during and after the decision process can lead to suboptimal decision outcomes since missing explanations can significantly decrease the user's trust in the system (Felfernig et al. (2006)). Also serial position effects can cause low quality decision outcomes (Jacowitz and Kahneman (1995); Bar-Hillel (2015)). This brings us to the first research question:

(Q1) *How to design a domain-independent decision support environment for groups of users?*

2. Configuration of decision tasks

In most cases where configuration technologies are applied, one (or a small group of) knowledge engineer(s) is in charge of knowledge base development and maintenance (Hoppenbrouwers et al. (2009); Wagner (2006); Hayes-Roth et al. (1983)). In such scenarios it is also assumed that only single users configure all the corresponding products and services. This assumption leads to scalability problems due to the fact that the transformation of domain knowledge into a configuration knowledge base as well as preference recording for a whole group are time-intensive tasks (Hayes-Roth et al. (1983); Richardson and Domingos (2003)). Modern configuration technologies should offer a possibility to cooperatively develop knowledge bases by groups of users as well as to jointly configure products and services (Felfernig et al. (2014d)). In such cases the construction of the knowledge base represents the decision task where a group of knowledge engineers decides which constraints best describe the product knowledge. Since decision tasks are unique and differ in terms of their corresponding process design, they require different combinations of features. Features in such contexts are, for instance, special preference visibilities during the decision process, restriction of participants who are allowed to enter additional decision alternatives or specific heuristics that support the recommendation of decisions. In order to ensure the consistency of selected feature combinations for the creator of a group decision task (in most cases a single user) it is essential to offer a corresponding configuration functionality. This brings us to our second research question:

(Q2) *How to design the creation process of a group decision task as a configuration problem?*

3. Personnel decisions

Literature shows that factors, such as interpersonal attraction, first impressions or the appearance of a job applicant are very often responsible that no concrete structure is followed during a job interview and this could lead to a subjective evaluation of a job candidate (Kobrynowicz and Biernat (1997); Dougherty et al. (1994); Graves and Powell (1988)). In such cases often no "fair" and objective decision is taken due to the fact that the assessment criteria change (Kobrynowicz and Biernat (1997); Dougherty et al. (1994); Graves and Powell (1988)). Since hiring procedures often concern more than one single person we are faced with a group decision problem. This results in the third research question:

(Q3) *How to best support groups of users in the context of personnel decisions?*

4. Fairness for all group members

An essential function of a group recommendation tool is the capability to collect and reuse information from past decision scenarios. This information can be used in future decision scenarios to achieve the best recommendation accuracy. One factor where information of past decision scenarios can be exploited in future decision contexts is fairness among the group members in the long run (Ariely and Zakay (2001)). Fairness is an important issue in group decision contexts – the degree of perceived fairness influences the willingness of group members to accept compromises in the resolution of conflicting preferences and also their degree of trust in other group members (Lind et al. (2001)). The research question is:

(Q4) *How to achieve long term fairness in group decision tasks?*

5. Decision biases in the context of group decision scenarios

Similar to single user decisions, decision biases can also affect group decisions. Without a technical support, humans often apply simple decision heuristics which cause different types of decision biases that lead to suboptimal decision outcomes. An example of a simple decision heuristic is *elimination by aspects* (Bettman et al. (1998)) where people eliminate alternatives which do not reach a minimum cut-off value for the most important attribute. This elimination process is repeated until a single option remains (Bettman et al. (1998)). A common form of displaying a larger number of items is a list representation. Humans evaluate items at the beginning and the end of a list significantly more often than those items which are placed in the middle of a list (Murphy et al. (2012)). This decision bias is called primacy/recency effect (serial position effect). Serial position effects can be explained by the fact that users are not interested in analyzing large item lists (Bar-Hillel (2015)). Another explanation for primacy/recency effects is the perspective as cognitive phenomenon based on the fact that items at the beginning and the end of a list are recalled more often (Felfernig et al. (2007a)). In the context of recommender systems for single users these effects have been studied, for instance,

by Felfernig et al. (2007a). The investigations clearly point out that item attributes which are shown to a user at the beginning and the end of a recommendation dialog are recalled significantly more often than those shown in the middle of a dialog.

Another specific type of decision biases are anchoring effects. Anchoring effects in group decision contexts cause decisions which are biased by the first preference-articulating person in the group (Adomavicius et al. (2011); Jacowitz and Kahneman (1995); Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). The optimal time to disclose the preference information of the other users is a key factor to counteract anchoring effects in group decision contexts (Stettinger et al. (2015a)). This leads us to the following two research questions:

(Q5.1) *How to counteract serial position effects in group decision support environments?*

(Q5.2) *How to counteract anchoring effects in group decision support environments?*

1.3. Contributions

Highly flexible support of group decisions. One major contribution of this thesis is to present CHOICLA which is a novel group recommendation support environment. CHOICLA advances the state-of-the-art because it is able to handle group decision tasks independently from their underlying domain. Due to the heterogeneity of decision tasks a key feature of such a software tool is a *high degree of flexibility*. This flexibility is introduced by defining the design of a group decision task (the underlying problem) as configuration problem (Stettinger et al. (2014)). Based on this technique we are able to build a model that is flexible with regard to the generation of problem-specific decision applications.

Structured support of personnel decisions. Another focus of this thesis is to introduce an approach to support groups of users in the context of personnel decisions. Personnel decisions are often not structured and the evaluation of the job candidates is for most parts subjective due to the fact that no concrete structure is followed and also evaluation criteria are not stable over the whole hiring procedure (Kobryniewicz and Biernat (1997); Dougherty et al. (1994); Graves and Powell (1988)). Switching from a star-based preference acquisition interface to a utility-based one, CHOICLA can offer a structured evaluation of the candidates. In such contexts the job-candidates are evaluated along dimensions which can be defined during the modeling phase of a decision task. Therefore the dimensions (evaluation criteria) are precisely tailored towards the current job-position because different job-positions require in many cases also different assessment criteria. The developed recommendation approach is based on a modified version of group-based MAUT (Stettinger and Felfernig (2014)) where the attribute values are subjective and the weights are fixed which is different compared to a typical group-based MAUT scenario (Dyer (2005)).

Fairness in the long run. Some group decision tasks such as selecting the appropriate restaurant

location for a Christmas party or a monthly dinner with colleagues come up regularly. In recurring decision tasks we can learn from past decisions and thus provide fair recommendations in future scenarios. Such recommendations can achieve a higher degree of fairness in the long run.

Measures to counteract decision biases. Furthermore we present techniques which help to counteract decision biases. If a system presents decision alternatives by means of a list, users are biased by the first and last items of the list. This can be explained by the fact that such edge items (first and last few items of a list) are recalled and evaluated significantly more often than those in the middle of a list (Felfernig et al. (2007a); Murphy et al. (2012)). We show how to counteract this kind of decision bias (primacy/recency effects) by changing the preference acquisition interface to a utility-based one. Providing insights to the preferences of the other participants before the individual evaluation is finished very often leads to anchoring effects. We found out that anchoring effects also exist in context of group decision scenarios and provide techniques (based on the results of our user studies) that can counteract such effects.

High degree of usability. CHOICLA represents one integrated tool where all the gained knowledge from our empirical studies is integrated. During the development of the CHOICLA technologies we spend a huge effort on making the system clear and understandable by end-users and therefore also evaluated the systems usability multiple times.

A summary of the research questions and the related contributions can be found in Table 1.1.

Research Questions	Contributions
(Q1) <i>How to design a domain-independent decision support environment for groups of users?</i>	In this contribution we give insights to CHOICLA which represents a novel group decision support environment with the vision of making group decisions in general more efficient as well as not to restrict the systems functions to specific application domains. As result of usability studies we could gain a first evidence that users are willing to apply the system in various domains (Stettinger et al. (2013); Stettinger (2014); Stettinger and Felfernig (2014)).

(Q2) How to design the creation process of a group decision task as a configuration problem?

Due to the fact that different group decision tasks require different features, a major requirement is a high degree of decision process flexibility. To achieve the vision of CHOICLA of providing a decision support environment which is independent from the underlying domain, an essential aspect is to integrate configuration knowledge into the process design of a group decision task. This allows the creator of a group decision task to fine-tune the settings and features on a high level of granularity. To keep always an eye on the perceived system's usability, CHOICLA builds upon an intelligent user interface which only displays features which are available in the current configuration context and which prevents users from finding him/herself in an inconsistent state (Stettinger et al. (2014); Felfernig et al. (2014d)).

(Q3) How to best support groups of users in the context of personnel decisions?

Individual perceptions of decision makers differ and thus often lead to subjective evaluations in the context of personnel decisions (Kobrynowicz and Biernat (1997); Graves and Powell (1988); Dougherty et al. (1994)). Literature shows that the assessment criteria are often not stable over the whole hiring procedure and evaluations quickly get subjective in case of interpersonal attraction, first impressions or the appearance of a job candidate (Kobrynowicz and Biernat (1997); Dougherty et al. (1994); Graves and Powell (1988)). CHOICLA facilitates a more objective hiring procedure by providing a flexible interface for defining well structured and objective evaluation criteria suitable for the current job position. In addition to that CHOICLA builds upon a modified MAUT (Stettinger and Felfernig (2014)) approach where the attribute values are subjective and the weights are fixed which is different compared to a typical MAUT scenario (Dyer (2005)).

(Q4) *How to achieve long term fairness in group decision tasks?*

Many decision tasks such as selecting a restaurant for a dinner with friends once a month or finding a place for a monthly meeting with a partner company occur regularly in groups of users. If all past decision outcomes are taken into account, fairness among the group members can be achieved in the long run. If, for example, a group of friends visits a cinema once every three months and the decision regarding which movie to watch in the past three decisions was not the favourite of one group member, the preferences of this group member should have more influence on the current group recommendation. In this thesis we provide insights to one possible way of achieving fairness among group members in recurring decision tasks (Stettinger (2014)).

(Q5.1) *How to counteract serial position effects in group decision support environments?*

The way of presenting a set of decision alternatives has a significant impact on the decision outcome. Items at the beginning and the end of a list are recalled and evaluated significantly more often than those in the middle of a list (Felfernig et al. (2007a); Bar-Hillel (2015); Murphy et al. (2012)). The same phenomenon can also be observed in *description texts* where the ordering of the argumentation items can have a significant impact on the evaluation. If a system uses simple rating interfaces such as five-star scales for the evaluation of the decision alternatives, the description text could have a significant impact on the decision outcome. Our studies showed that if the description text states all negative aspects at the beginning and the end, the five star-based evaluation is on average 1.5 stars less than if the *exact same description* presents all positive aspects at the beginning and the end of the descriptive text. In this thesis we present a possible way to counteract serial position effects by adopting the preference acquisition interface from a star-based scheme to an utility-based scheme (Stettinger et al. (2015b)).

(Q5.2) *How to counteract anchoring effects in the context of group decision scenarios?*

Anchoring effects describe situations in which a decision is biased by the first person who articulates his/her preferences (Adomavicius et al. (2011); Jacowitz and Kahneman (1995); Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). Results from social-psychological studies state that preference visibility among the group members in early stages of the decision process can cause suboptimal decision outcomes (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). On the other hand research from the field of recommender systems points out that in some cases some kind of preference transparency could be a useful functionality of a recommendation system (Jameson (2004)). Visible preferences could help users who are unsure regarding the evaluation and/or could improve user's understanding of the recommendation itself (Jameson (2004); Masthoff (2011)). Preference visibility settings have also a significant impact on the overall amount of information exchange among the participants. In this thesis we initially show the existence of anchoring effects in context of group decision scenarios as well as possibilities of counteracting anchoring effects. In addition to that we discuss results from our user study which confirm that explanations for group decisions play an important role since they can have a positive impact on the overall acceptance of group decisions (Stettinger et al. (2015a)). For this work we received the *James Chen Best Student Paper Award* on the 23rd Conference on User Modelling, Adaptation and Personalization (UMAP 2015).

Table 1.1.: Contributions of this thesis with regard to the research questions.

1.4. Thesis Outline

The thesis contains nine chapters which are organized as follows.

Chapter 1 provides an introduction and motivation of this thesis. In addition to that we summa-

alize our research questions as well as the contributions of this thesis which are related to the field of *group recommender systems*. A description of the structure of the thesis concludes this chapter.

In **Chapter 2** we introduce the research field of recommender systems. In this context we provide an overview of different recommendation approaches such as *collaborative filtering*, *content-based filtering*, *knowledge-based recommendations*, *group recommender systems* as well as *hybrid approaches*.

Chapter 3 gives an overview of knowledge-based recommenders. We show how knowledge-based recommenders can be developed in the context of a Human Computation (von Ahn (2005)) based knowledge acquisition environment called PEOPLEVIEWS. This method can reduce the scalability problems that are likely to occur during the development of knowledge-based recommenders. Furthermore we show in this chapter how the resulting recommendation knowledge can be exploited in a competition-based e-Learning environment called STUDYBATTLES. The presented knowledge acquisition technique could be very helpful for groups of knowledge engineers who have to decide about the structure of constraints.

Chapter 4 provides an introduction to state-of-the-art configuration systems as well as their strengths and limitations. Typical application environments of configuration technologies are "closed" which means that one single or a small group of knowledge engineer(s) is responsible for the development and maintenance of the knowledge bases. In this chapter features are presented which help to achieve "open" configuration environments where it should be possible to jointly configure products and services as well as to cooperatively develop knowledge bases.

In **Chapter 5** we illustrate how to design the creation process of a group decision task as configuration problem. This configuration functionality is needed for finding the right features for a decision task out of a large number of feature combinations. The fragment of the CHOICLA feature model depicted in Figure 5.1 results in 29 valid instances of different decision tasks.

In **Chapter 6** we provide an overview of CHOICLA where feedback concerning features as well as usability aspects from already conducted user studies is integrated. We exemplify the application of CHOICLA in the context of personnel decisions where we introduce new techniques in terms of a modified MAUT (Stettinger and Felfernig (2014)) approach that could achieve more transparent, fair, and structured personnel decisions. As an increment we also show possible approaches to achieve fairness among the group members in the long run. The chapter concludes with the results of a user study which showed that the system is accepted by users and has great potential in various domains.

Chapter 7 focuses on *serial position effects* which are a special form of decision bias. We present the results of a user study where we investigated the correlation between voting strategies and decision

biases and point out the potential of special voting strategies to counteract serial position (primacy/recency) effects.

In **Chapter 8** we concentrated on *anchoring effects* which represent another specific decision bias. On the basis of the conducted user study we discuss the optimal time for disclosing individual preferences and also show that explanations have the potential to increase the satisfaction of group members with various aspects of a group decision process.

Chapter 9 concludes this thesis and presents an outlook on future work.

Basic Approaches in Recommender Systems

This chapter is based on the results documented in Felfernig et al. (2014c). The author of this thesis provided major parts of this chapter in terms of writing and literature research.

2.1. Abstract

Recommendation systems support users in finding items of interest. In this chapter, we introduce the basic approaches of collaborative filtering, content-based filtering, and knowledge-based recommendation. We first discuss principles of the underlying algorithms based on a running example. Thereafter, we provide an overview of hybrid recommendation approaches which combine basic variants. We conclude this chapter with a discussion of newer algorithmic trends, especially critiquing-based and group recommendation.

2.2. Introduction

Recommendation systems (Burke et al. (2011); Jannach et al. (2010)) provide suggestions for items that are of potential interest for a user. These systems are applied for answering questions such as *which book to buy?* (Linden et al. (2003)), *which web site to visit next?* (Pazzani and Billsus (1997)), and *which financial service to choose?* (Felfernig et al. (2007b)). In software engineering scenarios, typical questions that can be answered with the support of recommendation systems are, for example, *which software changes probably introduce a bug?* (Bachwani (2012)), *which requirements to implement in the next software release?* (Felfernig et al. (2012b)), *which stakeholders should participate in the upcoming software project?* (Lim et al. (2010)), *which method calls might be useful in*

the current development context? (Tsunoda et al. (2005)), *which software components (or APIs) to reuse?* (McCarey et al. (2005)), *which software artifacts are needed next?* (Sahm and Maalej (2010)), and *which effort estimation methods should be applied in the current project phase?* (Peischl et al. (2010)). An overview of the application of different types of recommendation technologies in the software engineering context can be found in Robillard et al. (2010).

The major goal of this thesis chapter is to shed light on the basic properties of the three major recommendation approaches of (a) collaborative filtering (Ekstrand et al. (2011b); Goldberg et al. (1992); Konstan et al. (1997)), (b) content-based filtering (Pazzani and Billsus (1997)), and (c) knowledge-based recommendation (Burke (2000); Felfernig et al. (2006)). Starting with the basic algorithmic approaches, we exemplify the functioning of the algorithms and discuss criteria that help to decide which algorithm should be applied in which context.

The remainder of this chapter is organized as follows. In Section 2.3 we give an overview of collaborative filtering recommendation approaches. In Section 2.4 we introduce the basic concepts of content-based filtering. We close our discussion of basic recommendation approaches with the topic of knowledge-based recommendation (see Section 2.5). In Section 2.6, we explain example scenarios for integrating the basic recommendation algorithms into hybrid ones. Hints for practitioners interested in the development of recommender applications are given in Chapter 2.7. A short overview of further algorithmic approaches is presented in Chapter 2.8. This chapter is concluded with Chapter 3.7.

2.3. Collaborative Filtering

The itemset in our running examples is *software engineering related learning material* offered, for example, on an e-learning platform (see Table 2.1). Each learning unit is additionally assigned to a set of categories, for example, the learning unit l_1 is characterized by *Java* and *UML*.

Collaborative filtering (Ekstrand et al. (2011b); Konstan et al. (1997); Takacs et al. (2009)) is based on the idea of word-of-mouth promotion: the opinion of family members and friends plays a major role in personal decision making. In online scenarios (e.g., online purchasing – Linden et al. (2003)), family members and friends are replaced by so-called *nearest neighbors* who are users with a similar preference pattern or purchasing behavior compared to the current user. Collaborative filtering (see Figure 2.1) relies on two different types of *background data*: (a) a set of users and (b) a set of items. The relationship between users and items is primarily expressed in terms of *ratings* which are provided by users and exploited in future recommendation sessions for predicting the rating a user (in our case user U_a) would provide for a specific item. If we assume that user U_a currently interacts with a collaborative filtering recommendation system, the first step of the recommendation system is to identify the *nearest neighbors* (users with a similar rating behavior compared to U_a) and to extrapolate from the ratings of the similar users the rating of user U_a .

The basic procedure of collaborative filtering can best be explained based on a running example

learning unit (LU)	name	Java	UML	Management	Quality
l_1	Data Structures in Java	yes	yes		
l_2	Object Relational Mapping	yes	yes		
l_3	Software Architectures		yes		
l_4	Project Management		yes	yes	
l_5	Agile Processes			yes	
l_6	Object Oriented Analysis		yes	yes	
l_7	Object Oriented Design	yes	yes		
l_8	UML and the UP		yes	yes	
l_9	Class Diagrams		yes		
l_{10}	OO Complexity Metrics				yes

Table 2.1.: Example set of software engineering related learning units (LU) – this set will be exploited for demonstration purposes throughout this chapter. Each of the learning units is additionally characterized by a set of categories (*Java*, *UML*, *Management*, *Quality*), for example, the learning unit l_1 is assigned to the categories *Java* and *UML*.

(see Table 2.2) which is taken from the software engineering domain (collaborative recommendation of learning units). Note that in this chapter we focus on so-called memory-based approaches to collaborative filtering which – in contrast to model-based approaches – operate on uncompressed versions of the user/item matrix (Billsus and Pazzani (1998)). The two basic approaches to collaborative filtering are *user-based collaborative filtering* (Konstan et al. (1997)) and *item-based collaborative filtering* (Sarwar et al. (2001)). Both variants are predicting to which extent the active user would be interested in items which have not been rated by her/him up to now.

LU	name	U_1	U_2	U_3	U_4	U_a
l_1	Data Structures in Java	5.0			4.0	
l_2	Object Relational Mapping	4.0				
l_3	Software Architectures		3.0	4.0	3.0	
l_4	Project Management		5.0	5.0		4.0
l_5	Agile Processes			3.0		
l_6	Object Oriented Analysis		4.5	4.0		4.0
l_7	Object Oriented Design	4.0				
l_8	UML and the UP		2.0			
l_9	Class Diagrams				3.0	
l_{10}	OO Complexity Metrics				5.0	3.0
	average rating (\bar{r}_α)	4.33	3.625	4.0	3.75	3.67

Table 2.2.: Example collaborative filtering data structure (rating matrix): learning units (LU) and related user ratings (we assume a rating scale of 1–5).

User-based Collaborative Filtering. User-based collaborative filtering identifies the k-nearest neighbors of the active user (see Formula 2.1)¹ and – based on these nearest neighbors – calculates a predic-

¹For simplicity we assume k=1 throughout this thesis.

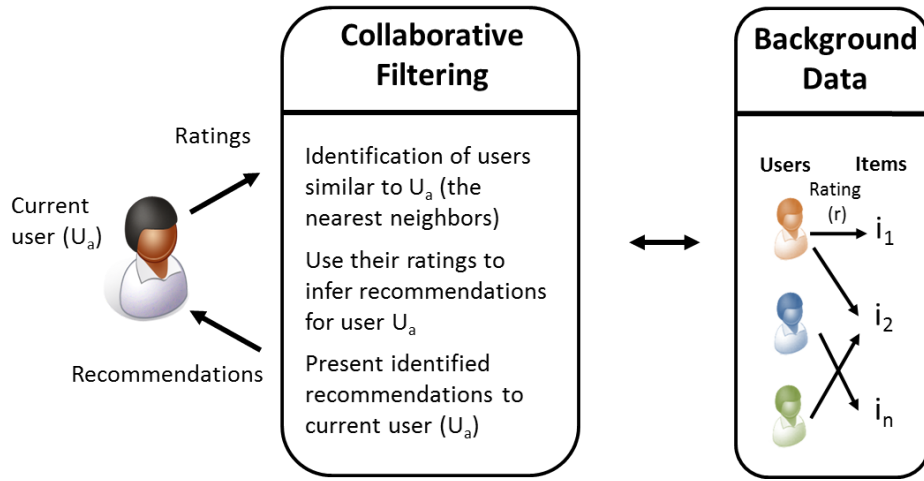


Figure 2.1.: Collaborative filtering (CF) dataflow: users are rating items and receive recommendations for items based on the ratings of users with a similar rating behavior – the nearest neighbors (NN).

tion of the active user’s rating for a specific item (learning unit). In the example of Table 2.2, user U_2 is the nearest neighbor ($k=1$) of user U_a (based on Formula 2.1) and his/her rating of learning unit l_3 will be taken as a prediction for the rating of U_a (rating = 3.0). The similarity between a user U_a (the current user) and another user U_x can be determined, for example, based on the Pearson correlation coefficient Jannach et al. (2010) (see Formula 2.1) where LU_c is the set of items that have been rated by both users, r_{α,l_i} is the rating of user α for item l_i , and \bar{r}_α is the average rating of user α . Similarity values resulting from the application of Formula 2.1 can take values on a scale of $[-1 .. +1]$.

$$similarity(U_a, U_x) = \frac{\sum_{l_i \in LU_c} (r_{a,l_i} - \bar{r}_a) \times (r_{x,l_i} - \bar{r}_x)}{\sqrt{\sum_{l_i \in LU_c} (r_{a,l_i} - \bar{r}_a)^2} \times \sqrt{\sum_{l_i \in LU_c} (r_{x,l_i} - \bar{r}_x)^2}} \quad (2.1)$$

The similarity values for U_a calculated based on Formula 2.1 are shown in Table 2.3. For the purposes of our example we assume the existence of at least two items per user pair (U_i, U_j) ($i \neq j$) in order to be able to determine a similarity. This criterion holds for users U_2 and U_3 .

	U_1	U_2	U_3	U_4
U_a	-	0.97	0.70	-

Table 2.3.: Similarity between user U_a and the users $U_j \neq U_a$ determined based on Formula 2.1. If the number of commonly rated items is below 2, no similarity between the two users is calculated.

A major challenge in the context of estimating the similarity between users is the *sparsity* of the rating matrix since users are typically providing ratings for only a very small subset of the set of offered items. For example, given a large movie dataset that contains thousands of entries, a user will typically be able to rate only a few dozens. A basic approach to tackle this problem is to take

into account the number of commonly rated items in terms of a *correlation significance* (Herlocker et al. (1999)), i.e., the higher the number of commonly rated items, the higher is the significance of the corresponding correlation. For further information regarding the handling of sparsity we refer the reader to Herlocker et al. (1999); Jannach et al. (2010).

The information about the set of users with a similar rating behavior compared to the current user (nearest neighbors NN) is the basis for predicting the rating of user U_a for an *item* that has not been rated up to now by U_a (see Formula 2.2).

$$prediction(U_a, item) = \bar{r}_a + \frac{\sum_{U_j \in NN} similarity(U_a, U_j) \times (r_{j, item} - \bar{r}_j)}{\sum_{U_j \in NN} similarity(U_a, U_j)} \quad (2.2)$$

Based on the rating of the nearest neighbor of U_a , we are able to determine a prediction for user U_a (see Table 2.4). The nearest neighbor of U_a is user U_2 (see Table 2.3). The learning units rated by U_2 but not rated by U_a are l_3 and l_8 . Due to the determined predictions (Formula 2.2), item l_3 would be ranked higher than item l_8 in a recommendation list.

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
U_2	-	-	3.0	5.0	-	4.5	-	2.0	-	-
U_a	-	-	-	4.0	-	4.0	-	-	-	3.0
<i>prediction for U_a</i>	-	-	3.045	-	-	-	-	2.045	-	-

Table 2.4.: User-based collaborative filtering based recommendations (predictions) for items that have not been rated by user U_a up to now.

Item-based Collaborative Filtering. In contrast to user-based collaborative filtering, item-based collaborative filtering searches for items (nearest neighbors – NN) rated by U_a that received similar ratings as items currently under investigation. In our running example, learning unit l_4 has already received a good evaluation (4.0 on a rating scale of 1–5) by U_a . The item which is most similar to l_4 and has not been rated by U_a is item l_3 ($similarity(l_3, l_4) = 0.35$). In this case, the nearest neighbor of item l_3 is l_4 (this calculation is based on Formula 2.3).

If we want to determine a recommendation based on item-based collaborative filtering, we have to determine the similarity (using the Pearson correlation coefficient) between two items l_a and l_b where U denotes the set of users who both rated l_a and l_b , r_{u, l_i} denotes the rating of user u on item l_i , and \bar{r}_i is the average rating of the i -th item.

$$similarity(l_a, l_b) = \frac{\sum_{u \in U} (r_{u, l_a} - \bar{r}_a) \times (r_{u, l_b} - \bar{r}_b)}{\sqrt{\sum_{u \in U} (r_{u, l_a} - \bar{r}_a)^2} \times \sqrt{\sum_{u \in U} (r_{u, l_b} - \bar{r}_b)^2}} \quad (2.3)$$

The information about the set of items with a similar rating pattern compared to the *item* under consideration (nearest neighbors NN) is the basis for predicting the rating of user U_a for the *item* (see

Formula 2.4). Note that in this case NN represents a set of items already evaluated by U_a . Based on the assumption of $k=1$, $prediction(U_a, l_3) = 4.0$, i.e., user U_a would rate item l_3 with 4.0.

$$prediction(U_a, item) = \frac{\sum_{it \in NN} similarity(item, it) \times r_{a,it}}{\sum_{it \in NN} similarity(item, it)} \quad (2.4)$$

For a discussion of advanced collaborative recommendation approaches we refer the reader to Koren et al. (2009); Sarwar et al. (2001).

2.4. Content-based Filtering

Content-based filtering (Pazzani and Billsus (1997)) is based on the assumption of monotonic personal interests. For example, users interested in the topic *Operating Systems* are typically not changing their interest profile from one day to another but will also be interested in the topic in the (near) future. In online scenarios, content-based recommendation approaches are applied, for example, when it comes to the recommendation of websites Pazzani and Billsus (1997) (news items with a similar content compared to the set of already consumed news).

Content-based filtering (see Figure 2.2) relies on two different types of background data: (a) a set of users and (b) a set of categories (or keywords) that have been assigned to (or extracted from) the available items (item descriptions). Content-based filtering recommendation systems calculate a set of items that are most similar to items already known to the current user (U_a).

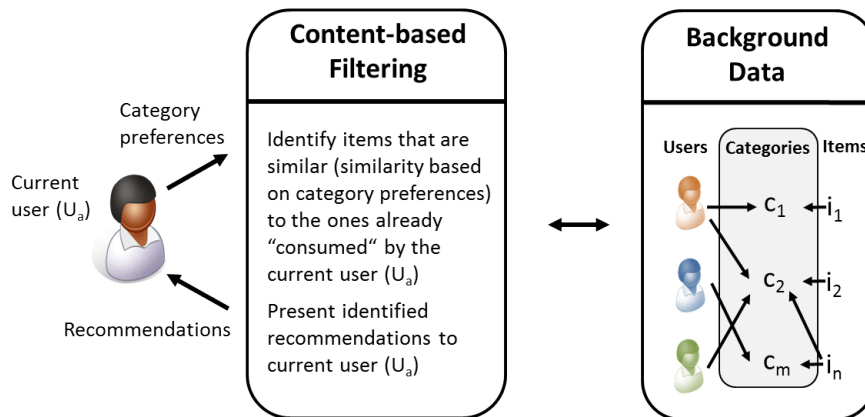


Figure 2.2.: Content-based filtering (CBF) dataflow: users are rating items and receive recommendations of items similar to those that have received a good evaluation from the current user (U_a).

The basic approach of content-based filtering is to compare the content of already consumed items (e.g., a list of news articles) with new items that can potentially be recommended to the user, i.e., to find items that are similar to those already consumed (positively rated) by the user. The basis

for determining such a similarity are *keywords* extracted from the item content descriptions (e.g., keywords extracted from news articles) or *categories* in the case that items have been annotated with the relevant meta-information. Readers interested in the principles of keyword extraction are referred to Jannach et al. Jannach et al. (2010). Within the scope of this chapter we focus on content-based recommendation which exploits item categories (see Table 2.1).

Content-based filtering will now be explained based on a running example which relies on the information depicted in Tables 2.1, 2.5, and 2.6. Table 2.1 provides an overview of the relevant items and the assignments of items to categories. Table 2.5 provides information on which categories are of relevance for the different users. For example, user U_1 is primarily interested in items related to the categories *Java* and *UML*. In our running example, this information has been derived from the rating matrix depicted in Table 2.2. Since user U_a already rated the items l_4 , l_6 , and l_{10} (see Table 2.2), we can infer that U_a is interested in the categories *UML*, *Management*, and *Quality* (see Table 2.5) where items related to the category *UML* and *Management* have been evaluated two times and items related to *Quality* have been evaluated once.

category	U_1	U_2	U_3	U_4	U_a
Java	3 (yes)			1 (yes)	
UML	3 (yes)	4 (yes)	3 (yes)	3 (yes)	2 (yes)
Management		3 (yes)	3 (yes)		2 (yes)
Quality				1 (yes)	1 (yes)

Table 2.5.: Degree of interest in different categories, for example, user U_1 accessed a learning unit related to the category *Java* three times. If a user accessed an item at least once, it is inferred that the user is interested in this item.

LU	rating (U_a)	name	Java	UML	Management	Quality	similarity (U_a, l_i)
l_1		Data Structures in Java	yes	yes			$\frac{2}{5}$
l_2		Object Relational Mapping	yes	yes			$\frac{2}{5}$
l_3		Software Architectures		yes			$\frac{1}{4}$
l_4	4.0	Project Management		yes	yes		—
l_5		Agile Processes			yes		$\frac{2}{4}$
l_6	4.0	Object Oriented Analysis		yes	yes		—
l_7		Object Oriented Design	yes	yes			$\frac{2}{5}$
l_8		UML and the UP		yes	yes		$\frac{4}{5}$
l_9		Class Diagrams		yes			$\frac{1}{4}$
l_{10}	3.0	OO Complexity Metrics				yes	—
user U_a				yes	yes	yes	

Table 2.6.: Example of content-based filtering: user U_a has already consumed the items l_4 , l_6 , and l_{10} (see Table 2.2). The item most similar (see Formula 2.5) to the preferences of U_a is l_8 and is now the best recommendation candidate for the current user.

If we are interested in an item recommendation for the user U_a we have to search for those items which are most similar to the items that have already been consumed (evaluated) by the U_a . This relies on the simple similarity metric shown in Formula 2.5 (Dice coefficient which is a variation of the Jaccard coefficient "intensively" taking into account category commonalities – see also Jannach et al. (2010)). The major difference to the similarity metrics introduced in the context of collaborative filtering is that in this case similarity is measured using keywords (in contrast to ratings).

$$\text{similarity}(U_a, \text{item}) = \frac{2 * |\text{categories}(U_a) \cap \text{categories}(\text{item})|}{|\text{categories}(U_a)| + |\text{categories}(\text{item})|} \quad (2.5)$$

2.5. Knowledge-based Recommendation

Compared to the approaches of collaborative filtering and content-based filtering, *knowledge-based recommendation* (Burke (2000); Felfernig et al. (2007a, 2006); Felfernig and Shchekotykhin (2006); Mandl et al. (2010)) does not primarily rely on item ratings and textual item descriptions but on deep knowledge about the offered items. Such deep knowledge (semantic knowledge Felfernig et al. (2006)) describes an item in more detail and thus allows for a different recommendation approach (see Table 2.7).

LU	name	obligatory	duration	semester	complexity	topics	eval
l_1	Data Structures in Java	yes	2	2	3	Java,UML	4.5
l_2	Object Relational Mapping	yes	3	3	4	Java,UML	4.0
l_3	Software Architectures	no	3	4	3	UML	3.3
l_4	Project Management	yes	2	4	2	UML,Management	5.0
l_5	Agile Processes	no	1	3	2	Management	3.0
l_6	Object Oriented Analysis	yes	2	2	3	UML,Management	4.7
l_7	Object Oriented Design	yes	2	2	3	Java,UML	4.0
l_8	UML and the UP	no	3	3	2	UML,Management	2.0
l_9	Class Diagrams	yes	4	3	3	UML	3.0
l_{10}	OO Complexity Metrics	no	3	4	2	Quality	5.0

Table 2.7.: Software engineering learning units (LU) described based on *deep knowledge*, for example, *obligatory* vs. *non-obligatory*, *duration* of consumption, recommended *semester*, *complexity* of the learning unit, associated *topics*, and average user rating (*eval*).

Knowledge-based recommendation (see Figure 2.3) relies on the following background data: (a) a set of rules (constraints) or similarity metrics and (b) a set of items. Depending on the given user requirements, rules (constraints) describe which items have to be recommended. The current user (U_a) articulates his/her requirements (preferences) in terms of item property specifications which are internally as well represented in terms of rules (constraints). In our example, constraints are represented solely by user requirements, no further constraint types are included (e.g., constraints that explicitly specify compatibility or incompatibility relationships). An example of such a constraint is *topics = Java*. It denotes the fact that the user is primarily interested in Java-related learning units. For a detailed discussion of further constraint types we refer the reader to Felfernig et al. Felfernig et al. (2006). Constraints are interpreted and the resulting items are presented to the user.² A detailed discussion of reasoning mechanisms that are used in knowledge-based recommendation can be found, for example, in Felfernig et al. Felfernig et al. (2006, 2009, 2013e).

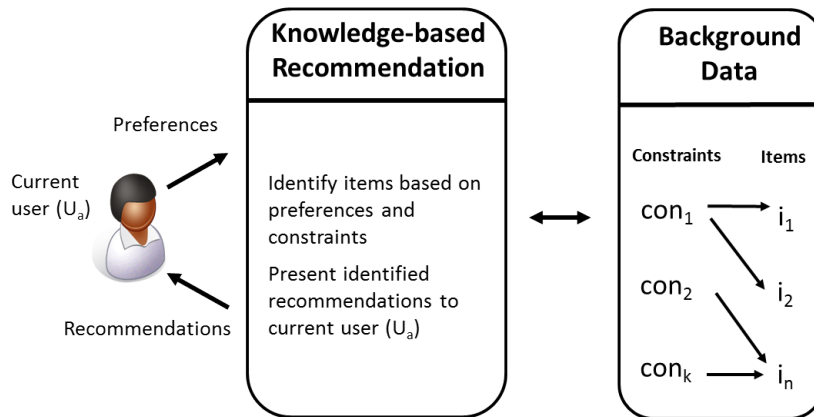


Figure 2.3.: Knowledge-based recommendation (KBR) dataflow: users are entering their preferences and receive recommendations based on the interpretation of a set of rules (constraints).

In order to determine a recommendation in the context of knowledge-based recommendation scenarios, a *recommendation task* has to be solved.

Definition (Recommendation Task). A recommendation task can be defined by the tuple (R, I) where R represents a set of user requirements and I represents a set of items (in our case: software engineering learning units $l_i \in LU$). The goal is to identify those items in I which fulfill the given user requirements (preferences).

A solution for a recommendation task (also denoted as recommendation) can be defined as follows.

Definition (Solution for a Recommendation Task). A solution for a recommendation task (R, I) is a set $S \subseteq I$ such that $\forall l_i \in S: l_i \in \sigma_{(R)} I$ where σ is the selection operator of a conjunctive query (Felfernig et al. (2009)), R represents a set of selection criteria (represented as constraints), and I represents an

²Knowledge-based recommendation approaches based on the determination of similarities between items will be discussed in Section 2.8.

item table (see, e.g., Table 2.7). If we want to restrict the set of item properties shown to the user in a result set (recommendation), we have to additionally include projection criteria π as follows: $\pi_{(attributes(I))}(\sigma_{(R)}I)$.

In our example, we show how to determine a solution for a given recommendation task based on a conjunctive query where user requirements are used as selection criteria (constraints) on an item table I . If we assume that the user requirements are represented by the set $R = \{r_1 : semester \leq 3, r_2 : topics = Java\}$ and the item table I consists of the elements shown in Table 2.7 then $\pi_{(LU)}(\sigma_{(semester \leq 3 \wedge topics = Java)}I) = \{l_1, l_2, l_7\}$, i.e., these three items are consistent with the given set of requirements.

Ranking items. Up to this point we only know which items can be recommended to a user. One wide-spread approach to rank items is to define a utility scheme which serves as a basis for the application of multi attribute utility theory (MAUT).³ Alternative items can be evaluated and ranked with respect to a defined set of interest dimensions. In the domain of e-learning units, example interest dimensions of users could be *time effort* (time needed to consume the learning unit) and *quality* (quality of the learning unit). The first step to establish a MAUT scheme is to relate the interest dimensions with properties of the given set of items. A simple example of such a mapping is shown in Table 2.8. In this example, we assume that obligatory learning units (learning units that have to be consumed within the scope of a study path) trigger more time efforts than non-obligatory ones, a longer duration of a learning unit is correlated with higher time efforts, and low complexity correlates with lower time efforts. In this context, lower time efforts for a learning unit are associated with a higher utility. Furthermore, we assume that the more advanced the semester, the higher is the quality of the learning unit (e.g., in terms of education degree). The better the overall evaluation (eval), the higher the quality of a learning unit (e.g., in terms of the used pedagogical approach).

We are now able to determine the user-specific utility of each individual item. The calculation of *item* utilities for a specific user U_a can be based on Formula 2.6.

$$utility(U_a, item) = \sum_{d \in Dimensions} contribution(item, d) \times weight(U_a, d) \quad (2.6)$$

If we assume that the current user U_a assigns a weight of 0.2 to the dimension *time effort* ($weight(U_a, time\ effort) = 0.2$) and a weight of 0.8 to the dimension *quality* ($weight(U_a, quality) = 0.8$) then the user-specific utilities of the individual items (l_i) are the ones shown in Table 2.9.

Dealing with Inconsistencies. Due to the logical nature of knowledge-based recommendation problems, we have to deal with scenarios where no solution (recommendation) can be identified for a given set of user requirements, i.e., $\sigma_{(R)}I = \emptyset$. In such situations we are interested in proposals for requirements changes such that a solution (recommendation) can be identified. For ex-

³A detailed discussion of the application of MAUT in knowledge-based recommendation scenarios can be found in Ardissono et al. (2003); Felfernig et al. (2006, 2008).

item property	time effort [1–10]	quality [1–10]
obligatory = yes	4	-
obligatory = no	7	-
duration = 1	10	-
duration = 2	5	-
duration = 3	1	-
duration = 4	1	-
complexity = 2	8	-
complexity = 3	5	-
complexity = 4	2	-
semester = 2	-	3
semester = 3	-	5
semester = 4	-	7
eval = 0–2	-	2
eval = >2–3	-	5
eval = >3–4	-	8
eval = >4	-	10

Table 2.8.: Contributions of item properties to the dimensions *time effort* and *quality*.

ample, if a user is interested in learning units with a duration of 4 hours, related to management, and a complexity level > 3 , then no solution can be provided for the given set of requirements $R = \{r_1 : duration = 4, r_2 : topics = management, r_3 : complexity > 3\}$.

User support in such situations can be based on the concepts of conflict detection (Junker (2004)) and model-based diagnosis (Falkner et al. (2011); Felfernig et al. (2004); Reiter (1987)). A conflict (or conflict set) with regard to an item set I in a given set of requirements R can be defined as follows.

Definition (Conflict Set). A conflict set is a set $CS \subseteq R$ such that $\sigma_{(CS)}I = \emptyset$. CS is minimal if there does not exist a conflict set CS' with $CS' \subset CS$.

In our running example we are able to determine the following minimal conflict sets CS_i : $CS_1 : \{r_1, r_2\}$, $CS_2 : \{r_2, r_3\}$. We will not discuss algorithms that support the determination of minimal conflict sets but refer the reader to the work of Junker (2004) who introduces a divide-and-conquer based algorithm with a logarithmic complexity in terms of the needed number of consistency checks.

Based on the identified minimal conflict sets, we are able to determine the corresponding (minimal) diagnoses. A diagnosis for a given set of requirements which is inconsistent with the underlying item table can be defined as follows.

Definition (Diagnosis). A diagnosis for a set of requirements $R = \{r_1, r_2, \dots, r_n\}$ is a set $\Delta \subseteq R$ such that $\sigma_{(R-\Delta)}I \neq \emptyset$. A diagnosis Δ is minimal if there does not exist a diagnosis Δ' with $\Delta' \subset \Delta$.

In other words, a diagnosis (hitting set) is a minimal set of requirements that have to be deleted from R such that a solution can be found for $R - \Delta$. The determination of the complete set of diagnoses

LU	time effort	quality	utility
l_1	14	13	$2.8+10.4=13.2$
l_2	7	13	$1.4+10.4=11.8$
l_3	13	15	$2.6+12.0=14.6$
l_4	17	17	$3.4+13.6=17.0$
l_5	25	10	$5.0+8.0=13.0$
l_6	14	13	$2.8+10.4=13.2$
l_7	14	11	$2.8+8.8=11.6$
l_8	16	7	$3.2+5.6=8.8$
l_9	10	10	$2.0+8.0=10.0$
l_{10}	16	17	$3.2 + 13.6 = 16.8$

Table 2.9.: Item-specific utility for user U_a ($utility(U_a, l_i)$) assuming the personal preferences for *time effort* = 0.2 and *quality* = 0.8. In this scenario, item l_4 has the highest utility for user U_a .

for a set of requirements inconsistent with the underlying item table (the corresponding conjunctive query results in \emptyset) is based on the construction of hitting set trees (Reiter (1987)). An example of the determination of minimal diagnoses is depicted in Figure 2.4. There are two possibilities of resolving the conflict set CS_1 . If we decide to delete the requirement r_2 , $\sigma_{\{r_1, r_3\}}I \neq \emptyset$, i.e., a diagnosis has been identified ($\Delta_1 = \{r_2\}$) and – as a consequence – all CS_i have been resolved. Choosing the other alternative and resolving CS_1 by deleting r_1 does not result in a diagnosis since the conflict CS_2 is not resolved. Resolving CS_2 by deleting r_2 does not result in a minimal diagnosis, since r_2 already represents a diagnosis. The second (and last) minimal diagnosis that can be identified in our running example is $\Delta_2 = \{r_1, r_3\}$. For a detailed discussion of the underlying algorithm and analysis we refer the reader to Reiter (1987). Note that a diagnosis provides a hint to which requirements have to be changed. For a discussion of how requirement repairs (change proposals) are calculated, we refer the reader to Felfernig et al. (2009).

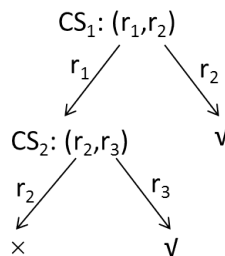


Figure 2.4.: Determination of the complete set of diagnoses (hitting sets) Δ_i for the given conflict sets $CS_1 = \{r_1, r_2\}$ and $CS_2 = \{r_2, r_3\}$: $\Delta_1 = \{r_2\}$ and $\Delta_2 = \{r_1, r_3\}$.

2.6. Hybrid Recommendations

After having discussed the three basic recommendation approaches of collaborative filtering, content-based filtering, and knowledge-based recommendation, we will now present some possibilities to combine these basic types.

The motivation for hybrid recommendation is the opportunity to achieve a better accuracy (Burke (2002)). There are different approaches to evaluate the accuracy of recommendation algorithms. These approaches can be categorized into *predictive accuracy metrics* such as the mean absolute error (MAE), *classification accuracy metrics* such as precision and recall, and *rank accuracy metrics* such as Kendall's Tau. For a discussion of accuracy metrics we refer the reader also to Gunawardana and Shani Gunawardana and Shani (2009) and Jannach et al. Jannach et al. (2010).

We now take a look at example design types of hybrid recommendation approaches (Burke (2002); Jannach et al. (2010)) which are *weighted*, *mixed*, and *cascade* (see Table 2.10). These approaches will be explained on the basis of our running example. The basic assumption in the following is that individual recommendation approaches return a list of *five* recommended items where each item has an assigned (recommender-individual) prediction out of $\{1.0, 2.0, 3.0, 4.0, 5.0\}$. For a more detailed discussion of hybridization strategies we refer the reader to Burke Burke (2002) and Jannach et al. Jannach et al. (2010).

method	description	example formula
weighted	predictions (s) of individual recommenders are summed up	$\text{score}(\text{item}) = \sum_{\text{rec} \in \text{RECS}} s(\text{item}, \text{rec})$
mixed	recommender-individual predictions (s) are combined into one recommendation result	$\text{score}(\text{item}) = \text{zipper-function}(\text{item}, \text{RECS})$
cascade	the prediction of one recommender is used as input for the next recommender	$\text{score}(\text{item}) = \text{score}(\text{item}, \text{rec}_n)$ $\text{score}(\text{item}, \text{rec}_i) = \begin{cases} s(\text{item}, \text{rec}_1), & \text{if } i = 1 \\ s(\text{item}, \text{rec}_i) * \text{score}(\text{item}, \text{rec}_{i-1}), & \text{otherwise.} \end{cases}$

Table 2.10.: Examples of hybrid recommendation approaches (RECS = set of recommenders, s = recommender-individual prediction, score = item score).

Weighted. Weighted hybrid recommendation is based on the idea of deriving recommendations by combining the results (predictions) computed by individual recommenders. A corresponding example

is depicted in Table 2.11 where the individual item scores of a collaborative and a content-based recommender are summed up. Item l_8 receives the highest overall score (9.0) and is ranked highest by the weighted hybrid recommender.⁴

items	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
$s(l_i, \text{collaborative filtering})$	1.0	3.0	-	5.0	-	2.0	-	4.0	-	-
$s(l_i, \text{content-based filtering})$	-	1.0	2.0	-	-	3.0	4.0	5.0	-	-
$\text{score}(l_i)$	1.0	4.0	2.0	5.0	0.0	5.0	4.0	9.0	0.0	0.0
$\text{ranking}(l_i)$	7	4	6	2	8	3	5	1	9	10

Table 2.11.: Example of *weighted* hybrid recommendation: individual predictions are integrated into one score. Item l_8 receives the best overall score (9.0).

Mixed. Mixed hybrid recommendation is based on the idea that predictions of individual recommenders are shown in one integrated result. For example, the results of a collaborative filtering and a content-based recommender can be ranked as sketched in Table 2.12. Item scores can be determined, for example, on the basis of the zipper principle, i.e., the item with highest collaborative filtering prediction value receives the highest overall score (10.0), the item with best content-based filtering prediction value receives the second best overall score, and so forth.

items	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
$s(l_i, \text{collaborative filtering})$	1.0	3.0	-	5.0	-	2.0	-	4.0	-	-
$s(l_i, \text{content-based filtering})$	-	1.0	2.0	-	-	3.0	4.0	5.0	-	-
$\text{score}(l_i)$	4.0	8.0	5.0	10.0	0.0	6.0	7.0	9.0	0.0	0.0
$\text{ranking}(l_i)$	7	3	6	1	8	5	4	2	9	10

Table 2.12.: Example of *mixed* hybrid recommendation: individual predictions are integrated into one score conform the zipper principle (best collaborative filtering prediction receives score=10.0, best content-based filtering prediction receives score=9.0 and so forth).

Cascade. The basic idea of cascade-based hybridization is that recommenders in a pipe of recommenders exploit the recommendation of the upstream recommender as a basis for deriving their own recommendation. The knowledge-based recommendation approach presented in Section 2.5 is an example of a cascade-based hybrid recommendation approach. First, items that are consistent with the given requirements are preselected by a conjunctive query Q . We can assume, for example, that $s(\text{item}, Q) = 1.0$ if the item has been selected and $s(\text{item}, Q) = 0.0$ if the item has not been selected. In our case, the set of requirements $\{r_1 : \text{semester} \leq 3, r_2 : \text{topics} = \text{Java}\}$ in the running example leads to the selection of the items $\{l_1, l_2, l_7\}$. Thereafter, these items are ranked conform to their utility for the current user (utility-based ranking U). The utility-based ranking (U) would determine the item order $\text{utility}(l_1) > \text{utility}(l_2) > \text{utility}(l_7)$ assuming that the current user assigns a weight of 0.8 to the interest dimension *quality* ($\text{weight}(U_a, \text{quality}) = 0.8$) and a weight of 0.2 to the interest dimensions

⁴If two or more items have the same overall score, a possibility is to force a decision by lot; where needed, this approach can also be applied by other hybrid recommendation approaches.

time effort ($\text{weight}(U_a, \text{time effort}) = 0.2$). In this example the recommender Q is the first one and the results of Q are forwarded to the utility-based recommender.

Other examples of hybrid recommendation approaches are the following (Burke (2002)). *Switching* denotes an approach where – depending on the current situation – a specific recommendation approach is chosen. For example, if a user has a low level of product knowledge, then a critiquing-based recommender will be chosen (see Section 2.8). Vice-versa, if the user is an expert, an interface will be provided where the user is enabled to explicitly state his/her preferences on a detailed level. *Feature combination* denotes an approach where different data sources are exploited by a single recommender. For example, a recommendation algorithm could exploit semantic item knowledge in combination with item ratings (see Table 2.7). For an in-depth discussion of hybrid recommenders we refer the reader to Burke (2002); Jannach et al. (2010).

2.7. Hints for Practitioners

2.7.1. Usage of Algorithms

The three basic approaches of collaborative filtering, content-based filtering, and knowledge-based recommendation exploit different sources of recommendation knowledge and have different strengths and weaknesses (see Table 2.13). Collaborative filtering (CF) as well as content-based filtering (CBF) are easy to set up (only basic item information is needed, e.g., item name and picture) whereas knowledge-based recommendation requires a more detailed specification of item properties (and in many cases also additional constraints). Both, CF and CBF are more adaptive in the sense that new ratings are automatically taken into account in future activations of the recommendation algorithm. In contrast, utility schemes in knowledge-based recommendation (see, for example, Table 2.9) have to be adapted manually (if no additional learning support is available Felfernig et al. (2013d)).

Serendipity effects are interpreted as a kind of accident of being confronted with something useful although no related search has been triggered by the user. They can primarily be achieved when using CF approaches. Due to the fact that content-based filtering does not take into account the preferences (ratings) of other users, no such effects can be achieved. Achieving serendipity effects for the users based on KBR is possible in principle, however, restricted to and depending on the creativity of the knowledge engineer (who is able to foresee such effects when defining recommendation rules). The term *ramp-up* problem denotes a situation where there is the need to provide initial rating data before the algorithm is able to determine reasonable recommendations. Ramp-up problems exist with both, CF as well as CBF: in CF users have to rate a set of items before the algorithm is able to determine the nearest neighbors. In CBF, the user has to specify interesting/relevant items before the algorithm is able to determine items that are similar to those that have already been rated by the user.

Finally, *transparency* denotes the degree to which recommendations can be explained to users. Explanations in CF systems solely rely on the interpretation of the relationship to nearest neighbors, for

example, *users who purchased item X also purchased item Y*. CBF algorithms explain their recommendations in terms of the similarity of the recommended item to items already purchased by the user: *we recommend Y since you already purchased X which is quite similar to Y*. Finally – due to the fact that they rely on deep knowledge – KBR are able to provide deep explanations which take into account semantic item knowledge. An example of such an explanation are diagnoses which explain the reasons as to why a certain set of requirements does not allow the calculation of a solution. Other types of explanations exist: why a certain item has been included in the recommendation and why a certain question has been asked to the user (Felfernig et al. (2006,?)).

Typically, CF and CBF algorithms are used for recommending low-involvement items⁵ such as movies, books, and news articles. In contrast, knowledge-based recommender functionalities are used for the recommendation of high-involvement items such as financial services, cars, digital cameras, and apartments. In the latter case, ratings are provided with a low frequency which makes these domains less accessible to CF and CBF approaches. For example, user preferences regarding a car could significantly change within a couple of years without being detected by the recommender system whereas such preference shifts are detected by collaborative and content-based recommendation approaches due to the fact that purchases occur more frequently and – as a consequence – related ratings are available for the recommender system. For an overview of heuristics and rules related to the selection of recommendation approaches we refer the reader to Burke and Ramezani Burke and Ramezani (2010).

algorithm	CF	CBF	KBR
easy setup	yes	yes	no
adaptivity	yes	yes	no
serendipity effects	yes	no	no
ramp-up problem	yes	yes	no
transparency	no	no	yes
high-involvement items	no	no	yes

Table 2.13.: Summarization of the strengths and weaknesses of collaborative filtering (CF), content-based filtering (CBF), and knowledge-based recommendation (KBR).

2.7.2. Recommendation Environments

Recommendation is an Artificial Intelligence (AI) technology successfully applied in different commercial contexts (Felfernig et al. (2013b)). As recommendation algorithms and heuristics are regarded as a major intellectual property of a company, recommender systems are often not developed on the basis of standard solutions but are rather based on proprietary solutions that are tailored to the specific

⁵The impact of a wrong decision (selection) is rather low, therefore users invest less evaluation efforts in a purchase situation.

situation of the company. Despite this situation, there exist a couple of recommendation environments that can be exploited for the development of different recommender applications.

Strands⁶ is a company that provides recommendation technologies covering the whole range of collaborative, content-based, and knowledge based recommendation approaches. MyMediaLite⁷ is an open-source library that can be used for the development of collaborative filtering based recommender systems. LensKit⁸ (Ekstrand et al. (2011a)) is an open source toolkit that supports the development and evaluation of recommender systems – specifically it includes implementations of different collaborative filtering algorithms. A related development is MovieLens⁹ which is a non-commercial movie recommendation platform. The MovieLens dataset (user \times item ratings) is publicly available and popular dataset for evaluating new algorithmic developments. Apache Mahout¹⁰ is a machine learning environment that also includes recommendation functionalities such as user-based and item-based collaborative filtering.

Open source constraint libraries such as Choco¹¹ and Jacop¹² can be exploited for the implementation of knowledge-based recommender applications. WeeVis¹³ is a Wiki-based environment for the development of knowledge-based recommender applications – resulting recommender applications can be deployed on different handheld platforms such as iOS, Android, and Windows 8. Finally, Choicla¹⁴ is a group recommendation platform that allows the definition and execution of group recommendation tasks (see Section 2.8).

2.8. Further Algorithmic Approaches

2.8.1. Critiquing-based recommendation

There are two basic approaches to support item identification in the context of knowledge-based recommendation.

First, *search-based* approaches require the explicit specification of search criteria and the recommendation algorithm is in charge of identifying a set of corresponding recommendations Felfernig et al. (2006); Tiihonen and Felfernig (2010) (see also Section 2.5). If no solution can be found for a given set of requirements, the recommendation engine determines diagnoses that indicate potential changes such that a solution (recommendation) can be identified. Second, *navigation-based* approaches support the navigation in the item space where in each iteration a reference item is presented

⁶strands.com

⁷www.mymedialite.net

⁸lenskit.grouplens.org

⁹www.movielens.org

¹⁰mahout.apache.org

¹¹www.emn.fr

¹²jacop.osolpro.com

¹³www.weevis.org

¹⁴choicla.com

to the user and the user either accepts the (recommended) item or searches for different solutions by specifying *critiques*. Critiques are simple criteria that are used for determining new recommendations that take into account the (changed) preferences of the current user. Examples of such critiques in the context of our running example are *less time efforts* and *higher quality* (see Figure 2.5). Critiquing-based recommendation systems are useful in situations where users are not experts in the item domain and prefer to specify their requirements on the level of critiques (Knijnenburg et al. (2011)). If users are knowledgeable in the item domain, the application of search-based approaches makes more sense. For an in-depth discussion of different variants of critiquing-based recommendation we refer the reader to Burke et al. (1997); Chen and Pu (2012); Gräsch et al. (2013); Mandl and Felfernig (2012); McCarthy et al. (2004); Ricci and Nguyen (2007).

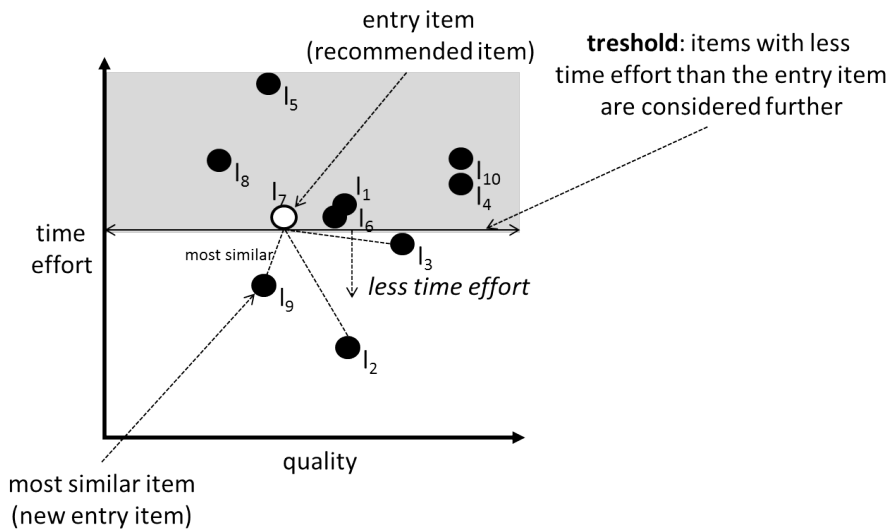


Figure 2.5.: Example of a critiquing scenario: an entry item (l_7) is shown to the user. The user specifies the critique "less time effort". The new entry item is l_9 since it is consistent with the critique and the item most similar to l_7 .

2.8.2. Group recommendation

Due to the increasing popularity of social platforms and online communities, group recommendation systems are becoming an increasingly important technology (Hennig-Thurau et al. (2012); Masthoff (2011)). Example application domains of group recommendation technologies include tourism McCarthy et al. (2006) (e.g., *which hotels or tourist destinations should be visited by a group?*) and interactive television Masthoff (2004) (*which sequence of television programs will be accepted by a group?*). In the majority, group recommendation algorithms are related to simple items such as hotels, tourist destinations, and television programs. The application of group recommendation in the context of our running example is shown in Table 2.14 (selection of a learning unit for a group).

items	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
alex	1.0	3.0	1.0	5.0	4.0	2.0	4.0	2.0	1.0	4.0
dorothy	5.0	1.0	2.0	1.0	4.0	3.0	4.0	2.0	2.0	3.0
peter	2.0	4.0	2.0	5.0	3.0	5.0	4.0	3.0	2.0	2.0
ann	3.0	4.0	5.0	2.0	1.0	1.0	3.0	3.0	3.0	4.0
<i>least misery</i>	1.0	1.0	1.0	1.0	1.0	1.0	3.0	2.0	1.0	2.0

Table 2.14.: Example of group recommendation: selection of a learning unit for a group. The recommendation (l_7) is based on the *least misery* heuristic.

The group recommendation task is to figure out a recommendation that will be accepted by the whole group. The group decision heuristics applied in the context is *least misery* which returns the lowest voting for alternative l_i as group recommendation. For example, the *least misery* value for alternative l_7 is 3.0 which is the highest value of all possible alternatives, i.e., the first recommendation for the group is l_7 . Other examples of group recommendation heuristics are *most pleasure* (the group recommendation is the item with the highest individual voting) and *majority voting* (the voting for an individual solution is defined by the majority of individual user votings - the group recommendation is the item with the highest majority value). Group recommendation technologies for high-involvement items (see Section 2.7) are the exception of the rule (see, e.g., Jameson (2004); Stettinger et al. (2013)). First applications of group recommendation technologies in the software engineering context are reported in Felfernig et al. (2012b). An in-depth discussion of different types of group recommendation algorithms can be found in Masthoff O'Connor et al. (2001); Jameson and Smyth (2007); Masthoff (2011).

2.9. Conclusions

This chapter provides an introduction to the recommendation approaches of collaborative filtering, content-based filtering, knowledge-based recommendation, and different hybrid variants thereof. While collaborative filtering based approaches exploit ratings of nearest neighbors, content-based filtering exploits categories and/or extracted keywords for determining recommendations. Knowledge-based recommenders should be used, for example, for products where there is a need to encode the recommendation knowledge in terms of constraints. Beside algorithmic approaches, we discussed criteria to be taken into account when deciding about which recommendation technology to use in a certain application context. Furthermore, we provided an overview of environments that can be exploited for recommender application development.

Human Computation Based Acquisition Of Financial Service Advisory Practices

This chapter is based on the results documented in Felfernig et al. (2015). The author of this thesis provided major contributions to this chapter in terms of user interface design and literature analyses.

3.1. Abstract

Knowledge-based recommenders support an easier comprehension of complex item assortments (e.g., financial services and electronic equipment). In this chapter we show (1) how such recommenders can be developed in a Human Computation based knowledge acquisition environment (PEOPLEVIEWS) and (2) how the resulting recommendation knowledge can be exploited in a competition-based e-Learning environment (STUDYBATTLE).

3.2. Introduction

Knowledge-based recommenders (Burke (2000)) support users on the basis of semantic knowledge about the item (product) domain.¹ One variant of knowledge-based recommenders are *constraint-based recommenders* (Felfernig and Burke (2008)) which exploit explicit constraints (rules) that encode the recommendation knowledge. Another variant are *critiquing-based recommenders* (Burke et al. (1997)): new items are presented to the user as long as the user is unsatisfied and articulates critiques (e.g., an item should be *cheaper*). In critiquing-based recommendation, new items are determined by similarity functions. For a detailed overview of recommendation approaches we refer to Burke et al. (2011); Jannach et al. (2010).

¹The terms *item* and *product* are used synonymously throughout the chapter.

In this chapter we focus on constraint-based recommenders, i.e., recommenders that are based on explicit recommendation rules (constraints). The development of such recommenders is often a time-consuming and error-prone process which can be primarily explained by the *knowledge acquisition bottleneck*: in the formalization of product domain and recommendation knowledge, misunderstandings can occur and as a result knowledge engineers encode this knowledge in an unintended fashion. The more recommenders have to be developed and maintained the higher the risk that the organization runs into a scalability problem where additional resources are needed to be able to perform knowledge engineering and maintenance.

An alternative to the hiring of additional staff for development and maintenance of recommendation knowledge bases is to change the underlying knowledge engineering paradigm. The idea of PEOPLE-VIEWS is to engage domain experts more deeply into knowledge engineering tasks. We do not want to "convert" them into technical experts but to define basic tasks (*micro tasks*) that are easy to understand and complete even for domain experts without the corresponding technical expertise. Micro tasks completed by users provide knowledge chunks that can be aggregated into a PEOPLEVIEWS recommender knowledge base.

The resulting PEOPLEVIEWS recommenders support customers (and especially in the financial services domain also sales representatives) in finding products that fit their wishes and needs. Using such a recommender, items are retrieved within the scope of a dialog (these systems are often also denoted as conversational) where users articulate their requirements and the system tries to identify corresponding solutions. Major advantages of such systems are reduced error rates in the phase of order acquisition, more time that can be invested in contacting new customers due to fewer errors, more satisfied customers, and also pre-informed customers due to the fact that recommender applications can be made publicly available.

Knowledge-based recommender systems have been applied in various item domains – due to the diversity of applications, we can only give some examples of applications of these systems. In the financial services domain, for example, the following applications of knowledge-based recommendation technologies are reported in the literature. Felfernig et al. (Felfernig et al. (2007b); Felfernig and Kiener (2005)) show an application in the context of investment decisions where recommenders are provided to sales representatives who exploit the recommenders in sales dialogs. Time savings are reported as one of the major improvements directly related to the application of recommendation technologies. Another application of knowledge-based technologies in financial services is presented by Fano and Kurth (Fano and Kurth (2003)) who introduce a simulation environment that can directly visualize the effects of financial decisions on the financial situation of a family.

Felfernig et al. (Felfernig et al. (2006)) present a digital camera recommender deployed on a large Austrian product comparison platform. Peischl et al. (Peischl et al. (2010)) show the application of constraint-based recommendation technologies in the domain of software effort estimation. WEE-

VIS(Reiterer et al. (2013))² is a MediaWiki³ based environment for the development and maintenance of constraint-based recommender applications – a couple of freely available recommenders have already been deployed. Knowledge-based technologies for the recommendation of business plans are introduced by Jannach and Bundgaard-Joergensen (Jannach and Bundgaard-Joergensen (2007)). The recommendation of equipment configuration in the context of smarthomes is introduced by Leitner et al. (Leitner et al. (2012)). Technologies that recommend changes in software development practices are introduced by Pribik and Felfernig (Pribik and Felfernig (2012)). Finally, Burke and Ramezani (Burke and Ramezani (2010)) show how to select recommendation algorithms by introducing rules for *recommending recommenders*.

In PEOPLEVIEWS, principles of Human Computation (von Ahn (2005)) are included into the development of knowledge-based recommenders. The idea of Human Computation is to let persons perform tasks in which they are better than computers, for example, the identification of product properties from a website. In the context of knowledge base development and maintenance the idea is to let domain experts perform tasks they are much better in compared to knowledge engineers who typically have less knowledge about the product domain and thus relieve the work of knowledge engineers. MATCHIN (Hacker and VonAhn (2009)) is based on the idea of preference elicitation by asking users what a person would typically prefer when having to choose between alternatives. Compared to this work, PEOPLEVIEWS allows to derive constraint-based recommenders which are the basis for intelligent user interfaces that support, for example, deep explanations (Friedrich (2004)) and the diagnosis and repair of inconsistent requirements (Felfernig et al. (2009, 2013e)).

The major contributions of this chapter are the following. First, we show how financial service recommender knowledge bases can be developed by a community of domain experts. Second, we sketch how such knowledge bases can also be exploited for teaching advisory practices on the basis of games (STUDYBATTLE environment). Third, we provide a discussion of major issues for future research.

The remainder of this chapter is organized as follows. In Section 3.3 we introduce basic concepts of Human Computation based knowledge construction. To give an impression of the PEOPLEVIEWS and the STUDYBATTLE user interface, we present example screenshots in Section 3.4. Preliminary results of empirical evaluations are shortly discussed in Section 3.5. In Section 3.6 we provide an overview of issues for future work. We conclude the chapter with Section 3.7.

3.3. Developing PEOPLEVIEWS Recommenders

The PEOPLEVIEWS environment supports two basic modes of interaction. First, recommender applications can be created in the *modeling mode* and second, the applications can be executed in the

²www.weevis.org.

³www.mediawiki.org.

user	email	pwd
Andrea	andrea@...	****
Mary	mary@...	*****
Luc	luc@...	*****
Torsten	torsten@...	****

Table 3.1.: Example users of PEOPLEVIEWS environment.

id	item name
Φ_1	Investment Fund A
Φ_2	Investment Fund B
Φ_3	Building Loan
Φ_4	Bond
Φ_5	Savings Book

Table 3.2.: Example set of items used in working example.

recommendation mode. In this chapter we discuss different tasks to be performed in order to create a PEOPLEVIEWS recommender. Table 3.1 provides an overview of the users of our working example. These users will jointly develop a PEOPLEVIEWS recommender.

Table 3.2 contains an overview of items (financial services) that are used in our working example. The *Investment Funds* (*A* and *B*) have a higher risk of loss and require that customers have a high willingness to take risks, otherwise these services will not be recommended. *Building Loan*, *Bond*, and *Savings Book* are lower-risk items. In the current version of PEOPLEVIEWS, items can be characterized by additional item attributes, however, these attributes are not used by recommendation rules constructed from micro contributions.

In PEOPLEVIEWS, user requirements $req_i \in REQ$ are specified as assignments of *user attributes*. For our financial services recommender we define a set of user attributes which are enumerated in Table 3.3. In the current version of the system, user attributes are defined by the creators of a recommender application, i.e., attribute definitions can not be extended by other users who contribute to the further development of the application on the basis of micro tasks.

In the PEOPLEVIEWS *recommendation mode*, user attributes can be used to specify user (customer) requirements $req_i \in REQ$. In the *modeling mode*, user attributes represent a central element of a micro task: given a certain item, users are asked to estimate which values of user attributes are compatible with the item, i.e., are a criteria for selecting and recommending the item. The evaluation of items with regard to user attributes is the central micro task implemented in the current PEOPLEVIEWS prototype. A detailed evaluation of the example items (Table 3.2) regarding the user attributes *goal*, *runtime*, and *risk* is provided in Table 3.4.

Each row of Table 3.4 specifies a so-called *user-specific filter constraint* (Felfernig et al. (2014a)), i.e., a filter constraint (specified by a user) regarding a specific item. For example, user *Luc* specified

user attribute	question to user	attribute domain
goal (gl)	What are your personal goals?	{Studies, Pension, Speculation, Car, House, World trip, noval}
runtime (rt)	When is the money needed?	{in 1 year, in 2 years, in 3-5 years, in 5-10 years, in 10-20 years, in more than 20 years, noval}
risk (ri)	Preparedness to take risks?	{low, medium, high, noval}

Table 3.3.: User attributes $u \in U$ of example financial services recommender.

Pension and *Speculation* as possible goals that lead to an inclusion of the item *Investment Fund B* into a recommendation. Furthermore, *Luc* believes that a user should have a high preparedness to take risks (attribute *risk*) and should need the payment in 3-5 years, 5-10 years or 10-20 years from now on. Semantically, an item X is selected by a user-specific filter constraint if all the preconditions are fulfilled.

In order to derive *recommendation-relevant filter constraints* (recommendation rules) (Felfernig et al. (2014a)), user-specific filter constraints have to be aggregated. An example of this aggregation step is depicted in Table 3.5. For each item all related user-specific filter constraints are integrated into one constraint. Each row in this table has to be interpreted as a filter constraint for a specific item, for example, the constraint in the first row of Table 3.5 is the following. The item Φ_1 (*Investment Fund A*) is *included* (recommended) if the user requirements regarding goal (gl), runtime (rt), and risk (ri) are consistent with the condition of the recommendation-relevant filter constraint $gl \in \{\textit{Studies}, \textit{Pension}, \textit{Speculation}, \textit{noval}\} \wedge rt \in \{\textit{in 5-10 year}, \textit{in 10-20 years}, \textit{noval}\} \wedge ri \in \{\textit{medium}, \textit{high}, \textit{noval}\} \rightarrow \textit{include}(\Phi_1)$.

Table 3.5 includes the complete set of recommendation-relevant filter constraints (recommendation rules). Exactly these conditions are applied by PEOPLEVIEWS to determine recommendations for a user. In PEOPLEVIEWS, each item has exactly one related recommendation-relevant filter constraint; each such filter constraint is represented by one row in Table 3.5. The general logical representation of a recommendation-relevant filter constraint f for an item Φ is shown in Formula 3.1. In this context, $\textit{values}(\Phi, u)$ is the set of supported domain values of user attribute $u \in U$ (see Table 3.4). The constant *noval* denotes the fact that no value has been selected for the corresponding user attribute.

$$f(\Phi) : \bigwedge_{u \in U} u \in \textit{values}(\Phi, u) \cup \{\textit{noval}\} \rightarrow \textit{include}(\Phi) \quad (3.1)$$

user	item name (id)	goal	runtime	risk
An-drea	Investment Fund A (Φ_1)	Studies, Pension, Speculation	in 5-10 years, in 10-20 years	high
Luc	Investment Fund A (Φ_1)	Pension, Speculation	in 5-10 years, in 10-20 years	high
Mary	Investment Fund A (Φ_1)	Pension, Speculation	in 5-10 years, in 10-20 years	medium, high
Torsten	Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Luc	Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Mary	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years, in 10-20 years	low, medium, high
An-drea	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years	low, medium
Luc	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years	low, medium
Mary	Bond (Φ_4)	Studies, Car, House	in 2 years, in 3-5 years, in 5-10 years	low, medium
An-drea	Savings Book (Φ_5)	Studies, Car, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low
Torsten	Savings Book (Φ_5)	Studies, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low

Table 3.4.: Example of *user-specific filter constraints* (= micro contributions).

item name (id)	goal	runtime	risk
Investment Fund A (Φ_1)	Studies, Pension, Speculation	in 5-10 years, in 10-20 years	medium, high
Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years, in 10-20 years	low, medium, high
Bond (Φ_4)	Studies, Car, House	in 2 years, in 2-5 years, in 5-10 years	low, medium
Savings Book (Φ_5)	Studies, Car, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low

Table 3.5.: Example of *recommendation-relevant filter constraints* which are the result of integrating user-specific filter constraints (see Table 3.4).

For each pair $(\Phi, val \in values(\Phi, u))$, PEOPLEVIEWS determines a corresponding support value (see Formula 3.2). In this context, $occurrence(\Phi, val)$ denotes the number of times, value val occurs in a user-specific filter constraint for item Φ and $occurrence(\Phi)$ denotes the number of times an item Φ is referred in a user-specific filter constraint. For example, $support(\Phi_1, Studies) = \frac{1}{3}$.

$$support(\Phi, val) = \frac{occurrence(\Phi, val)}{occurrence(\Phi)} \quad (3.2)$$

The complete set of support values is depicted in Table 3.6. In PEOPLEVIEWS, an item Φ can have an associated *rating* ($rating(\Phi)$) which represents an item evaluation with regard to quality and related services. Such a rating can be determined, for example, by calculating the average of the individual user item ratings.⁴ For simplicity, we do not take into account user ratings in the utility function discussed below (see Formula 3.3).

Depending on the requirements articulated by the current user (see, e.g., Table 3.7), PEOPLEVIEWS determines and ranks a set of relevant items as follows. First, recommendation-relevant filter constraints are applied to pre-select items that fulfill the user requirements $REQ = \{req_1, req_2, \dots, req_k\}$. In our example, the set $\{Investment Fund A, Building Loan\}$ would be selected by the recommendation-relevant filter constraints (see Table 3.5).

The determined recommendation set must be ranked before being presented to the user. In PEOPLEVIEWS, item ranking is based on the following utility function (see Formula 3.3). The utility of each item is derived from the support values of individual requirements (see Formula 3.2).

$$utility(\Phi, REQ) = \sum_{req \in REQ} support(\Phi, req) \quad (3.3)$$

The item ranking of our working example as a result of applying Formula 3.3 is depicted in Table 3.8. For example, $utility(\Phi_3, REQ = \{goal = Studies, goal = Pension, runtime = in 5-10 years, risk = medium\}) = support(\Phi_3, goal = Studies) + support(\Phi_3, goal = Pension) + support(\Phi_3, runtime = in 5-10 years) + support(\Phi_3, risk = medium) = 1.0 + 1.0 + 1.0 + 1.0 = 4.0$.

3.4. User Interface

3.4.1. PEOPLEVIEWS

In this chapter we discuss the PEOPLEVIEWS user interface⁵ and also show how PEOPLEVIEWS recommendation knowledge can be exploited by the STUDYBATTLE learning environment. The PEOPLEVIEWS homescreen is depicted in Figure 3.1. For applying PEOPLEVIEWS recommenders, there

⁴Similar to ratings provided by platforms such as *amazon.com*.

⁵The user interface is currently only available in German.

item name (id)	attribute:value	support value
Investment Fund A (Φ_1)	goal: Studies	0.33
	goal: Pension, Speculation	1.0
	runtime: in 5-10 years, in 10-20 years	1.0
	risk: medium	0.33
	risk: high	1.0
Investment Fund B (Φ_2)	goal: Pension, Speculation	1.0
	runtime: in 3-5 years, in 5-10 years, in 10-20 years	1.0
	risk:high	1.0
Building Loan (Φ_3)	goal: Studies, Pension, Car, House	1.0
	runtime:in 5-10 years	1.0
	runtime:in 10-20 years	0.33
	risk:low, medium	1.0
	risk:high	0.33
Bond (Φ_4)	goal: Studies, Car, House	1.0
	runtime:in 2 years, in 3-5 years, in 5-10 years	1.0
	risk:low, medium	1.0
Savings Book (Φ_5)	goal: Studies, House, World trip	1.0
	goal:Car	0.5
	runtime:in 1 year, in 2 years, in 3-5 years, in 5-10 years	1.0
	risk:low	1.0

Table 3.6.: Support values (see Formula 3.2) derived from user-specific filter constraints (see Table 3.4).

id	requirement
req_1	$goal = \text{Studies}$
req_2	$goal = \text{Pension}$
req_3	$runtime = \text{in 5-10 years}$
req_4	$risk = \text{medium}$

Table 3.7.: Example set of user requirements ($req_i \in REQ$).

item name (id)	utility	rank
Building Loan (Φ_3)	4.0	1
Investment Fund A (Φ_1)	2.66	2

Table 3.8.: Utility-based ranking of items in the recommendation set.

is no explicit need for being logged in. Recommenders can be selected and activated directly from the homescreen (see the tag cloud in Figure 3.1).

If users are logged in, they are allowed to contribute to the development of PEOPLEVIEWS recommender applications. Only the creators of a recommender application are allowed to define user attributes. Other users can complete micro tasks in terms of evaluating items with regard to a defined set of user attributes. The list of user attributes used in our working example is depicted in Figure 3.2 (corresponds to the entries of Table 3.3).

Logged-in users are also allowed to enter new items to the recommender product catalog. The PEOPLEVIEWS representation of product catalogs is exemplified in Figure 3.3 (corresponds to the list of items shown in Table 3.2).

The interface for evaluating an item with regard to a set of user attributes is depicted in Figure 3.4. The screenshot depicts the evaluation of *Building Loan* with regard to the user attribute *goal*. After having completed the definition of a PEOPLEVIEWS recommender, the recommender can directly be executed. The user interface of our financial services recommender is depicted in Figure 3.5.

3.4.2. STUDYBATTLE

Recommendation-relevant filter constraints can be further exploited for generating different learning applications that are part of the STUDYBATTLE environment. STUDYBATTLE is a game-based learning environment which can be utilized as an environment for learning product knowledge and sales practices. Examples of STUDYBATTLE games are the following.

Assign Properties. Figure 3.6 depicts an example user interface of a STUDYBATTLE application that implements a quiz related to knowledge about the relationship between user attributes and items. In the example, users have the task to assign items on the left hand side to user attribute values on the right hand side where each product has to be assigned to at least one attribute value and vice-versa.

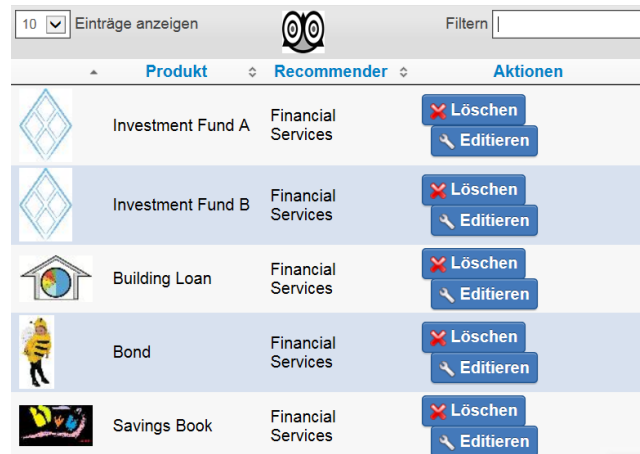


Figure 3.1.: PEOPLEVIEWS homescreen – the current version of the user interface is provided in German. The homescreen explains the basic functionalities of the system (development, maintenance, and execution of recommender applications).

Namen	Fragen	Aktionen	
goal	What are your personal goals?	Löschen	Editieren
runtime	When is the money needed?	Löschen	Editieren
risk	Preparedness to take risks?	Löschen	Editieren

Hinzufügen

Figure 3.2.: PEOPLEVIEWS: example user attributes.








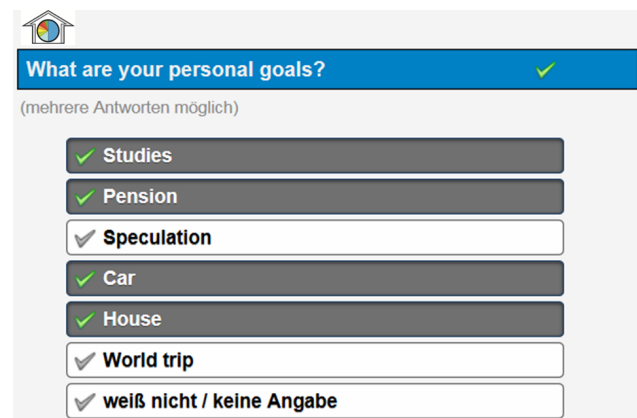

Produkt	Recommender	Aktionen
 Investment Fund A	Financial Services	<input type="button" value="Löschen"/> <input type="button" value="Editieren"/>
 Investment Fund B	Financial Services	<input type="button" value="Löschen"/> <input type="button" value="Editieren"/>
 Building Loan	Financial Services	<input type="button" value="Löschen"/> <input type="button" value="Editieren"/>
 Bond	Financial Services	<input type="button" value="Löschen"/> <input type="button" value="Editieren"/>
 Savings Book	Financial Services	<input type="button" value="Löschen"/> <input type="button" value="Editieren"/>

Figure 3.3.: PEOPLEVIEWS: example of an item list.



 What are your personal goals?

(mehrere Antworten möglich)

- Studies
- Pension
- Speculation
- Car
- House
- World trip
- weiß nicht / keine Angabe

Figure 3.4.: PEOPLEVIEWS: example of an item evaluation user interface (evaluation of item *Building Loan* with regard to the user attribute *goal*).

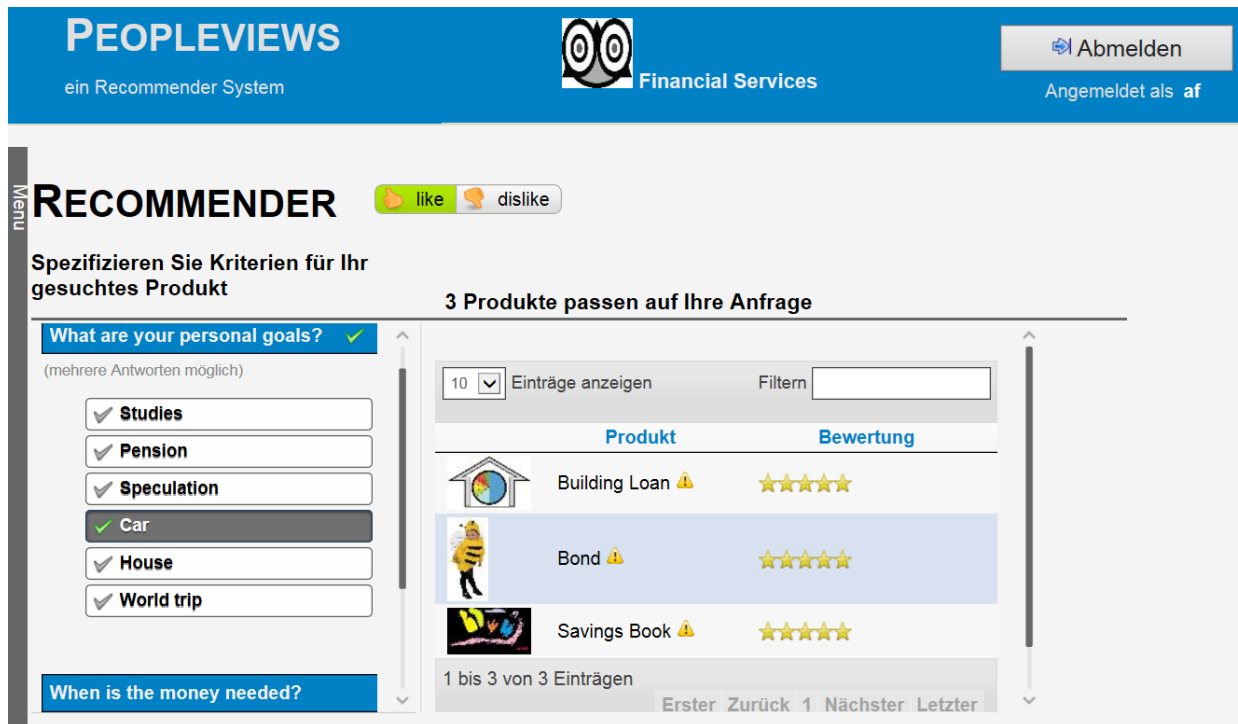


Figure 3.5.: PEOPLEVIEWS: example of a recommender application (Financial Services).

Find Items. A different version of the game depicted in Figure 3.6 is to ask for products that fulfill certain criteria (represented by a combination of user attribute settings).

Find Incompatibilities. This game focuses on combinations of user attribute values that do not lead to a solution, i.e., users have to specify combinations of user attribute values from which they think that no corresponding solution could be found.

Maximize Requirements. The task is to identify minimal sets of requirements (from a given set of requirements REQ) that have to be deleted from REQ such that the remaining requirements lead to at least one solution. This game type reflects the principles of model-based diagnosis (de Kleer et al. (1992); Reiter (1987)), i.e., support users in learning and improving repair behavior in situations where no solution can be identified.

Maximize Items. A similar task is focused on the repair of item sets; in this context the task of users is to identify a maximal set of items from a given set of items such that there exists at least one combination of user attribute values that lead to these items (not necessarily exclusively). An additional criteria could be that at least n items from the original item list must remain in the result set.

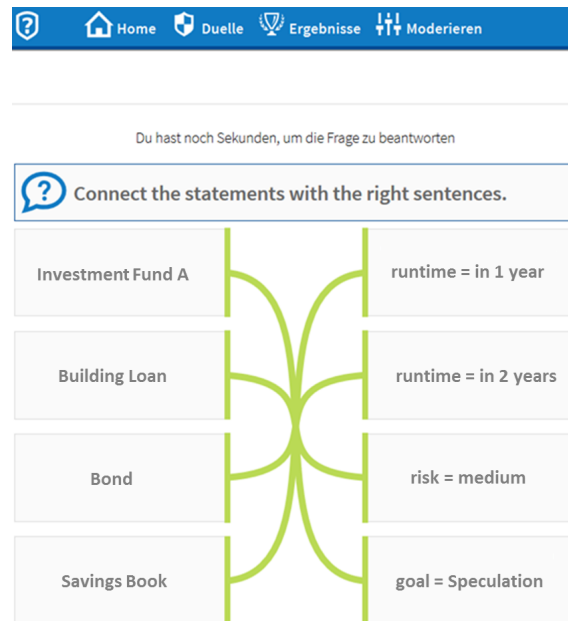


Figure 3.6.: STUDYBATTLE "Assign Properties" learning application. The task of the user is to relate items with corresponding attribute values.

3.5. Preliminary Evaluation Results

Human Computation based Knowledge Acquisition. Applying Human Computation concepts (von Ahn (2005)) in the context of recommender application development and maintenance has the potential to lift the burden of enormous engineering and maintenance efforts from the shoulder of knowledge engineers. Micro tasks as sketched in this chapter can be structured in a way that they are understandable for domain experts without a computer science background. Knowledge gained from completed micro tasks can be easily integrated into a corresponding recommender knowledge base. Due to the increasing size and complexity of knowledge bases, the development of such technologies is crucial since they help to tackle scalability issues which otherwise could cause a complete failure with regard to a company-wide recommender deployment. As such, PEOPLEVIEWS technologies can be considered as a first step towards more scalable development methods that will also help to further increase the popularity of knowledge-based (recommendation) technologies.

Usability. An initial user study has been conducted with an early version of PEOPLEVIEWS at the Graz University of Technology (Felfernig et al. (2014a)). N=161 (15% female and 85% male) students interacted with the system with the goal to develop different recommender applications. After having completed the development, the study participants had to complete a questionnaire which was based on the system usability scale (SUS) (Bangor et al. (2008)). Evaluation results regarding the SUS aspects are summarized in Figure 3.7. Besides usability questions, further feedback has been provided by the study participants, for example, the majority of the participants (69% of all study participants)

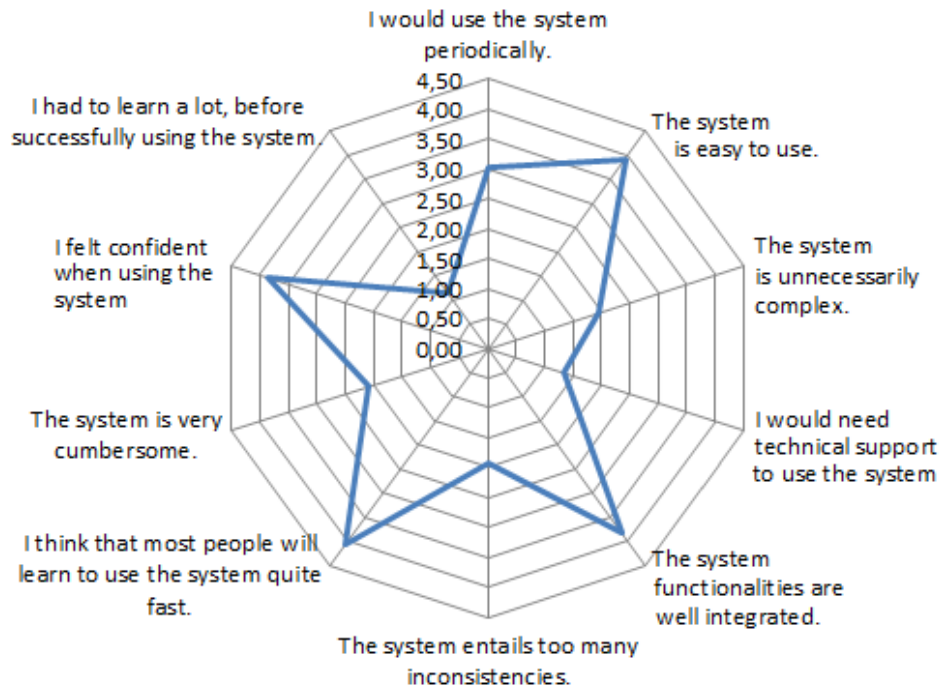


Figure 3.7.: Results of a SUS-based usability study Bangor et al. (2008) of the PEOPLEVIEWS environment.

would like to further contribute to PEOPLEVIEWS recommenders. 56% out of those participants who wanted to contribute agreed to contribute within a time frame of less than 30 minutes per week.

3.6. Future Work

The major goal of this chapter was to provide an overview of the PEOPLEVIEWS recommendation environment. There are many issues for future work that we want to tackle and integrate corresponding solutions in upcoming PEOPLEVIEWS versions.

Weighting of Item Evaluations. In the current PEOPLEVIEWS version it is possible to assign user attribute values to items, i.e., to specify which criteria are relevant for the selection of a certain item. In future versions of PEOPLEVIEWS it will be possible to integrate weights into item evaluations. This maybe does not play a major role in financial service related recommender applications but can be important in other domains were nuances and personal tastes play a more important role. For example, in the context of recommending digital cameras, it can be important to specify degrees regarding certain camera properties, for example, the degree to which a camera is able to support sports photography.

Further Micro Tasks. In the current system version, the only micro task to be completed is to define the relationship (compatibility properties) between items and corresponding user attribute values. In

name	description
item quality check	check whether a certain item belongs to a specific recommender (is an existing recommender-related item)
attribute quality check	check whether a certain attribute belongs to a specific recommender (user attribute or item attribute exists in the item domain)
attribute value quality check	check whether a certain value belongs to the domain of an attribute (user attribute or item attribute)
graphic check	check whether a certain figure belongs to a certain item
evaluate item	assign user attribute values to items
attribute value utility check	derive a ranking that shows which items best support a user attribute value

Table 3.9.: Example list of micro tasks to be integrated in PEOPLEVIEWS.

future versions of PEOPLEVIEWS we will extend this list of micro tasks (see Table 3.9).

User Selection for Micro Tasks. An important enhancement will be the inclusion of methods that automatically select users for a given set of micro tasks and also take into account fairness in the distribution of micro tasks. As detected in our initial studies, users are willing to contribute to the further development of PEOPLEVIEWS recommenders. An important issue in this context is to find the users with the right expertise for certain tasks and also to not overload users. Our approach in this context will be to maintain user profiles which are derived from observing the activities of a user within PEOPLEVIEWS. For example, if a user selects a certain item when interacting with the financial services recommender, the keywords extracted from the corresponding item description are stored in the user profile. If (in the future) micro tasks related to similar items (items with a similar description) have to be completed, users with expertise regarding such items will be the preferred contact persons.

Games. Games will be another mechanism for data collection in the PEOPLEVIEWS modeling mode. A single user game will be included that is quiz-based. The overall goal is to guess user attribute settings correctly that best describe a certain item. In a second game two users will jointly try to figure out user attribute values that best describe shown items. The more matching item evaluations exist the better the team performs.

Dependencies between User Attributes and Item Attributes. An extension of the current PEOPLE-

VIEWES version will be the possibility to identify direct relationships between user attribute values and technical product properties. This is not the case in the current PEOPLEVIEWES version since dependencies are only defined between user attribute values and items.

Recommendation Algorithms. The current version of PEOPLEVIEWES relies on the discussed recommendation-relevant filter constraints – item ranking is based on a utility-based evaluation (see Formula 3.3). In future versions of PEOPLEVIEWES we will extend the quality of recommendation algorithms by, for example, adapting the determination of support values. If, for example, additional information about the performance of a certain user is available (e.g., performance with regard to correctly completed micro tasks in the past), this information can be used to increase/decrease the weight of a user when determining support values. Finally, when users are specifying their requirements, future versions of PEOPLEVIEWES will allow the specification of preferences (weights) which indicate user preferences regarding certain requirements. This will also include approaches to the learning of weights (users should not have to specify all weights explicitly).

Inconsistency Management. Given a set of customer requirements it could be the case that no solution can be presented to the user. In upcoming versions of PEOPLEVIEWES we will focus on integrating state-of-the-art diagnosis algorithms that help to automatically determine repair actions in such inconsistent situations (Felfernig et al. (2011)). These repairs will take into account user weights (preferences) and thus minimize the number of interaction cycles needed to find a reasonable solution. In addition to this more intelligent management of inconsistent requirements, we will integrate mechanisms that help to consolidate the set of user-specific filter constraints in order to make the resulting recommendation-relevant filter constraints more compact. Consolidation will be achieved, for example, on the basis of redundancy detection algorithms (Felfernig et al. (2011)).

Quality Management. The major task of quality management is to assure the quality of the dataset collected on the basis of different micro tasks. Quality assurance must be capable of detecting and preventing manipulations of the dataset (also under the assumption that anonymous users are allowed to complete micro tasks), it must also identify changes to the given set of user-specific filter constraints that help to improve the prediction quality of recommendation algorithms. Quality assurance is also responsible for the generation of micro tasks that need to be completed in order to improve the overall quality of the PEOPLEVIEWES datasets. The micro tasks generated by quality assurance are summarized as an *agenda* – this agenda is forwarded to micro task scheduling that is responsible for distributing micro tasks to the PEOPLEVIEWES user community.

3.7. Conclusions

In this chapter we gave an overview of the PEOPLEVIEWES recommendation environment which exploits concepts of Human Computation to integrate domain experts more deeply into knowledge base development and maintenance processes. PEOPLEVIEWES knowledge bases can be exploited to gen-

erate learning applications which can be used in the STUDYBATTLE environment. A major focus of this chapter was to show how PEOPLEVIEWS can be applied in the context of financial service recommendation. The concepts presented in this chapter have the potential to avoid scalability issues which already exist in many knowledge-based environments due to the increasing size and complexity of knowledge bases.

Towards Open Configuration

This chapter is based on the results documented in Felfernig et al. (2014d). Major parts in terms of writing and literature research have been provided by the author of this thesis.

4.1. Abstract

Configuration technologies are typically applied in *closed* settings where one (or a small group of) knowledge engineer(s) is in charge of knowledge base development and maintenance. In such settings it is also assumed that only single users configure the corresponding products and services. Nowadays, a couple of scenarios exist that require more *openness*: it should be possible to cooperatively develop knowledge bases and to jointly configure products and services, even by adding new features or constraints in a flexible fashion. We denote this integration of groups of users into configuration-related tasks as *open configuration*. In this chapter we introduce features of open configuration environments and potential approaches to implement these features.

4.2. Introduction

Configuration (Felfernig et al. (2014b); Hvam et al. (2007); Stumptner (1997)) is one of the most successful technologies of Artificial Intelligence (AI). It is applied in many domains such as telecommunication (Fleischanderl et al. (1998)), furniture (Haag (1998)), and financial services (Felfernig et al. (2007b)). Most configuration-related functionalities are assuming closed settings where knowledge bases are developed by a single (or a small group of) knowledge engineer(s) and the corresponding configurators are applied by single users. Implementing configurator applications this way entails drawbacks which become manifest in terms of *scalability problems* in knowledge engineering

(Richardson and Domingos (2003)) and *suboptimal decisions* if a single user decides for the whole group (Felfernig et al. (2012b)).

Scalability Problems. The transformation of domain knowledge into a configuration knowledge base is an effortful process often characterized by a knowledge acquisition bottleneck (Hayes-Roth et al. (1983)) that is considered as a major obstacle for a sustainable application of knowledge-based technologies (Hoppenbrouwers et al. (2009); Wagner (2006)). To tackle this bottleneck, efficient approaches have been developed that support graphical knowledge engineering (Felfernig et al. (2000a); Hotz et al. (2013)) and intelligent debugging (Felfernig et al. (2004, 2012a); Schubert et al. (2010)).

These approaches help to improve the efficiency of knowledge engineering but still do not solve the problem of *missing scalability*: the increasing amount and complexity of configuration knowledge bases exceeds the resources available for performing the corresponding development and maintenance operations (Huang et al. (2008); Richardson and Domingos (2003)). In order to assure scalability, future configuration technologies have to support a deeper integration of a wider group of users (e.g., product developers, marketing experts, sales representatives, and knowledge engineers) into knowledge engineering. Related solutions should go beyond state-of-the-art approaches that are focusing on experienced knowledge engineers and programmers (Hvam et al. (2007)) by allowing the completion of knowledge engineering tasks by the mentioned groups. We denote this approach as *community-based knowledge engineering*.

Suboptimal Decisions. A basic assumption of existing configuration systems is that products and services are typically configured by single users. However, many scenarios exist where not a single user but a group of users is in charge of configuring a product (see Section 4.4). Existing configuration environments do not take into account such scenarios which often leads to situations where a single user has to "encode" the requirements and preferences of a whole group. This can lead to suboptimal configurations (decisions) that do not reflect the group preferences in an optimal fashion. Future configuration technologies should take into account the fact that groups of users can be engaged in configuration processes and provide group decision mechanisms that help the group to jointly configure a product in a consensual fashion. We denote this type of configuration as *group-based configuration*. Especially in scenarios where multiple stakeholders define and configure products, enhanced flexibility is required: configurator users may request to add or refine product features and constraints which can be seen, for example, in open innovation (Chesbrough (2003)) or postponement scenarios (Forza et al. (2008); Yang and Burns (2003)). We subsume such activities under the term *flexible product enhancement*.

The concepts of *community-based knowledge engineering*, *group-based configuration*, and *flexible product enhancement* can be summed up under the notion of *open configuration*. In this chapter we sketch functionalities which have to be provided by open configuration environments. In Section 4.3 we introduce features and potential technological solutions to tackle the issue of scalability in knowledge engineering scenarios. In Section 4.4 we discuss features of group-based configuration. In

micro task topic	description
variables	definition/evaluation of variables included in V
questions	definition/evaluation of questions related to $v_i \in V$
dialog sequences	definition/evaluation of question sequences
constraints	definition/evaluation of constraints in C
examples	definition/evaluation of test cases in T
diagnoses	evaluation of conflict resolution alternatives for C

Table 4.1.: Community-based knowledge engineering: example micro tasks.

Section 4.5 we discuss aspects of product enhancement in open configuration. With Section 4.6 we provide a discussion of related work. We conclude the chapter with Section 4.7.

4.3. Community-based Knowledge Engineering

In the following we will discuss aspects that become relevant if we want to integrate a larger group of users into configuration knowledge engineering. For the sake of simplicity and without loss of generality we assume that a configuration knowledge base is represented in terms of a constraint satisfaction problem (CSP) (Mackworth (1977)) consisting of a set of variables $V = \{v_1, \dots, v_n\}$ with corresponding domain definitions ($\text{dom}(v_i)$), and a set of constraints $C = \{c_1, \dots, c_m\}$. We base our discussions on the following simplified financial services configuration knowledge base.

- $V = \{\text{willingness to take risks (wr), expected return rate (rr), investment period (ip)}\}$
- $\text{dom}(wr) = \{\text{low, medium, high}\}$, $\text{dom}(rr) = \{<6\%, 6-9\%, >9\%\}$, $\text{dom}(ip) = \{\text{shortterm, mediumterm, longterm}\}$
- $C = \{c_1 : wr = \text{medium} \rightarrow ip \neq \text{shortterm},$
 $c_2 : wr = \text{high} \rightarrow ip = \text{longterm},$
 $c_3 : ip = \text{longterm} \rightarrow rr = <6\% \vee rr = 6-9\%,$
 $c_4 : rr = >9\% \rightarrow wr = \text{high},$
 $c_5 : rr = 6-9\% \rightarrow wr \neq \text{low} \wedge wr \neq \text{medium}\}$

In cases where one or a small group of knowledge engineers is in charge of developing and maintaining a configuration knowledge base, attributes (component types), domains, and related constraints are typically formalized on the basis of examples and textual descriptions provided by domain experts (Hvam et al. (2007)). If the product domain knowledge has to be adapted, the whole process is restarted, i.e., domain experts articulate the change requests in an informal fashion and knowledge engineers implement the needed adaptations.

The correctness of changes performed on a knowledge base can be evaluated, for example, on the basis of regression tests where positive and negative test cases are used to figure out whether the

knowledge base shows the intended behavior (Felfernig et al. (2004)). Positive test cases (examples) are a specification of an intended behavior of the knowledge base and negative test cases exemplify unintended behavior. Existing approaches to configuration knowledge base testing and debugging exploit positive test cases to detect errors/deficiencies by inducing conflicts in the incorrect configuration knowledge base. Such conflicts are minimal sets of constraints that are responsible for the faulty behavior of the knowledge base and therefore have to be adapted by knowledge engineers.

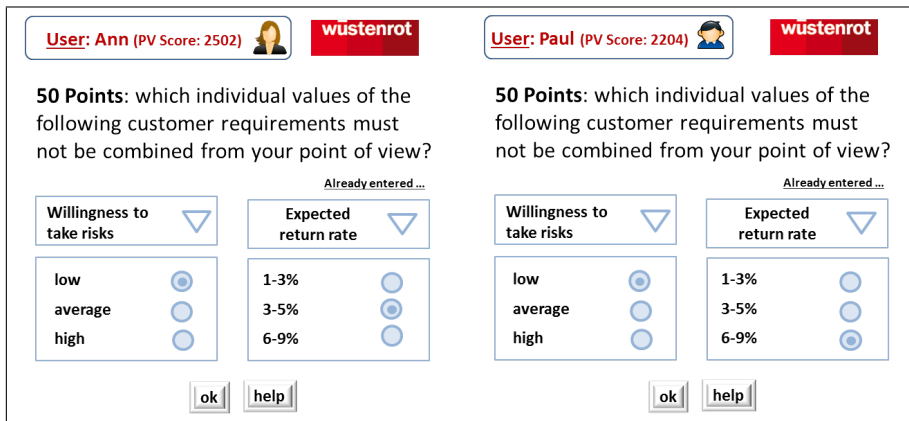


Figure 4.1.: Sketch of a user interface for game-based knowledge acquisition. The overall goal of the game is that both players agree on the set of incompatible value combinations of a given set of variables. This user interface can be regarded as a micro task template for the acquisition of incompatibility constraints.

Community-based Knowledge Engineering. Intelligent testing and debugging (Felfernig et al. (2004)) is an important contribution to the improvement of knowledge engineering processes. However, the growing size and complexity of configuration knowledge bases often makes it hard for individual knowledge engineers to keep track of new developments and adaptations. As a consequence, more time is needed to provide a new production version of the configuration knowledge base and the probability of including erroneous constraints increases. In order to assure scalability, it is important to integrate end-users more deeply into knowledge base development and maintenance and thus to exploit unemployed knowledge engineering potentials.

In the following we discuss issues that have to be taken into account when integrating groups into *community-based knowledge engineering* processes. An in-depth integration of a larger group of users allows knowledge engineers to delegate basic engineering tasks (so-called *micro tasks*). Table 4.1 provides an overview of micro task topics. For each topic a couple of different concrete micro tasks can be defined, for example, a variable can be defined but also evaluated with regard to the appropriateness of its domain definition.

In order to figure out variables (component types) relevant for the configuration knowledge base, users should be allowed to enter proposals for variables and component types (including the corresponding domain definitions) on their own. Variables are often associated with questions posed to the

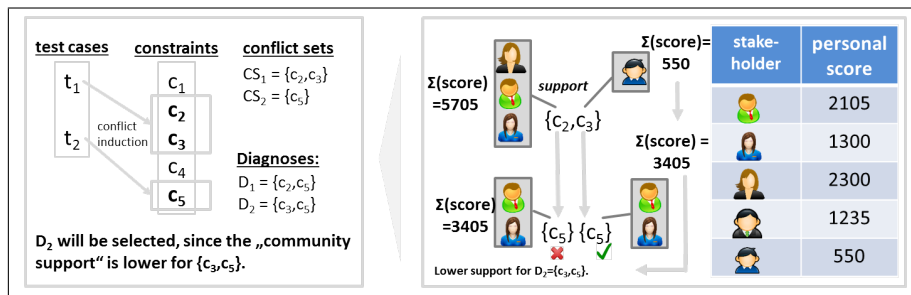


Figure 4.2.: Group-based diagnosis of a faulty configuration knowledge base. Diagnoses are selected by taking into account the expertise of users/knowledge engineers: the higher the *personal score* (value derived from his/her personal contributions), the higher the weight given to his/her opinion.

user of a configurator application – alternative formulations of such questions and also the sequences in which these questions are posed should be defined and evaluated by users. In addition to structural properties typically defined in terms of variables or component types and their relationships, constraints define additional restrictions on possible combinations of variable values (components).

Especially in community-based scenarios, where a larger number of users interacts with the knowledge engineering environment, engineering practices will change in the sense that users are providing knowledge chunks in a collaborative fashion and the knowledge engineering environment is in charge of aggregating this information. In this context, it is necessary to have mechanisms that automatically distribute knowledge acquisition tasks among users in a systematic fashion (e.g., depending on the workload, knowledge level, and preferences of users). Such tasks can be represented in a more-or-less traditional form of todo-lists but can also be represented in terms of so-called *games with a purpose* (von Ahn (2006)) which is an upcoming trend also in the knowledge engineering field (Siorpaes and Hepp (2008)).

A simple example of such a knowledge acquisition interface is depicted in Figure 4.1. In this example game, the users *Ann* and *Paul* have the task to cooperatively figure out combinations of customer requirements that are incompatible, i.e., induce an inconsistency with the knowledge base. The players have successfully completed their task if they, for example, selected the same set of assignments as candidates for incompatibilities. The underlying assumption of this game is that *Ann* does not know the input of *Paul* and vice-versa.

Further examples of gamification-based interfaces for configuration knowledge acquisition are: cooperative definition of relevant variables (including their domains), the estimation of intuitive dialog sequences (which questions should be asked in which order), the derivation of further constraint types (e.g., filter constraints that match user requirements to corresponding technical product properties), and the estimation of accepted repair rankings in situations where no solution could be found. Such scenarios can be supported by input templates that represent micro-tasks (see Figure 4.1).

Testing and Debugging. The definition and evaluation of (positive and negative) test cases is a crucial issue since the correctness of a test suite directly influences the correctness of the results determined by a configurator. In Felfernig et al. (2004) positive and negative examples are exploited for debugging knowledge bases on the basis of the concepts of model-based diagnosis (Reiter (1987)). In this context, positive examples are exploited for inducing conflicts in a configuration knowledge base. A negative example is assumed to be integrated in negated form into the knowledge base in the case that it has not been rejected by the knowledge base. On the basis of the following two test cases (examples) we can show how positive examples are used to find errors in the knowledge base. Both test cases are in conflict with constraints in the configuration knowledge base introduced in Section 4.3.

- $t_1 : wr = high \wedge rr = >9\%$
- $t_2 : rr = 6-9\% \wedge wr = medium$

A conflict between a test case t and a set of constraints in the configuration knowledge base can be defined as a *conflict set* $CS \subseteq C$: $CS \cup t$ inconsistent. Such a conflict set CS is *minimal* if there does not exist another conflict set CS' with $CS' \subset CS$. To resolve a minimal conflict, only one element has to be deleted from CS . In our example, the test case t_1 is in conflict with the constraints c_2 and c_3 and test case t_2 is in conflict with the constraint c_5 . Consequently we have two different (and minimal) conflict sets which are $CS_1: \{c_2, c_3\}$ and $CS_2: \{c_5\}$. Resolving these conflicts results in two different diagnoses, namely $D_1 = \{c_2, c_5\}$ and $D_2 = \{c_3, c_5\}$, i.e., a diagnosis is a hitting set (Reiter (1987)) which includes at least one constraint from each of the given conflict sets.

Typically, there are many alternative diagnoses and the question has to be answered which of these is acceptable for the users engaged in testing and debugging. Figure 4.2 depicts a basic approach of integrating knowledge about the users expertise in the determination of a diagnosis. For the conflict $CS_1 = \{c_2, c_3\}$, the majority of users prefers to keep c_2 as-is and to delete or change c_3 to resolve the conflict. Since CS_2 is a singleton, no alternatives exist for resolving the conflict, i.e., c_5 must be selected. Overall, the elements in the diagnosis $D_2 = \{c_3, c_5\}$ have a lower community support and therefore will be changed or deleted by the users in order to restore the consistency with the test-suite $\{t_1, t_2\}$.

4.4. Group-based Configuration

An assumption of existing configuration environments is that there is no need for additional configuration support in scenarios where groups of users are jointly configuring their preferred product or service. A major consequence of this assumption is that single users are forced to encode the preferences of a group which is often done in a suboptimal fashion.

Within the scope of an industry study with representatives of N=25 companies applying configurators we figured out that none of the existing configuration environments provides technologies that

ID	domain for group-based configuration	components and constraints	decision makers
1	software release plans	requirements, releases, dependencies, preferences	stakeholders in software project
2	product line scoping and open innovation	(new) features, constraints between features, preferences	representatives from different departments, customers
3	bundle configuration (e.g., hotel, flight, tour, etc.)	(new) destinations, hotels, sightseeing tours, (resource) constraints, preferences	travel group
4	stakeholder selection for a new software project	(new) persons, constraints regarding competences and resources, preferences	(initial) team members
5	architectural design in software development	components, interfaces, technologies, constraints between components, preferences	(distributed) software project members
6	financial service configuration	financial services, resource constraints, preferences	family members
7	building configuration (e.g. smart home, office block)	rooms, furniture, light control equipment, constraints between components, preferences	family members, suppliers, company representatives
8	funding decisions	project proposals, resource constraints, preferences	evaluators, consultants, decision makers

Table 4.2.: Application scenarios for *group-based configuration* identified within the scope of a study with N=25 companies applying configuration systems.

support groups of users in jointly configuring a solution. However, there is a strong agreement on the fact that such technologies have to be included in future configurators. The study participants reported different scenarios for the application of group-based (socially aware) configuration technologies. Social awareness in this context denotes the fact that specific properties of group decision processes are explicitly taken into account by the configuration environment (e.g., the need to achieve consensus among group members). Examples of such scenarios are depicted in Table 4.2.

In these scenarios a group of users is in charge of *jointly configuring a product or service*, for example, when configuring a *holiday trip* (bundle configuration) for a group of friends (Jameson (2004)), the requirements and preferences of all group members should be taken into account. When configuring a *software release plan*, the preferences of individual stakeholders regarding the assignment of requirements to releases have to be taken into account (Ninaus et al. (2014)).

Taking into account requirements and preferences of group members requires decisions regarding *trade-offs*. In the context of holiday trips such a trade-off could be the acceptance of a lower-quality

destination	Lindwurm	Großglockner	Pyramidenkogel	Isonzo Valley
Ben	1	1	0	0
John	1	1	0	0
Kate	0	0	1	1
least misery	1	0	1	0
majority voting	1	1	0	0

Table 4.3.: Example set of tourist destinations (in the Alps-Adriatic area). The assumption in this example is that each person is allowed to articulate at most two preferences and the trip must include at least two destinations.

hotel which is much nearer to the sightseeing destination preferred by a specific user. When configuring software release plans, a trade-off could concern the postponement of a specific requirement to a later release while increasing the importance level of this requirement (to avoid further postponements).

The determination of trade-offs must be based on preference aggregation mechanisms (Masthoff (2011)) that take into account the preferences of all group members as far as possible. For example, the *least misery* strategy avoids massive discriminations of individual group members by minimizing the maximum number of trade-offs to be accepted by an individual. In contrast, *majority voting* follows the opinions of the majority of the group members which can lead to discriminations against individuals.

An example of the application of the least misery strategy in the context of deciding about a common sightseeing trip is depicted in Table 4.3. In this simplified example, each person is allowed to select at most two destinations and the corresponding trip must include two destinations. Since Ben and John have similar preferences, majority voting would discriminate Kate. In contrast, least misery tries to find a trade-off that has the potential to create group consensus. For a detailed discussion of preference aggregation mechanisms we refer the reader to Masthoff (2011).

A major issue for future research is the consideration of longer time periods. For example, if a group of friends jointly configures a holiday trip every year, the aggregation mechanisms used by the group-based configuration environment should take into account (as far as possible) the degree to which individuals had to accept trade-offs in the past and use this information for the recommendation of fair trade-offs in future configuration sessions.

On the technical level the above mentioned properties require basic research in the following areas.

First, constraint-based search methods have to be extended with mechanisms that help to predict (partial) configurations which are of relevance for the group. This requires learning methods for search heuristics (Schrijvers et al. (2013)) that help to predict relevant configurations in an efficient fashion. Furthermore, it is important that configurators are able to determine similar and diverse configurations efficiently which could also be achieved on the basis of the mentioned heuristics.

Second, the determination of trade-offs for inconsistent requirements and preferences has to be based on efficient diagnosis methods integrated with intelligent preference aggregation mechanisms (Masthoff (2011)) that can help to better predict trade-offs acceptable for all group members. These aggregations must take into account the histories stored in interaction logs in order to guarantee decision fairness in the long run.

Third, negotiation and argumentation mechanisms have to be developed which support individuals to express acceptable trade-offs. In our holiday configuration scenario an example of such a statement is "I accept to visit Greece this year if we agree to organize a trip to Italy next year". Such arguments cannot be expressed on the basis of existing preference representations.

4.5. Flexible Product Enhancement

The ability to include additional variables (component types), values (components), and constraints in a flexible fashion is important for the implementation of open configuration.

Product line scoping (John et al. (2006)) (in the context of software product line engineering) is in the need of such a flexibility since the features and constraints element of the product line are not completely predefined at the beginning of the engineering process. A larger group of users has to jointly decide which components (features) and constraints should be part of the product line. Thus, product line scoping can be interpreted as open configuration where new alternatives and constraints (and preferences) can be integrated within the scope of the configuration (product line scoping) process.

Open innovation (Chesbrough (2003)) reflects the idea of integrating customer communities into new product development processes of a company. In this context, variability modeling for product lines also requires the support of an easy integration of new component types, components, and constraints which reflect features to be supported by future products. In both scenarios, the integration of new items has to be supported by corresponding group decision processes (see Section 4.4), for example, before a new feature is integrated into the model, the group has to perform the needed validation steps and decide about the inclusion of the feature. This also holds for the afore mentioned scenarios of release planning and holiday trip configuration.

A further example of the need for flexible enhancements are *postponement strategies* (Forza et al. (2008); Yang and Burns (2003)). An example is the automotive industry, where basic car configurations are delivered to dealers who can then integrate additional components such as MP3 players and tow-bars, i.e., are enabled to integrate their own products and services into the basic configuration delivered by car producers. Conform to the definition given in Forza et al. (2008), the mentioned scenario is of *type-III* where customers are allowed to specify additional equipment when they already have a more precise idea of the interior of the car. The corresponding configuration model has to provide flexible interfaces that allow an easy integration of new component types, components, and constraints. A knowledge representation concept that can be exploited in this context are *contextual*

models (Felfernig et al. (2000b)) which allow a systematic extension of existing base diagrams with additional items relevant in a specific context (e.g., the car dealer context). In such scenarios, developers of configurator solutions also have to take into account that – depending on the additional items introduced – search heuristics (Schrijvers et al. (2013)) have to be adapted in order to assure efficient search.

4.6. Related and Future Work

Intelligent testing and debugging methods for configuration knowledge bases have been introduced in Felfernig et al. (2004) where positive test cases can detect errors by inducing conflicts in a configuration knowledge base. Conflicts are then resolved on the basis of model-based diagnosis (Reiter (1987)). In open configuration scenarios, testing and debugging approaches have to be adapted to group-based settings where diagnosis discrimination has to take into account group preferences.

Bessiere et al. (Bessiere et al. (2007)) introduced basic mechanisms to the learning of constraint sets. In this context, knowledge bases are learned on the basis of positive and negative examples. Generated examples are presented to users who have to decide whether the examples are positive or negative. Learning is based on a so-called *bias* that is a knowledge base generated from a vocabulary (variables, domains, and operators). The bias is systematically reduced on the basis of the information included in the examples, for instance, all conflicts induced in the bias by a positive example have to be resolved. In the case of a negative example, at least one conflict must be preserved which guarantees the rejection of the negative example. Approaches to the application of association rule mining for configuration knowledge discovery are discussed in Huang et al. (2008). An important research issue in this context is to assure the understandability and manageability of the derived configuration knowledge (Felfernig et al. (2013c)).

Human Computation is based on the idea of passing those tasks to humans which are easy to solve for them but are not solvable by computers (von Ahn (2005)). Related research has already been conducted in the areas of ontology construction (concept learning) (Siorpaes and Hepp (2008)) and sentiment analysis in text documents (Musat et al. (2012)). A major idea of the work presented in this chapter is to exploit the concepts of Human Computation as a central mechanism for configuration knowledge base construction and maintenance. These mechanisms go beyond concept learning (Siorpaes and Hepp (2008)) and include tasks such as diagnosis discrimination, test case classification and evaluation, and configuration dialog design.

Preferences are not known beforehand but are constructed within the scope of a decision process (Bettman et al. (1998); Teppan and Felfernig (2009)). As a result, biases occur which often lead to suboptimal decisions. Concepts to deal with (group) decision problems in recommender systems are discussed in Felfernig et al. (2010, 2006); Jameson (2004); Mandl et al. (2010); Ninaus et al. (2014). A major issue for future research in this context is an in-depth investigation of decision biases in group

decision making. An important question is to which extent biases are compensated or become more intense when groups decide.

4.7. Conclusions

In this chapter we introduced central ideas and research questions related to open configuration. Openness in this context is related to the idea of a closer integration of end-users into configuration knowledge base development and maintenance operations and of supporting decision processes in scenarios where groups of users are in charge of configuring a product or service. Furthermore, open configuration is often characterized by the need of being able to integrate new items (e.g., component types, components, and constraints) "on the fly". On the basis of the results of a first industry study we reported example application domains and discussed related research challenges. The concepts presented in this chapter can be applied in a broad range of scenarios which go beyond *open configuration*. Further example application domains are (constraint-based) scheduling (Baptiste et al. (2001)), recommender systems (Jannach et al. (2010)), and utility evaluation where user groups are in charge of evaluating alternatives (Felfernig et al. (2013d)).

Configuring Decision Tasks

This chapter is based on the results documented in Stettinger et al. (2014). The author of this thesis wrote most parts of this chapter.

5.1. Abstract

In most cases, decision tasks are individual and different decision tasks require different combinations of features. Features can be, for instance, special preference visibilities during the decision process or specific heuristics that support the recommendation of decisions. To find the right features for a decision task it is essential to offer a corresponding configuration functionality. In this chapter we illustrate how the design of a decision task can be represented as a configuration problem. The underlying configuration knowledge is already integrated in a tool called CHOICLA.

5.2. Introduction

Decisions have to be taken in different situations - for example a decision about the destination for the next holidays or a decision about which restaurant to choose for a dinner with friends. Decision scenarios can differ from each other in terms of their process design. Some decision scenarios rely on a preselected decision heuristic that defines the criteria for taking the decision, for example, a group decides to use majority voting for deciding about the next restaurant visit. Furthermore, the visibility of the preferences of other users is an important feature that can be configured by the creator of a decision task.

In this chapter we show how the design of decision tasks (the underlying process) can be defined as a configuration problem. The major advantage of this approach is that making the process design of decision tasks configurable introduces the flexibility that is needed due to the heterogeneity of decision

problems. This way we are able to build a model that is flexible with regard to the implementation (generation) of problem-specific decision applications. The knowledge representations introduced in the following are included in the CHOICLA decision support environment (see www.choicla.com).

The remainder of this chapter is organized as follows. In the next section (Section 5.3) we discuss features that are essential to the design of a decision task. In Section 5.4 we introduce dependencies that exist between features. In Section 5.5 we provide insights into group recommendation approaches integrated in the CHOICLA environment. We then discuss related and future work and thereafter conclude the chapter.

5.3. Configuring a decision task

In the following we discuss different features that are relevant when designing (configuring) a decision task. On a formal level, we represent a *decision task configuration problem* as a constraint satisfaction problem Mackworth (1977) and (Felfernig et al. (2014b)) (CSP – see Definition 1).

Definition 1 (Constraint Satisfaction Problem). A CSP consists of (1) a set of finite-domain variables $X = \{x_1, x_2, \dots, x_n\}$ and (2) a set of constraints $C = \{c_1, c_2, \dots, c_m\}$. For each variable x_i out of X there exists a finite set D_i (domain of the variable) of possible assignments. Possible variable assignments can be limited via constraints. A complete assignment (every variable has a corresponding value) which is consistent with the constraints in C is denoted as a solution for a CSP.

For the purpose of better understandability we use a feature model notation to express variability properties of decision tasks. A feature model (FM) represents a set of possible features and relationships between them. Features are arranged hierarchically which is basically a tree structure with one root feature (Benavides et al. (2010)). Within this tree structure the nodes are the features and the edges are the relationships (constraints). A more detailed discussion of different feature model representations can be found in Batory (2005), Benavides et al. (2010) and Felfernig et al. (2013a).

Six different types of constraints (relationships) are typically used for the construction of feature models (Batory (2005), Benavides et al. (2010)): *mandatory*, *optional*, *alternative*, *or*, *requires* and *excludes*. Feature models are representing configurable products which can be formalized in the form of a CSP. A feature f is *included* if the value is set to 1 - otherwise it is said to be *excluded*. We will exemplify this formalization on the basis of feature model depicted in Figure 5.1. Figure 5.1 shows a fragment of the CHOICLA feature model ¹.

The CSP representation of the feature model depicted in Figure 1 is the following:

$$V = \{f_1, f_2, \dots, f_{21}\}$$

¹A more in-depth discussion of the CHOICLA decision support environment can be found in Stettinger et al. (2013).

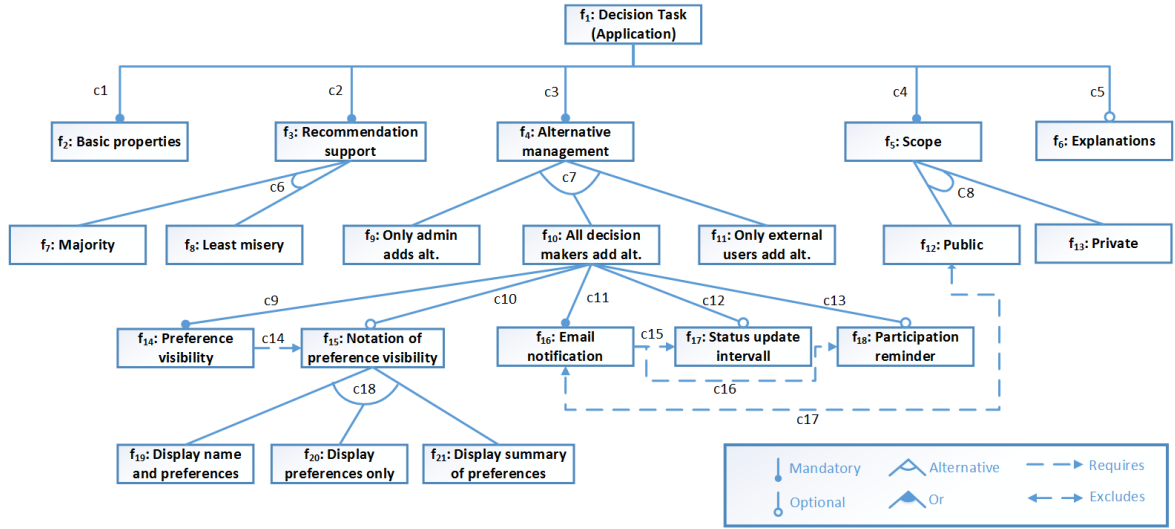


Figure 5.1.: Fragment of the CHOICLA feature model. In this model, f_i are used as abbreviation for the individual features, for example, f_1 is the short notation for feature *Decision Task (Application)*.

$$\text{dom}(f_1) = \text{dom}(f_2) = \dots = \text{dom}(f_{21}) = \{0, 1\}$$

$$c_1 : f_1 \leftrightarrow f_2$$

$$c_2 : f_1 \leftrightarrow f_3$$

$$c_3 : f_1 \leftrightarrow f_4$$

$$c_4 : f_1 \leftrightarrow f_5$$

$$c_5 : f_6 \rightarrow f_1$$

$$c_6 : (f_7 \leftrightarrow (\neg f_8 \wedge f_3)) \wedge (f_8 \leftrightarrow (\neg f_7 \wedge f_3))$$

$$c_7 : (f_9 \leftrightarrow (\neg f_{10} \wedge \neg f_{11} \wedge f_4)) \wedge (f_{10} \leftrightarrow (\neg f_9 \wedge \neg f_{11} \wedge f_4)) \wedge (f_{11} \leftrightarrow (\neg f_9 \wedge \neg f_{10} \wedge f_4))$$

$$c_8 : (f_{12} \leftrightarrow (\neg f_{13} \wedge f_5)) \wedge (f_{13} \leftrightarrow (\neg f_{12} \wedge f_5))$$

$$c_9 : f_{14} \leftrightarrow f_{10}$$

$$c_{10} : f_{15} \rightarrow f_{10}$$

$$c_{11} : f_{16} \leftrightarrow f_{10}$$

$$c_{12} : f_{17} \rightarrow f_{10}$$

$$c_{13} : f_{18} \rightarrow f_{10}$$

$$c_{14} : f_{14} \rightarrow f_{15}$$

$$c_{15} : f_{16} \rightarrow f_{17}$$

$$c_{16} : f_{16} \rightarrow f_{18}$$

$$c_{17} : \neg(f_{16} \wedge f_{12})$$

$$c_{18} : (f_{19} \leftrightarrow (\neg f_{20} \wedge \neg f_{21} \wedge f_{15})) \wedge (f_{20} \leftrightarrow (\neg f_{19} \wedge \neg f_{21} \wedge f_{15})) \wedge (f_{21} \leftrightarrow (\neg f_{19} \wedge \neg f_{20} \wedge f_{15}))$$

We will now discuss different basic properties of decision task configuration problems. In this context we explain the individual features and constraints depicted in Figure 5.1.

Basic properties. Each decision task is characterized by a name, a corresponding description, and a picture that represents the decision task (summarized in the feature *Basic Properties* for simplification purposes).

Management of alternatives. There are different possibilities to support alternative management within the scope of a decision task. *First*, only the creator of a decision task is allowed to add alternatives – this could be the case if a person is interested to know the opinions of his/her friends about a certain set of alternatives (e.g., alternative candidates for the next family car). Another related scenario are so-called "Micro-Polls" where the creator is only interested in knowing the preference distribution of a larger group of users. *Second*, in some scenarios it should be possible that all decision makers can add alternatives – a typical example of such a scenario is the group-based decision regarding a holiday destination or a hotel Jameson (2004). In this context, each user should be allowed to add relevant alternatives. An example scenario of the *third* case (only external users can add alternatives) is the support of group-based personnel decisions – in this context it should be possible that persons apply for a certain position (the application itself is interpreted as the addition of a new alternative to the decision task).

Scope. The scope of a decision task denotes the external visibility. The scope "private" allows only invited users to participate, i.e., the task is not visible for other users except those who have been invited. If the scope is "public", the decision task is visible to all users – this is typically the case in the context of so-called Micro-Polls. The selection of the scope has an impact on other features – related aspects will be discussed in Section 5.4.

Preference visibility. The visibility of individual preferences of the other participants involved in a decision process can have an impact on decision quality (see Felfernig et al. (2012b), Jameson (2004), and Jameson and Smyth (2007)). There occur some decision scenarios where all participants should exactly know which person articulated a rating of an alternative. If, for example, a date for a business meeting is the topic of the decision task it is very essential to find a date where all division managers can attend the meeting and therefore it is important to know the individual preferences of the participants in that case. But there are of course decision scenarios where preference visibility can lead to disadvantages for some participants but still some kind of transparency of the preferences is helpful to come to the best decision. In such cases a summary of all given preferences of an alternative is a good way to support the participants best during the decision process. A summary prevents all participants from statistical inferences but still can help participants who are not sure about which rating to select.

Email notification. If this feature is set, emails can be used to exchange information about the current state of the decision process. For example, the status update interval specifies in which intervals participants of a decision process receive a summary of the current status of the decision process. The

restaurant	Martin	Dave	George	Ben
Clocktower	5	3	5	4
Häuserl im Wald	3	3	5	3
La Botte	5	3	3	3
El Gaucho	4	3	4	4

Table 5.1.: Examples of user-specific ratings with regard to the available decision alternatives (restaurants).

active participation reminder is a feature which helps to trigger need for closure. If this feature is set, a maximum inactive time (without looking at the current status of the decision task) for the participants can be set. After this time is elapsed an email will be sent to the corresponding participants to encourage an active participation at the decision task.

Recommendation support. In context of group decision tasks another very essential aspect is the aggregation function (recommendation heuristic). Aggregation functions can help to foster consensus in a group decision process, furthermore, user studies show that these functions also help to increase the degree of the perceived decision quality (see, for example Felfernig et al. (2012b)). Preferences of individual users can be aggregated in many different ways and there exists no standard heuristic which fits for every decision scenario. To support groups of users in different scenarios the selection of recommendation heuristics is a necessary feature which has to be configured by the creator of a decision task. Some basic aggregation heuristics which can be used in such cases are described below. For an in-depth discussion of basic types of aggregation heuristics see, for example, the overview of Masthoff (Masthoff (2011)). The example given in Table 5.1 represents the individual ratings of the participants for the defined alternatives. The results of applying the decision heuristics discussed below are depicted in Table 5.2.

Majority Voting (see Formula 5.1) determines the value (d) that a majority of the users selected as voting for a specific solution s where $eval(u, s)$ denotes the rating for solution s defined by user u . For example, the majority of votings for *Clocktower* is 5 (see Table 5.2).

$$MAJ(s) = \max_{d \in \{1..5\}} (\#(\bigcup_{u \in Users} eval(u, s) = d)) \quad (5.1)$$

Least Misery (see Formula 5.2) returns the lowest voting for solution s as group recommendation. For example, the LMIS value for the $s = \textit{Clocktower}$ is 3.

$$LMIS(s) = \min(\bigcup_{u \in Users} eval(u, s)) \quad (5.2)$$

Most Pleasure (see Formula 5.3) returns the highest voting for solution s as group recommendation.

solution	MAJ	LMIS	MPLS	GDIS	ENS
Clocktower	5	3	5	5	5
Häuserl im Wald	3	3	5	3	3
La Botte	3	3	5	3	3
El Gaucho	4	3	4	4	4

Table 5.2.: Results of applying the aggregation functions to the user preferences shown in Table 5.1. MAJ = Majority Voting; LMIS = Least Misery; MPLS = Most Pleasure; GDIS = Lowest Group Distance; ENS = Ensemble Voting. This example is based on the preference information in Table 5.1.

For example, the MPLS value for the $s = \textit{Clocktower}$ is 5.

$$MPLS(s) = \max\left(\bigcup_{u \in Users} eval(u, s)\right) \quad (5.3)$$

Group Distance (see Formula 5.4) returns the value d as group recommendation which causes the lowest overall change of the individual user preferences. For example, the GDIS value for $s = \textit{Clocktower}$ is 5 (or, alternatively 4).

$$GDIS(s) = \minarg_{(d \in \{1..5\})} \left(\sum_{u \in Users} |eval(u, s) - d| \right) \quad (5.4)$$

Finally, *Ensemble Voting* (see Formula 5.5) determines the majority of the results of the individual voting strategies $H = \{\textit{MAJ}, \textit{LMIS}, \textit{MPLS}, \textit{GDIS}\}$. For example, the ensemble-based majority voting for *Clocktower* is 5.

$$ENS(s) = \maxarg_{(d \in \{1..5\})} \left(\# \left(\bigcup_{h \in H} eval(h, s) = d \right) \right) \quad (5.5)$$

Explanations. Explanations can play an important role in decision tasks since they are able to increase the trust of users in the outcome of a decision process (Felfernig et al. (2006)). When configuring a decision task in CHOICLA, explanations can be selected as a feature of the decision process. In the current version of CHOICLA, explanations are supported by simply allowing the creator of the decision process to include textual argumentations as to why a certain decision alternative has been selected as "the final decision". If this feature is selected, the administrator of a decision task has to enter some explanatory text, if not, the entering of such a text remains just an option.

5.4. Dependencies among features

We now discuss examples of constraints that restrict the combinations of features as shown in the feature model of Figure 5.1. The constraint-based representation of these constraints is shown in the CSP definition of the feature model given in Section 5.3.

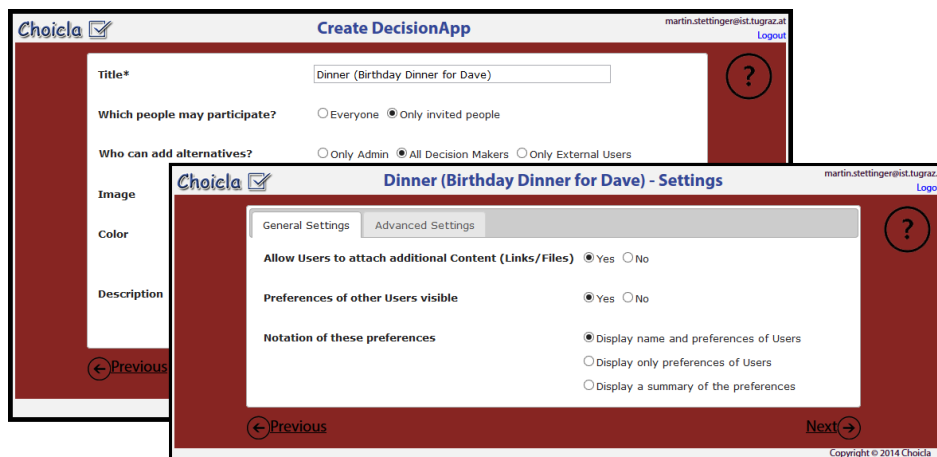


Figure 5.2.: CHOICLA: definition of a decision task. Basic settings & further configurable features in case the decision makers are allowed to contribute own alternatives during the decision process.

Scope of a decision. If a decision task is public, there are restrictions regarding the support of message interchange (e.g., via email) and the visualization of the preferences of other users. In the case that a decision task is private, it is in both cases possible to choose. Preferences can (but must not) be made visible to other users and the type of possible message interchange can be specified. The differentiation between public and private decision tasks also has an impact on other system properties. For example, if a decision task is defined as private, the corresponding decision application can not be reused by other users, i.e., found as a result via the CHOICLA search interface.

Preference visibility. A dependency of type 'requires' exists between the feature *preference visibility* and the corresponding notation of visibility. Preference visibility denotes a functionality where the individual preferences of other users are made visible for the current user. The type of visualization can only be selected in the case that the *preference visibility* feature is has been selected by the designer of a decision task.

Email notification. Similar to the visibility of preferences, the type of supported message exchange (e.g., via email) can only be specified in the case that the creator of the decision task decided to support email notifications. As already mentioned, email communication is only supported if the scope of the decision task is private.

These simple examples already show the need to manage decision task related variability in a structured fashion. Our knowledge representation approach allows for a product line oriented development

of decision support functionalities and makes systems much more flexible for future requirements and corresponding extensions.

5.5. Configuring decision tasks in CHOICLA

In the following we give an example of how a decision task can be configured in the CHOICLA decision support environment (www.choicla.com). The application knowledge base of CHOICLA is currently rule-based. For reasons of easier maintenance and adaptability we apply reasoning and CSP for future versions of CHOICLA.

Parts of the user interface that supports a creator of a decision task are depicted in Figure 5.2. The possible parametrizations correspond to the features in the model of Figure 5.1. If, for example, a specific feature A depends on the inclusion of another feature B, this is taken into account in the user interface, i.e., such a feature (feature A) can only be selected, if the other feature (feature B) is also selected. In the example of Figure 5.2, the *scope* of the decision task is private (only invited users can participate), all decision makers are allowed to add alternatives, and for all participants of the decision process the preferences of other users are visible (names as well as preferences). Note that in the CHOICLA environment there are many additional features that can be selected within the scope of a decision task configuration process.

For understandability reasons we kept our working example simple and focused on aspects that give the reader an impression of the basic underlying configuration problem. The user interface for the inclusion of alternatives is depicted in Figure 5.3.



Figure 5.3.: CHOICLA: user interface for addition of decision alternatives. The dots in the upper right corner of every symbol indicate whether there is an information in this category available or not. The meaning of the used symbols is (from left to right): *edit, delete, geographical information, files, links and comments*.

Figure 5.4 shows how the decision alternatives can be voted by the individual users of a decision task.

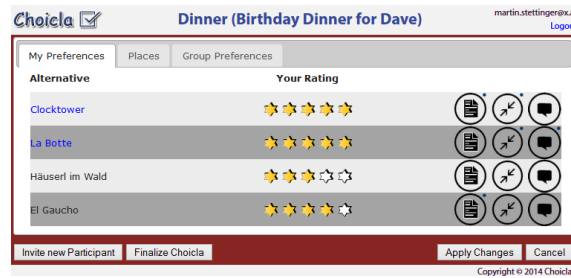


Figure 5.4.: CHOICLA: user interface for individual voting of decision alternatives. Each alternative can be voted by a five-star scale. The tab *Places* shows the geographical distribution of the decision alternatives (if available). In tab *Group Preferences* the actual group recommendation as well as the individual preferences of the other users (if feature f_{14} is set) is presented to the users. The process where the "final decision" can be set is triggered by the button *Finalize Choicla*.

5.6. Related and Future Work

There exist a couple of online tools which support different types of decision scenarios. *The Decider*² is a tool that allows the creation of issues and decision alternatives – the corresponding recommendation is provided to users who are articulating their preferences regarding the given decision alternatives. Rodriguez et al. (Rodriguez et al. (2007)) introduce *Smartocracy* which is a decision support tool which supports the definition of tasks (issues or questions) and corresponding solutions. Solution selection (recommendation) is based on exploiting information from an underlying social network which is used to rank alternative solutions. *Dotmocracy*³ is a method for collecting and visualizing the preferences of a large group of users. It is related to the idea of participatory decision making – it's major outcome is a graph type visualization of the group-immanent preferences. *Doodle*⁴ focuses on the aspect of coordinating appointments – similarly, VERN (Yardi et al. (2005)) is a tool that supports the identification of meeting times based on the idea of unconstrained democracy where individuals are enabled to freely propose alternative dates themselves. Compared to CHOICLA these tools are not able to customize their decision processes depending on the application domain and are also focused on specific tasks. Furthermore, no concepts are provided which help to improve the overall quality of group decisions, for example, in terms of integrating explanations, recommendations for groups, and consistency management for user preferences.

The support of group decision processes on the basis of recommendation technologies is a new and upcoming field of research (see, e.g., Masthoff et al. (Masthoff (2011))). The application of group recommendation technologies is still restricted to specific domains such as interactive television (Masthoff (2004)), e-tourism (Jameson (2004); McCarthy et al. (2006)), software requirements engineering (Felfernig et al. (2012b)), and ambient intelligence (Perez et al. (2010)).

²labs.riseup.net.

³dotmocracy.org.

⁴doodle.com.

Future Work. Our future work will focus on the analysis of further application domains for the CHOICLA technologies. Our vision is to make the design (implementation) of group decision tasks as simple as possible. The resulting decision task should be easy to handle for users and make group decisions in general more efficient. Within the scope of our work we will also focus on the analysis of decision phenomena within the scope of group decision processes. Phenomena such as decoy effects (Huber et al. (1982)) and anchoring effects (Jacowitz and Kahneman (1995)) are well known for single-user cases but are not investigated in group-based decision scenarios. Finally, we will also focus on the development of further group recommendation heuristics. In this context, our major goal is to make the CHOICLA datasets available to the research community in an anonymized fashion for experimentation purposes.

5.7. Conclusions

In this chapter we have shown how to represent the design of decision tasks as a configuration problem. In this context, we gave a short introduction to the CHOICLA group decision environment which supports the flexible design and execution of different types of group decision tasks. Compared to existing group decision support approaches, CHOICLA provides an end user modelling environment which supports an easy development and execution of group decision tasks.

Choicla: Intelligent Decision Support for Groups of Users in the Context of Personnel Decisions

Parts of this chapter have been published in Stettinger (2014), Stettinger and Felfernig (2014), and Stettinger et al. (2013). Imperative parts of this chapter, such as literature research, algorithmic approaches as well as the user study have been provided by the author of this thesis.

6.1. Abstract

Group recommendation technologies have been successfully applied in domains such as interactive television, music, and tourist destinations. Existing technologies are focusing on specific domains and do not offer the possibility of supporting different kinds of decision scenarios. The *Choicla* group decision support environment advances the state of the art by supporting decision scenarios in a domain-independent fashion. In this chapter we present an overview of the *Choicla* environment and exemplify its application in the context of personnel decisions.

6.2. Introduction

Decisions in everyday life often come up in groups, for example, a decision about the destination for the next holidays or a decision about which restaurant to choose for a dinner. Knowledge about the preferences of other users in early phases of a decision process can lead to sub-optimal decision outcomes (Mojzisch and Schulz-Hardt (2010)). Missing explanations can lead to a lower level of trust

in recommendations (Felfernig et al. (2006)). So-called anchoring effects (Jacowitz and Kahneman (1995)) are responsible for decisions which are biased by the voting of the first preference-articulating person. If single persons have to take a decision in place of persons who are not available for a meeting, the outcome of the decision can also be negatively influenced. Decision processes are often not open in the sense that it is impossible to easily integrate new decision alternatives or change the individual preferences within the scope of a decision process - both aspects can lead to low-quality decision outcomes (see Molin et al. (1997)). In many cases, the criteria for the decision remain unclear since there is no explanation of the outcome of "the final decision". All these mentioned threats can negatively influence the quality of group decisions.

One major goal of the *Choicla* environment is to facilitate group decision making and improve the overall quality of decision outcomes. The idea of this environment is to support definitions of different types of decision tasks in a domain-independent fashion while taking into account the above mentioned risk factors. In order to achieve this goal, *Choicla* builds upon different group recommendation algorithms (Masthoff (2011)) which are used for determining alternative solutions for the participants of a group decision process.

One example of the application of *Choicla* is to support groups of users in context of personnel decisions with the aim of achieving a more structured, fair, and transparent way of job interviews as well as to find the most suitable candidate for the job advertisement. Other typical scenarios for the application of *Choicla* technologies are the decision about which restaurant to select for a dinner or - in a scientific community - a decision regarding the selection of the destination of next year's conference. The remainder of this chapter is organized as follows. In Section 6.3 we provide insights to (1) the *Choicla* modelling process where participants can design decision tasks from scratch and (2) the intelligent management of already created decision apps. In Section 6.4 we give an overview of the personnel decision scenario. We then discuss related & future work (Section 6.6) and thereafter conclude the chapter.

6.3. Choicla Decision Support

Because decision scenarios differ from each other in their process design, a variety of parameters is needed to specify all relevant properties of a decision task. We will now discuss basic features (parameters) which can be configured (modelled) by the creator of a decision task. In this context we refer to the example features depicted in Figure 6.1.

6.3.1. Design of Decision Apps

Because decision scenarios differ from each other, some decision scenarios rely on a preselected decision heuristic that defines the criteria for taking the decision, for example, a group decides to use *majority voting* for deciding about the next restaurant or cinema visit. The design of decision tasks

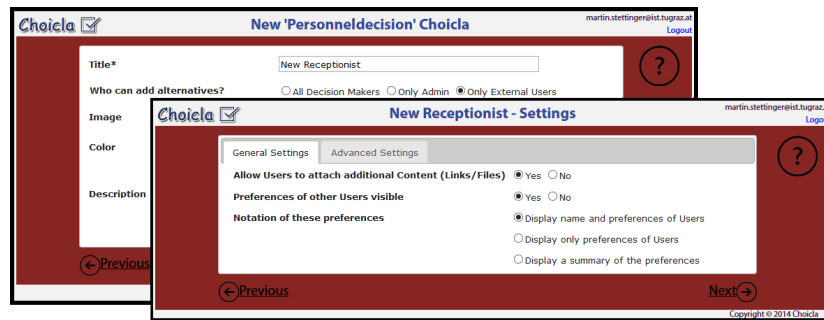


Figure 6.1.: *Choicla*: definition of a decision task. Basic settings & further configurable features.

(the underlying process) can be interpreted as a configuration problem (see Stettinger et al. (2014)). The achieved flexibility of making the process design of a decision task configurable is needed due to the heterogeneity of decision problems. This way the *Choicla* components are organized as a kind of a software product line that is open in terms of the implementation (generation) of problem-specific decision applications.

Explanations. Explanations can have an important role in decision tasks since they are able to increase the trust of users in the outcome of a decision process (Felfernig et al. (2006)). When designing a decision task in *Choicla*, explanations can be selected as a feature of the decision process. If this feature is selected, the administrator of a decision task has to enter some explanatory text, if not, the entering of such a text remains just an option.

Administration of Decision Alternatives. The administration of decision alternatives within the scope of a decision task can be supported in different ways. *First*, only the initiator of a decision task is allowed to add alternatives – this could be desired if a person is interested in knowing the opinions of his/her friends about a concrete set of alternatives (e.g., alternative candidates for the next family car). Another related scenario are so-called ”Micro-Polls” where the initiator is only interested in knowing the preference distribution of a larger group of users. *Second*, in some scenarios it is important that all decision makers can add alternatives during the decision task by themselves – a common example of such a scenario is the group-based decision regarding a holiday destination or a hotel (Jameson (2004)). In such a context, each participant should be allowed to add relevant alternatives. The support of group-based personnel decisions can be seen as an example scenario of the *third* case (only external users can add alternatives) – in this context it should be possible that candidates apply for a certain job position (the application itself is interpreted as the addition of a new alternative to the decision task). The selection of the next conference location where proposers can submit their material is another example.

Preference Visibility. The scope ”private” allows only invited users to participate, i.e., the decision

task is only accessible for invited users and not accessible for other users. If the scope is "public", the decision task is accessible for all users – this is typically the case in the context of so-called Micro-Polls. The decision quality can be influenced if the individual preferences of the other participants are visible during the decision process (see Felfernig et al. (2012b) and Jameson (2004)). There exist decision scenarios where all participants profit from the knowledge of who entered which rating. If, for example, the decision task is to find a date for a business meeting it is essential to find a date where all managers can attend the meeting and therefore it is important to know the individual preferences of the participants. On the other hand there are decision scenarios where full preference visibility can lead to disadvantages for some participants but some kind of transparency of the individual preferences is helpful to achieve a reasonable decision. In such cases a summary of all given preferences is a feasible way to support decision makers (participants). A summary prevents the participants from statistical inferences to the individual preferences but still can help participants who are unsure about how to rate.

Recommendation Support. In the context of group decision tasks, an essential aspect is the aggregation function (recommendation heuristic). In a group decision process aggregation functions can help to foster consensus. User studies show that these functions also help to increase the degree of the perceived decision quality (see, for example Felfernig et al. (2012b)). Individual user preferences can be aggregated in many different ways and there exists no default heuristic which fits for every decision scenario. To provide a support for groups of users in different decision scenarios, the selection of recommendation heuristics is a key feature which has to be configured by the initiator of a decision task. Due to space limitations we only describe selected aggregation heuristics below. Masthoff (Masthoff (2011)) gives an overview of basic aggregation heuristics such as *Majority Vote (MAJ)*, *Average Vote (AVV)*, *Least Misery (LMIS)*, and *Most Pleasure (MPLS)* which are also available in the *Choicla* environment.

Group Distance (GD) (see Formula 6.1) returns the value d as group recommendation which causes the lowest overall change of the individual user preferences where $eval(u, s)$ denotes the rating for a solution s defined by user u .

$$GD(s) = \underset{d \in \{1..5\}}{\operatorname{minarg}} \left(\sum_{u \in Users} |eval(u, s) - d| \right) \quad (6.1)$$

Ensemble Voting can be seen as an example of a meta-aggregation function included in *Choicla*. Ensemble Voting (see Formula 6.2) determines the majority of the results of the individual voting strategies $H = \{MAJ, AVV, LMIS, MPLS, GD\}$ where $eval(h, s)$ denotes the result of an individual voting strategy for a solution s .

$$ENS(s) = \underset{d \in \{1..5\}}{\operatorname{maxarg}} \left(\# \left(\bigcup_{h \in H} eval(h, s) = d \right) \right) \quad (6.2)$$

To achieve fairness among all group members in long term scenarios (e.g., recurring decision tasks), we developed a modified version of the *Group Distance* algorithm (see Formula 6.4), where all past decisions influence the actual recommendation. A user-weighting $w(u)$ is used to ensure that users whose individual ratings often differ from related decisions (represented by $distance(u)$), are privileged.

$$w(u) = \frac{1}{\frac{1}{\sum distances(u)+1}} \quad (6.3)$$

$$GD'(s) = \mathit{minarg}_{d \in \{1..5\}} \left(\sum_{u \in Users} w(u) * |eval(u,s) - d| \right) \quad (6.4)$$

		North& South	Trattoria	El Gauchò	La Botte
Dec. Sit. 1	Martin	5	3	4	4
	Dave	2	3	5	3
	George	5	3	3	3
	Ben	4	3	4	4
	Rec. GD	5	3	4	4
Dec. Sit. 2	Martin	5	2	2	4
	Dave	3	3	4	2
	George	4	3	3	3
	Ben	4	4	3	5
	Rec. GD	4	3	3	4
Dec. Sit. 3	Martin	5	3	4	3
	Dave	3	5	3	3
	George	4	3	4	5
	Ben	4	4	5	5
	Rec. GD'	4	5	4	3

Table 6.1.: User-specific ratings with regard to the available decision alternatives (restaurants) in recurring decision situations. The group recommendation is marked bold in the corresponding decision situation.

If we look at the example in Table 6.1 we notice that in *Dec. Sit. 3* the group recommendation is *Trattoria* although the individual aggregation heuristics (Masthoff (2011)) would calculate a different group recommendation. *Dec. Sit. 1 and 2* are responsible for a user-weighting in *Dec. Sit. 3* which leads to this recommendation. The used weights of the participants calculated by Formula 6.3 in *Dec. Sit. 3* are the following: **Martin: 1, Dave: 6, George: 2, and Ben: 3**. *Dave* has the highest weight in *Dec. Sit. 3*. This approach allows to achieve fairness among the group members in long term decisions.

6.3.2. Choicla Decision Apps

After the design process has been finished, the creator of the decision task as well as all invited participants (after accepting the invitation) see a corresponding decision app directly on the personal home screen (see Figure 6.2).

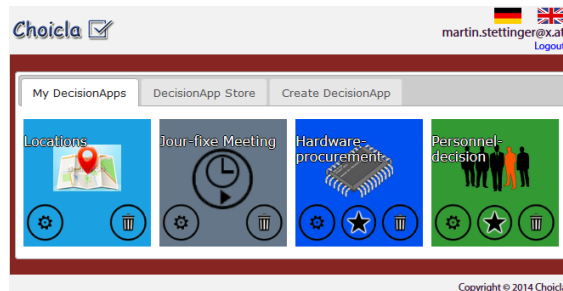


Figure 6.2.: *Choicla*: Home screen of a registered user. The symbols within the tiles trigger actions which can be performed in the current state of the decision app. Possible actions are (from left to right): configuration, evaluation (only possible if the decision app is publicly available over the store), and delete.

The tab *DecisionApp Store* contains publicly available decision apps which can be searched and installed on the personal *Home Screen*. This method prevents a creation from scratch every time for frequent decision tasks such as, for example, scheduling decision tasks. In such a case the decision process can be triggered right after the download of a decision app. This reuse technique has the potential to reduce the entry barrier for using *Choicla* and keep the interaction simple – especially for people who want to start a decision process quickly. The tab *Create DecisionApp* allows a user to design a completely new decision app from scratch.

Due to the fact that many decision tasks occur regularly – for example, a group of friends go for dinner once a month – a concept is needed to manage a potentially large number of decision tasks. To keep the potentially large number of decision tasks manageable, every decision app consists of a variable number of instances. A concrete instance of a decision app can be accessed within the corresponding decision app - all instances of a concrete decision app will be loaded when the decision app is opened. The created instance of the example depicted in Figure 6.1 is accessible in the "Personnel-decision" app (see Figure 6.2). This mechanism offers the possibility of an exact documentation of all past decisions and is also a basis for supporting recurring decision tasks.

6.4. Choicla Personnel Decisions

6.4.1. Users View

Personnel decisions are often influenced by various factors. Such factors are, for example, if a candidate has physical handicaps, in most cases no concrete structure is followed during the job interview

and the evaluation often gets subjective. In such a case the assessment criteria of the candidates change and no "fair" and objective decision can be made. Another important factor is that in most cases personnel decisions come up in groups of users which means that often more than one person is affected by the hiring procedure.

To prevent groups from unsystematic reviews, *Choicla* offers a structured and fair way to evaluate candidates of a job position. Figure 6.3 shows the evaluation of the candidates in context of our working example (new receptionist) for a particular decision maker.



Figure 6.3.: *Choicla*: example of individual ratings. Each user can take a look at the current recommendation and adapt his/her preferences if needed.

To keep the screen understandable, only the line with the aggregated information of a candidate is visible - by clicking on this line, several dimensions including their actual ratings show up for the corresponding candidate (only visible for first candidate in Figure 6.3). In order to avoid misunderstandings in context of evaluation the sliders of the first candidate are automatically displayed if the screen is loaded. Due to the fact that depending on the advertised job position different assessment criteria are needed, the dimensions on which a candidate can be evaluated can be chosen by the creator of a decision task. If we look at the example in Figure 6.3 we can see that for the "New Receptionist" the dimensions *English skills*, *Communication*, *Friendliness*, and *Punctuality* are chosen.

In situations where there are candidates for whom not all criteria (dimensions) have been evaluated or there exists a discrepancy between individual evaluations, special markers are used to point out open issues. This approach creates need for closure (see, e.g., Ninaus et al. (2014)), i.e., users are additionally motivated to make the candidate evaluations complete and consistent.

If a candidate should be excluded from the application procedure in early phases (e.g., some criteria are not met), this can be achieved by using the "Manage Candidates" button (a new menu shows up). The early exclusion of an unsuitable candidate supports more clarity since only the "relevant" candidates are displayed.

The tab *Group Preference* presents the current group recommendation, after a predefined number (the threshold) of participants articulated their preferences. This threshold prevents from statistical inferences to the individual preferences of other participants (only in combination with a "private" decision

scope - see Section 6.3). The group recommendation in context of personnel decisions is based on the MAUT-principle (multi-attribute-utility-theory (Dyer (2005))). A group recommendation based on the MAUT-principle (see Formula 6.5) returns the average value of all individual MAUT values of all participants as group recommendation for one candidate (solution s). A group member's individual MAUT value represents the weighted average of all personal ratings of the dimensions of an alternative. This means that the attribute values are subjective and the weights are fixed which is different in a typical MAUT scenario.

$$MAUT(s) = \sum_{u \in Users} \frac{\sum_{d \in s} eval(u, d) * weight(d)}{|dimensions|} \quad (6.5)$$

If we look at the individual ratings in Figure 6.3 we notice the values 8, 5, 8, and 5 for the dimensions. For simplification purposes we assume in our example that all dimensions have the same weight ($w_{d1} = w_{d2} = w_{d3} = w_{d4} = 5$). Due to Formula 6.5, the individual MAUT value for the actual user of the first alternative is 32.5. To present the evaluation of a solution (candidate) within a five star scale, these values have to be normed.

6.4.2. Candidates View

All previous described options and screens can only be accessed by the decision makers of the decision task itself and can of course not be seen by the applicants of the job position. During the design phase of a decision task the input fields (e.g., name, age, and application text) which are then visible by the applicants during the application process can be defined. Figure 6.4 shows the view of an applicant in our running example "New Receptionist".

Figure 6.4.: *Choicla*: example of the entering of application data. Each applicant can insert his/her personal data needed for the advertised job position.

All the added information of the candidates is then prepared and accessible for the decision makers

during the assessment phase - see Figure 6.3. This way of adding solutions to a decision process shifts the burden of entering candidate information by a single person - in most cases a secretary - to the applicants.

6.5. User Study

In order to evaluate the *Choicla* group decision environment, we conducted a first small-scale user study. In this study we made the software accessible to ten persons outside our university who were not aware of the fact that the system has been developed at our university. Each of these participants defined a decision task with the support of the *Choicla* environment and then made accessible the decision task to a group of other persons. The average number of persons participating in a decision scenario was around 5. The overall number of users who interacted with *Choicla* within the scope of our study was N=48 (38% female, 62% male). Within the scope of this empirical study we could gain a first evidence that users consider the current version of the system applicable and can imagine to apply *Choicla* functionalities in various domains. For an in-depth discussion of this user study we refer to Stettinger et al. (2013).

6.6. Related & Future Work

There exist a couple of online tools supporting decision scenarios. Rodriguez et al. (Rodriguez et al. (2007)) describes a system called *Smartocracy*. *Smartocracy* is a decision support tool which supports the definition of tasks in terms of issues or questions and corresponding solutions. The recommendation (solution selection) is based on exploiting information from an underlying social network which is used to rank alternative solutions. *Dotmocracy*¹ includes a method for collecting and visualizing the preferences of a large group of users. It is related to the idea of participatory decision making – its major outcome is a graph type visualization of the group-immanent preferences. *Doodle*² is an internet calendar tool with the focus on coordinating appointments. *VERN* (Yardi et al. (2005)) is (very similar to *doodle*) a tool that supports the identification of meeting times. *VERN* is based on the idea of unconstrained democracy where individuals are enabled to freely propose alternative dates themselves. A major advantage of *Choicla*³ compared to these tools is that users of *Choicla* are able to customize their decision processes depending on the application domain and can also focus on specific tasks. Furthermore, the mentioned tools provide no concepts which help to improve the overall quality of group decisions, for example, in terms of integrating explanations, recommendations for groups, and consistency management for user preferences.

Recommendation approaches in the line of *Choicla* are also presented in Sangeetha et al. (Kutty et al.

¹dotmocracy.org.

²doodle.com.

³www.choicla.com.

(2012)) and Malinowski et al. (Malinowski et al. (2008)). Sangeetha et al. (Kutty et al. (2012)) introduce recommendation approaches that support people-to-people recommendation (detection of latent relationships between similar users) whereas Malinowski et al. (Malinowski et al. (2008)) discuss approaches (based on fitness measures) that support the pre-selection of candidates for existing teams (groups). In contrast, *Choicla* focuses on supporting a group decision where parameters such as the fit of a candidate with an existing group are represented in terms of MAUT dimensions.

Our *future work* will focus on the analysis of further application domains for the *Choicla* technologies. Our vision is to make the design (implementation) of group decision tasks as simple and straightforward as possible. The resulting decision task should be easy to handle for users and make group decisions in general more efficient. Our focus will also be on the analysis of decision phenomena within the scope of group decision processes. Phenomena such as decoy effects (Huber et al. (1982), Teppan and Felfernig (2009)) and anchoring effects (Jacowitz and Kahneman (1995)) have been well studied for single-user cases, however, in group-based decision scenarios no studies have been conducted.

Biases can be induced if a system is open in the sense that new decision alternatives can be added during the decision process. However, such a feature is imperative in cases where all possible decision alternatives are not available from the beginning. The group preferences can also be influenced by the order of the incoming individual preferences due to the fact that the participants of a group will perceive already selected alternatives more attractive than new options (Neale et al. (2004)). If consensus out of discussion is reached in early phases, literature shows that this consensus is cognitive resistant to changes. That means that additional information which is added later in a decision process will be adapted to already defined consensus and due to this it is very unlikely that another alternative is chosen (Lind et al. (2001)). Such a phenomenon can be explained by the *assimilating effect* which is ascribable to the *dissonance theory* (Festinger (1957)). The *assimilating effect* states that individuals are motivated to reduce psychological incongruity or discrepancy that is very likely to arise if new information is added to a present perception (Neale et al. (2004)). A high group cohesion intensifies this effect, because within such a group the fear of exclusion is higher (see Lind et al. (2001)). Future versions of *Choicla* will reduce this effect by providing a special way of preference visibility which, for example, only shows the preferences of other users for those participants who completed their individual ratings of the alternatives. Another research direction in this context is if such mechanisms can increase the willingness of participants to articulate their real preferences.

Another research direction is the support of long term decisions (Ariely and Zakay (2001)) (see Section 6.3.1). Fairness is an important issue in this context - the degree of perceived fairness influences the willingness of group members to accept compromises in the resolution of conflicts of opinions and also their trust in other group members (Lind et al. (2001)). A further issue for future work is to figure out which group recommendations help to achieve consensus more quickly.

We want to emphasize that one of our major goals is to make the *Choicla* datasets available to the research community in an anonymized fashion for experimentation purposes.

6.7. Conclusions

In this chapter we gave a short introduction to *Choicla* which supports the flexible design and execution of different types of group decision tasks with a focus on personnel decisions. With the help of *Choicla* it is possible to achieve more transparent, fair, and structured personnel decisions. Compared to existing group decision support approaches, *Choicla* provides an end user modelling environment which supports an easy development and execution of group decision tasks. We also discussed further research directions which can help to extend the available functionality of the *Choicla* environment.

Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment

This chapter is based on the results documented in Stettinger et al. (2015b). The author of this thesis provided major contributions in terms of literature research, intelligent preference acquisition interfaces as well as the user study and wrote major parts of this chapter.

7.1. Abstract

Decisions are often suboptimal due to the fact that humans apply simple heuristics which cause different types of decision biases. CHOICLA is an environment that supports decision making for groups of users. It supports the determination of recommendations for groups and also includes mechanisms to counteract decision biases. In this chapter we give an overview of the CHOICLA environment and report the results of a user study which analyzed two voting strategies with regard to their potential of counteracting serial position (primacy/recency) effects when evaluating decision alternatives.

7.2. Introduction

Several decisions in everyday life occur in the context of groups, for example, a decision regarding the restaurant to choose for a dinner with business partners or a decision regarding the cinema movie to watch with a group of friends. There exist various decision biases which can negatively influence the quality of group decisions (Mandl et al. (2010)). *Anchoring effects* (Jacowitz and Kahneman (1995)) are responsible for decisions which are biased by a shown reference value. For example,

the average rating of other users in the context of rating an item in collaborative filtering scenarios (Adomavicius et al. (2011)) or the evaluation of an item by the first group member shown to other group members can trigger anchoring effects (Felfernig et al. (2012b); Mojzisch and Schulz-Hardt (2010)). Anchoring biases in collaborative filtering scenarios can be counteracted with an adaptation of the preference acquisition interface, for example, by an adaptation of the underlying rating scale visualization (Adomavicius et al. (2014)). In the context of group decision scenarios, anchoring effects can be controlled by not completely disclosing the preferences of other group members in early stages of a decision process (Felfernig et al. (2012b)).

Another example of decision biases are *decoy effects* which denote the fact that irrelevant (inferior) items in an item set significantly influence the selection behavior of users (Tversky and Simonson (1993)). In the context of recommendation scenarios, decoy effects can occur in the item selection process, i.e., if a set of candidate items is recommended to a user (Teppan and Felfernig (2012)). Decoy effects can be detected and counteracted on the basis of predictive models that estimate dominance relationships between items in a candidate (consideration) set and propose to remove items from the candidate set before showing it to the user (Teppan and Felfernig (2012)).

In this chapter we focus on *primacy/recency effects* (serial position effects) that represent situations in which items presented at the beginning and the end of the list are evaluated significantly more often than items in the middle of a list (Murphy et al. (2012)). An explanation of this effect is that users are not interested in evaluating large lists to identify those items that best fit their preferences. Primacy/recency effects are also explained as a cognitive phenomenon since items at the beginning and the end of a list are also recalled more often (Felfernig et al. (2007a)). In recommendation scenarios these effects have been investigated, for example, by Felfernig et al. (Felfernig et al. (2007a)). The results of their studies clearly show that item properties shown to a user at the beginning and the end of a recommendation dialog are recalled significantly more often than properties in the middle of a dialog.

There exist a couple of online tools that support group decision scenarios. SMARTOCRACY provides support for voting scenarios in social network contexts where information from the social network is applied to rank recommendations (Rodriguez et al. (2007)). DOODLE¹ focuses primarily on the aspect of coordinating meetings and does not include additional mechanisms to determine recommendations for groups of users. Similarly, VERN (Yardi et al. (2005)) is a tool that supports the identification of meeting times based on the idea of unconstrained democracy where individuals are enabled to freely propose alternative dates themselves. DOTMOCRACY² deals with larger groups of users and provides a method for collecting and visualizing group preferences. The system is based on the idea of participatory decision making – it's major outcome is a graph type visualization of the group-immanent preferences. Compared to CHOICLA, these tools focus on specific scenarios, i.e., do not allow a flexible definition of decision functionalities depending on the scenario at hand. Furthermore, exist-

¹doodle.com.

²dotmocracy.org.

ing tools do not include recommendation and explanation functionalities which can help to increase trust in recommendations (Felfernig et al. (2006)) and improve the perceived decision support quality (Felfernig et al. (2012b)).

Typical CHOICLA scenarios range from industrial settings (e.g., selection of conference locations, selection of new employees, and evaluation of project proposals) to decision scenarios in private settings (e.g., selection of a restaurant for a dinner with friends, selection of a hotel for a holiday trip with friends, and the selection of a movie to watch with friends in a cinema). The contributions of this chapter are the following. (1) we present CHOICLA which is a novel domain-independent decision support tool for groups of users and (2) we show how to counteract primacy/recency effects occurring in the evaluation of decision alternatives.

The remainder of this chapter is organized as follows. In Section 7.3 we provide insights to the CHOICLA design process where users (creators of decision apps) can model decision tasks from scratch. In Section 7.4 we provide an overview of the intelligent management of already created decision apps. In Section 7.5 we report the results of an empirical study which focused on (1) usability aspects and (2) possibilities to counteract serial position (primacy/recency) effects in the evaluation of decision alternatives. Finally, we discuss issues for future work and conclude the chapter.

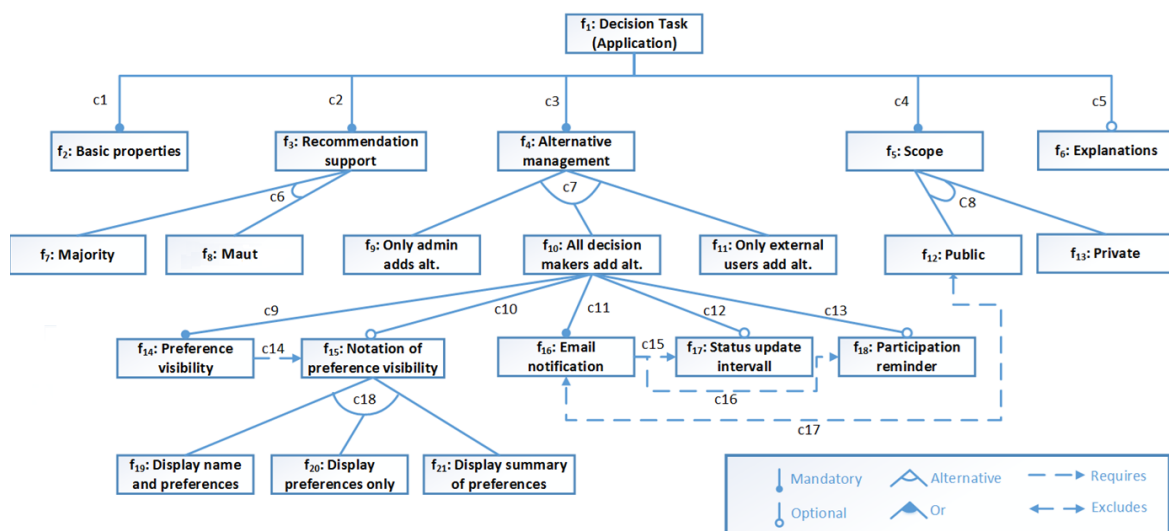


Figure 7.1.: Fragment of the CHOICLA feature model. f_i is used as abbreviation for the individual features, for example, f_3 is the short notation for feature *Recommendation Support*.

Figure 7.2.: Example set of parameters which can be selected when creating a new CHOICLA decision app.

7.3. Choicla Environment

7.3.1. Configurability of Decision Apps

Since decision scenarios differ in terms of their process design, a variety of parameters (features) is needed to make decision processes configurable (see Figure 7.1). CHOICLA offers a variety of features that represent user interface elements (and corresponding functionalities) that can be included in a decision process. For example, the creator of a new CHOICLA decision app can select (set) the feature f_{21} which indicates that in the new decision app preferences of other users will be shown in terms of a basic summary (individual user preferences will not be displayed).

In order to make decision processes (and corresponding user interface elements) configurable in an intelligent fashion, a configuration model is needed that expresses the set of features and their relationships (Benavides et al. (2013); Stumptner (1997)). This configuration model is an integrative part of CHOICLA. An example of such a relationship is the *requires* constraint between the features f_{14} and f_{15} expressing the fact that the selection of the first one also requires the selection of the latter. An example fragment of the CHOICLA feature model is depicted in Figure 7.1. We will not discuss this feature model in detail but rather explain major properties on the basis of selected features.³ An example screenshot of the CHOICLA user interface for defining the parameters of a decision app is depicted in Figure 7.2.

³More details on the CHOICLA feature model can be found in Stettinger et al. (2014).

CHOICLA includes different types of *group recommendation heuristics* that can be selected to be used within the scope of a group decision process. Figure 7.1 includes the heuristics *Majority* and *Maut* (see Dyer (2005)). The first one is a basic implementation of majority voting, the second one implements *Multi Attribute Utility Theory* (MAUT) for group-based settings. Both approaches to aggregate group preferences are discussed in Paragraph *Recommendation Support*. Solution alternatives can be managed in different ways, for example, if feature f_9 has been selected, only the creator of the corresponding CHOICLA decision app is allowed to add decision alternatives. The scope of a decision app (see also Section *Choicla Decision Apps*) can be *public* or *private*. If it is private, the app cannot be reused by other other users, otherwise it can be reused.

7.3.2. Recommendation Support

CHOICLA includes different group recommendation heuristics (e.g., majority voting and group-based MAUT) that can foster consensus in group decision making (see, e.g., Felfernig et al. (2012b); Masthoff (2011)). The degree of perceived decision quality of the participants can be significantly increased by using such aggregation functions – see, for example, Masthoff (2011); Stettinger et al. (2013). In the following we exemplify group recommendation heuristics (aggregation functions) that can be selected by the creator of a decision app during the corresponding configuration process.

For the explanation of CHOICLA aggregation functions we use a working example which is related to the task of restaurant selection by a group of friends. Table 7.1 represents the individual restaurant ratings of the participants. For these ratings, group recommendations can be derived on the basis of the CHOICLA group decision heuristics (see Table 7.2).

restaurant	Martin	Rene	Philip	Stefan
Aphrodite	4	3	4	4
Zeus	5	3	5	4
Hermes	5	3	3	3
Poseidon	3	3	5	3

Table 7.1.: Examples of individual user ratings with regard to the available decision alternatives (restaurants).

Majority Voting (see Formula 7.1) determines the value (d) that a majority of the users selected as voting for a specific solution s where $eval(u, s)$ denotes the rating for solution s defined by user u . For example, the majority of votings for *Aphrodite* is 4 (see Table 7.2).

$$MAJ(s) = \max_{d \in \{1..5\}} \arg(\# \left(\bigcup_{u \in Users} eval(u, s) = d \right)) \quad (7.1)$$

Least Misery (see Formula 7.2) returns the lowest voting for solution s as group recommendation. For

solution	MAJ	LMIS	MPLS	GDIS	ENS
Aphrodite	4	3	4	4	4
Zeus	5	3	5	5	5
Hermes	3	3	5	3	3
Poseidon	3	3	5	3	3

Table 7.2.: Results of applying the aggregation functions to the individual preferences shown in Table 7.1. MAJ = Majority Voting; LMIS = Least Misery; MPLS = Most Pleasure; GDIS = Lowest Group Distance; ENS = Ensemble Voting. This example recommendation values are based on the preference information in Table 7.1.

example, the LMIS value for $s = \textit{Aphrodite}$ is 3.

$$LMIS(s) = \min\left(\bigcup_{u \in Users} eval(u, s)\right) \quad (7.2)$$

Most Pleasure (see Formula 7.3) returns the highest voting for solution s as group recommendation. For example, the MPLS value for the $s = \textit{Aphrodite}$ is 4.

$$MPLS(s) = \max\left(\bigcup_{u \in Users} eval(u, s)\right) \quad (7.3)$$

Group Distance (see Formula 7.4) returns the value d as group recommendation which causes the lowest overall change of the individual user preferences. For example, the GDIS value for $s = \textit{Aphrodite}$ is 4.

$$GDIS(s) = \text{minarg}_{(d \in \{1..5\})} \left(\sum_{u \in Users} |eval(u, s) - d| \right) \quad (7.4)$$

Finally, *Ensemble Voting* (see Formula 7.5) determines the majority of the results of the individual voting strategies $H = \{\textit{MAJ}, \textit{LMIS}, \textit{MPLS}, \textit{GDIS}\}$. For example, the ensemble-based majority voting for *Aphrodite* is 4.

$$ENS(s) = \text{maxarg}_{(d \in \{1..5\})} \left(\#\left(\bigcup_{h \in H} eval(h, s) = d\right) \right) \quad (7.5)$$

Figure 7.3 depicts a CHOICLA user interface where a 5-star rating scale can be used to evaluate alternatives. In some cases there is a need for a more detailed evaluation of decision alternatives in terms of *interest dimensions* – this is typically the case in a MAUT-based scenario (Dyer (2005)). Evaluation criteria (MAUT dimensions) differ from decision task to decision task and therefore can be modeled during the creation process of a decision app. A MAUT-based group recommendation

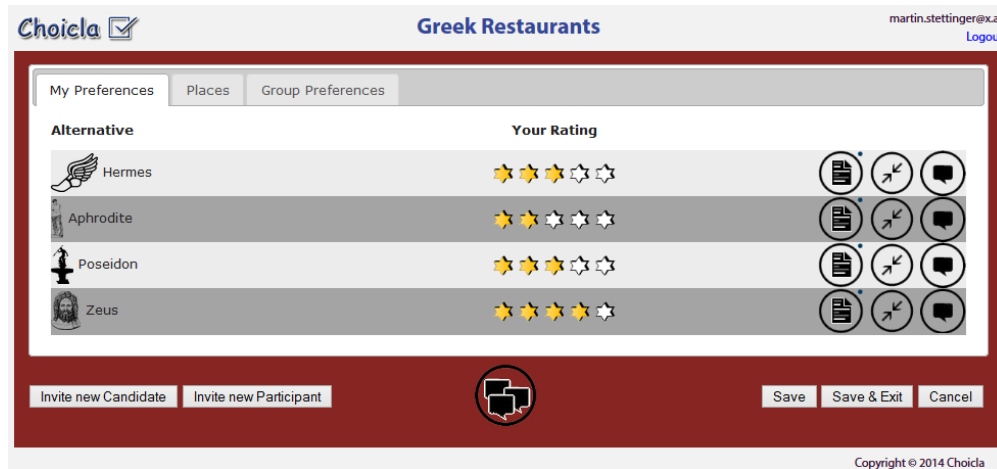


Figure 7.3.: Evaluation of the decision alternatives based on a 5-star scale.



Figure 7.4.: Dimension-based evaluation of the decision alternatives in CHOICLA.

for a specific solution s represents the sum of individual MAUT values of participants of the group decision task. A weighted average of all personal ratings of the dimensions of an alternative represents the individual MAUT value of a group member. In our current implementation the attribute values are subjective and the weights of the dimensions are fixed (predefined) which is different in a typical MAUT scenario.

$$MAUT(s) = \sum_{u \in Users} \frac{\sum_{d \in s} eval(u, d) * weight(d)}{|dimensions|} \quad (7.6)$$

If we look at the individual ratings of the dimensions of the restaurant *Zeus* in Figure 7.4 we notice the values 9, 7, 8, and 10. For simplification purposes we assume in our example that all dimensions have the same weight ($wd1 = wd2 = wd3 = wd4 = 5$). The current user's individual MAUT value of the restaurant *Zeus* is $(9*5 + 7*5 + 8*5 + 10*5)/4$ which results in 42.5. To display the MAUT-based evaluation of a solution (restaurants in our working example), utility (MAUT) values are transformed to a 5-star scale.

CHOICLA also offers a modified version of the *Group Distance* algorithm (see Formula 7.8). This modified version of the *Group Distance* algorithm (GD') takes into account past decision outcomes. A user-weighting $w(u)$ is used to ensure that users whose individual ratings often differ from related decisions (represented by $distance(u)$), are privileged. This user-weighting $w(u)$ considers the total sum of all the distances between the individual ratings of a user to the "final decision" in the particular decision situation (see Formula 7.7). If a history of past decision outcomes is available, the recommendation is influenced by these user-weightings.

$$w(u) = \sum distance(u) + 1 \quad (7.7)$$

$$GD'(s) = \min_{d \in \{1..5\}} \left(\sum_{u \in Users} w(u) * |eval(u, s) - d| \right) \quad (7.8)$$

7.3.3. Decision Alternatives

In CHOICLA, there are different ways in which decision alternatives can be entered into the system. (1) *only the creator* of a decision task can add/modify decision alternatives – this is needed if a person is interested in the opinions of his/her friends about the given alternatives (e.g., alternative candidates for the next digital camera). Another example are so-called "Micro-Polls" where the creator is only interested in knowing the preference distribution of a large group of users. (2) *all decision makers* are allowed to add decision alternatives – a typical example of such a scenario is the group-based decision regarding a restaurant or a hotel Jameson (2004). (3) *only external users* are allowed to add alternatives, i.e., users who do not participate in the decision process. For example, job applicants should be able to add a bundle consisting of the application documents as decision alternative to a (personnel) decision task (the application itself is interpreted as a new alternative - see Stettinger and Felfernig (2014)). Another related example is the selection of next year's conference location where proposers submit their material in a similar fashion.

7.3.4. Evaluation of Decision Alternatives

As shown in Figure 7.4 we used *Ambience, Price, Quality* and *Location* as dimensions in our working example. The stars in Figure 7.4 show the average evaluation based on the values given for the

different dimensions of the decision alternatives. To keep the screen in Figure 7.4 comprehensible, per default only the accumulated ratings of the decision alternatives are shown – the detailed evaluation of the dimensions is hidden and only displayed if users click on the corresponding alternative. The tab *Places* contains, if available, the geographical information of the decision alternatives. The third tab *Group Preferences* only shows up if a (predefined) threshold in terms of the number of participations is reached. It displays the actual group recommendation based on the selected recommendation heuristic (see Section 7.3) in terms of a bar chart. This threshold prevents the participants from statistical inferences to the individual user preferences, especially if no complete preference visibility is wanted at all. Figure 7.5 visualizes the current group recommendation.

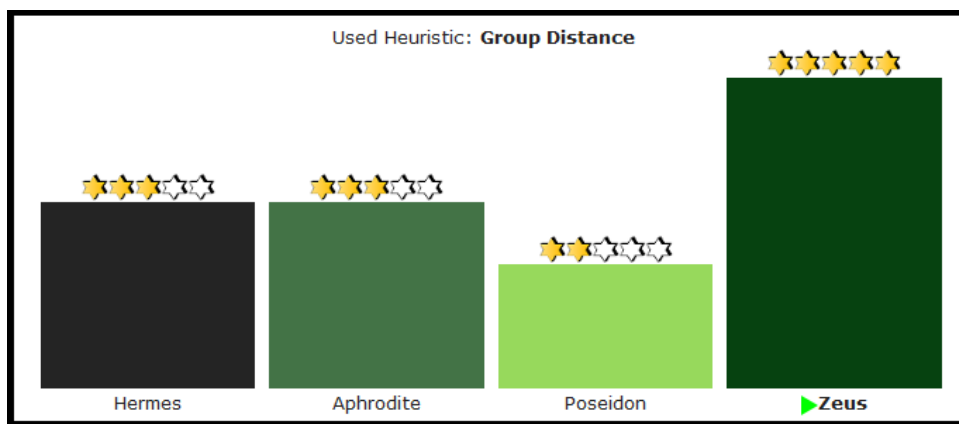


Figure 7.5.: Actual group recommendation based on the *Group Distance* recommendation function (see Formula 7.4). The group recommendation is highlighted with bold font as well as an green arrow.

7.3.5. Explanations

To increase the trust of the participants in the outcome of a decision process, explanations are indispensable (Felfernig et al. (2006)). During the design of a CHOICLA decision app explanations can be selected as a feature (see Figure 7.1).

CHOICLA supports explanations by allowing the creator of a decision task to attach argumentations as to why an alternative has been selected as group decision. If this feature is selected, the creator has to enter an explanatory text, if not, the entering of such a text remains just an option. If the creator selects the *recommended alternative* as the "final decision", CHOICLA generates an explanation according to the selected recommendation function. For example, if *majority voting* is selected, CHOICLA generates the following explanatory text for the group recommendation *Zeus*: *5-star evaluations were selected more often than other possible evaluations.*

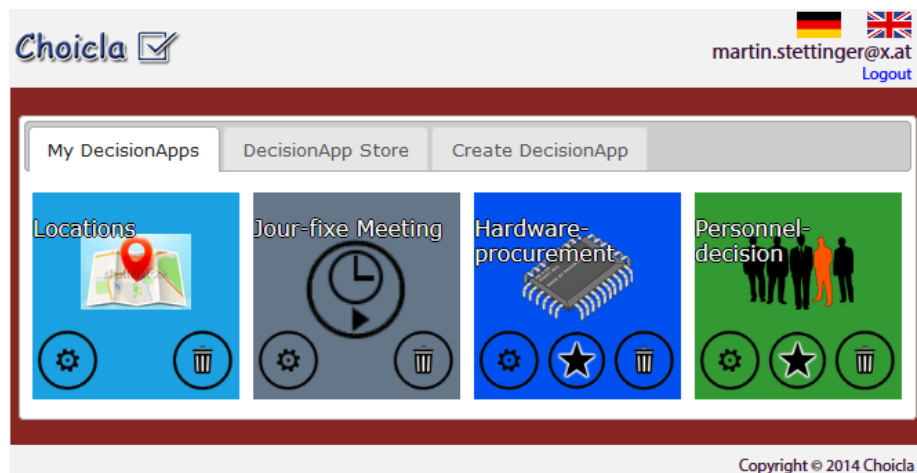


Figure 7.6.: CHOICLA: personal home screen of an registered user. In this setting, the user has installed four different CHOICLA decision apps which are the selection of a location, a meeting time for a jour-fixe meeting, new hardware as well as personnel decisions.

7.3.6. Preference Visibility

The insight to the individual preferences of all participants involved in a decision process can have a significant impact on decision quality (see Jameson (2004) and Nunamaker et al. (1991)). There exist decision tasks where the detailed insight into the preferences of all participants is an advantage and of course others where an opposite effect can occur. If, for example, the decision task deals with finding an appointment for a management meeting it is essential to find a date where all heads of different departments can attend the meeting and therefore it is important to know the individual preferences of the participants.

In some decision scenarios full preference visibility can lead to disadvantages for some participants but some kind of transparency of the preferences is helpful to achieve a reasonable decision. In such cases CHOICLA can support the decision makers by offering the possibility to display solely a summary of all given preferences regarding an alternative. A summary prevents all participants from statistical inferences but still can help participants who are unsure about their preferences. All the mentioned features (see Figure 7.1) can be configured during the modeling process of a decision app. The interface of a CHOICLA app is automatically generated conform the selected features (see Figure 7.6).

7.4. Choicla Decision Apps

After all features relevant for a new CHOICLA decision app have been selected, the corresponding decision app can be generated automatically. Figure 7.6 depicts the personal home screen of a CHOICLA

user who created four group decision apps (selecting a location, determining an appointment, selecting new hardware, and deciding about new employees). Within each of these apps a user can create new decision tasks (instances) and invite (potential) group members (participants) via email. After a participant has accepted the invitation, the decision app also shows up on the personal home screen of the participant.

The tab *DecisionApp Store* (see Figure 7.6) contains all publicly available decision apps. Decision apps can be searched and directly installed to the individual home screen. This mechanism can save a lot of effort for very common decision tasks because the creator can reuse/modify an already created decision app instead of creating a new one from scratch. This kind of reuse makes the interaction more intuitive and can also reduce barriers for using CHOICLA. The third tab *Create DecisionApp* offers the possibility to create a completely new decision app from scratch.

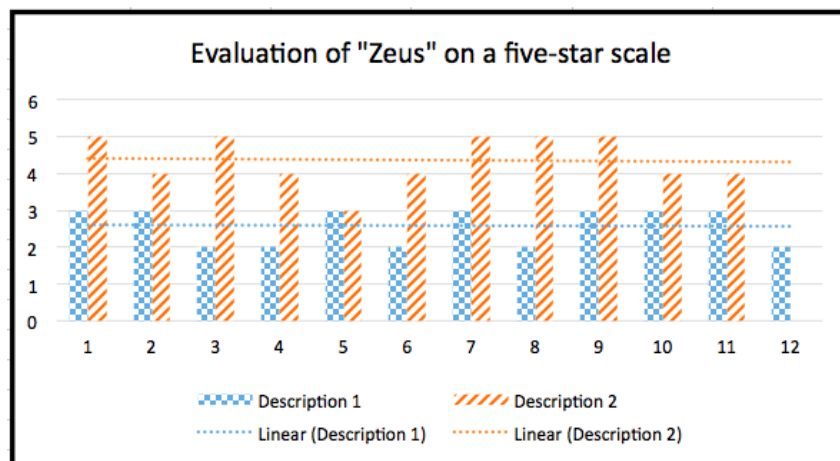


Figure 7.7.: Evaluations of the decision alternative *Zeus* on basis of a 5-star scale. The dotted lines show the average voting value of the alternative. **Description 1** states the negative aspects at the very beginning and the end (negative salient) and **Description 2** states the positive aspects at the very beginning and the end (positive salient).

7.5. User Study

We conducted a user study with the goals (1) to evaluate the usability of the current version of CHOICLA and (2) to investigate whether specific evaluation methods for decision alternatives can counteract serial position (primacy/recency) effects. In this context, the term *item* refers to individual arguments/explanations in descriptions (the difference between descriptions shown to user is the ordering of argumentations, i.e., the same individual argumentations were included in both descriptions). In this study we made CHOICLA accessible to N=44 persons outside our university who were interested in applying a group decision support software but were not aware of the fact that the system has been developed at our university. The participants (34% female, 66% male) had no computer

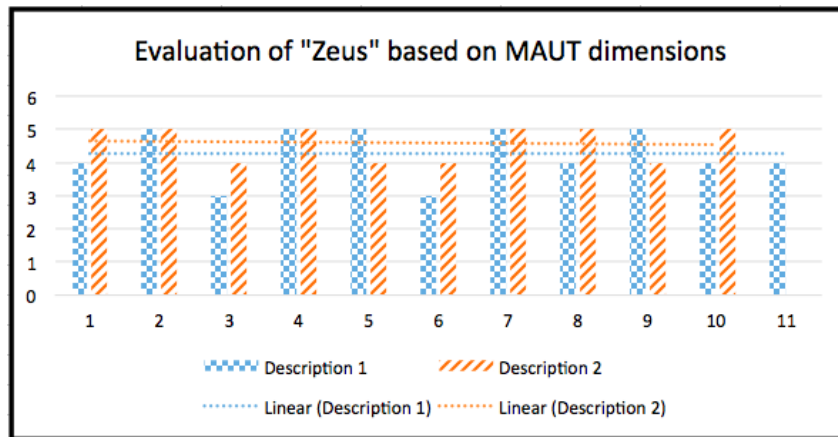


Figure 7.8.: Evaluations (scale: [1..5]) of the decision alternative *Zeus* on basis of MAUT dimensions. The dotted lines show the average voting value of the alternative. **Description 1** states the negative aspects at the very beginning and the end (negative salient) and **Description 2** states the positive aspects at the very beginning and the end (positive salient).

science oriented technical background knowledge.

The participants of the study were organized into groups of 5–6 persons who had the task of evaluating restaurants they would like to visit for a dinner. For this study, the restaurants were made anonymous, i.e., the names used in our working example (Hermes, Aphrodite, Poseidon, and Zeus) were not disclosed. N=23 participants (groups 1–4: 3 groups with 6 participants and 1 group with 5 participants) had to articulate their preferences on the basis of a 5-star scale, the remaining participants (N=21, groups 5–8: 1 group with 6 and 3 groups with 5 participants) rated the restaurants with regard to the predefined MAUT dimensions (*Ambience, Price, Quality and Location*). We anonymized all restaurants in terms of their name and also made no statement on the type of food (for example "italian", "asian", "steak", ...) to prevent a bias triggered by favorite food tastes. Our major goal was to figure out whether the evaluation based on MAUT (Dyer (2005)) can counteract primacy/recency effects (if such effects exist). Before evaluating an alternative (restaurant), each participant had to read the corresponding description which focused on the four mentioned interest dimensions (see Figure 7.4).

solution	5-star scale	MAUT dimensions
groups 1–2	ns (Description 1)	-
groups 3–4	ps (Description 2)	-
groups 5–6	-	ns (Description 1)
groups 7–8	-	ps (Description 2)

Table 7.3.: Study setting: assignment of groups to different types of preference definition support (5-star scale vs. MAUT dimensions) and descriptions of alternatives (*ns*: negative salient = *Description 1* vs. *ps*: positive salient = *Description 2*).

For each restaurant, two descriptions were available which only differ in terms of the ordering of individual argumentations: (1) the *negative salient* description where negative properties of the alternative were positioned at the beginning and the end of the description (*Description 1*) and (2) the *positive salient* description where positive properties of the alternative were positioned at the beginning and the end of the description (*Description 2* – see Table 7.3).

An example of a positive salient description (restaurant *Zeus*) is depicted in Table 7.4, Table 7.5 shows the corresponding negative salient description. With the help of these alternative descriptions we wanted to figure out whether primacy/recency effects exist when evaluating decision alternatives. We hypothesized that the higher effort which comes with an evaluation using MAUT helps to counteract primacy/recency effects. This leads to better decision outcomes because biases such as the primacy/recency effects can be significantly reduced if evaluations are performed more accurately.

Positive salient description (Zeus)
Breathtaking view over the city center. Easily accessible via the highway, plenty of parking opportunities. Terrific ambiente with an open-view kitchen. Side dishes and sauces can be freely customized. Friendly service. Upper price segment. Food quality does not justify the price. Personel leaves the impression of unkemptness. Poor cleanliness. Elegant furniture and a good place for relaxing.

Table 7.4.: Positive salient description (= *Description 2*) of a restaurant.

Negative salient description (Zeus)
Upper price segment. Food quality does not justify the price. Breathtaking view over the city center. Easily accessible via the highway, plenty of parking opportunities. Terrific ambiente with an open-view kitchen. Side dishes and sauces can be freely customized. Friendly service. Elegant furniture and a good place for relaxing. Personel leaves the impression of unkemptness. Poor cleanliness.

Table 7.5.: Negative salient description (= *Description 1*) of a restaurant.

As shown in Figure 7.5, every decision alternative is represented by a bar. To make an easy comparison to the individual preferences possible, the group recommendation displays the alternatives in the same order as the individual preferences (see Figure 7.3). The bars of the chart in Figure 7.5 contain also the individual rating information of all participants (displayed by moving the cursor over a corresponding bar). This information is only available if the preference visibility feature has been

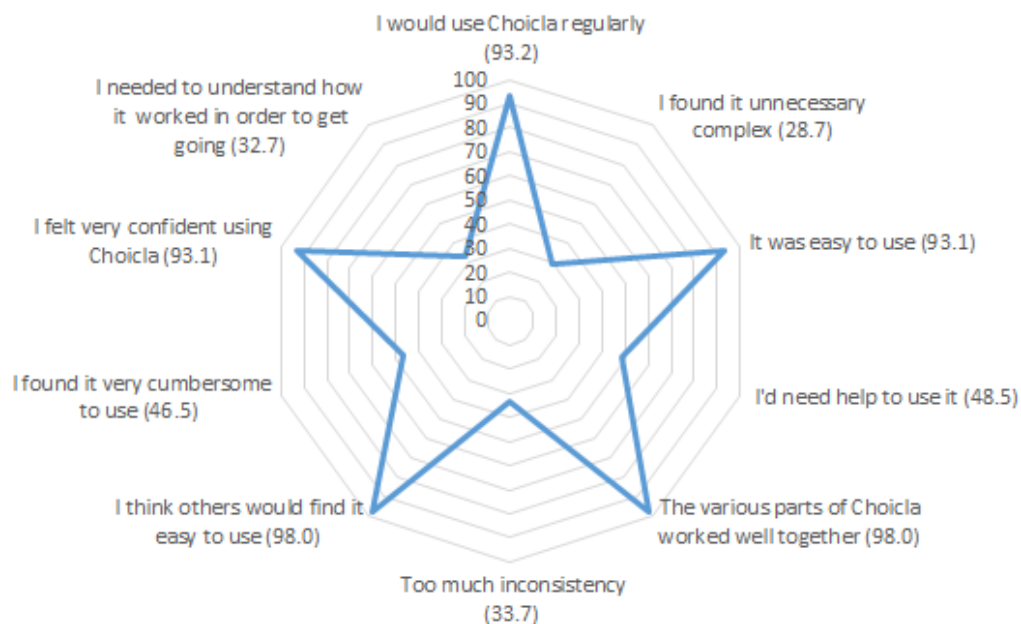


Figure 7.9.: Results of the SUS-based usability study of the second group (MAUT-based evaluation of alternatives) (N=21).

set correspondingly when creating the decision app.

Results of the User Study The study shows that the restaurants with negative aspects at the beginning and at the end of the description are rated lower than those with positive aspects at the beginning and the end of the description. This was the case when 5-star ratings were used (on an average 1.5 stars less – see Figure 7.7). The outcome of a two-sample t-test confirms that the two sample sets have different corresponding mean evaluations ($p < 0.05$). With 5-star rating scales, in 83.2% of the cases restaurants with a *positive salient* description were chosen.

If MAUT dimensions were used to evaluate the alternatives there was no significant difference between the two versions in terms of the average evaluation (see also Figure 7.8). In this case 53.6% of the participants selected a restaurant with with a *positive salient* description (46.4% selected a restaurant with a *negative salient* description).

Consequently, a MAUT-based evaluation of the decision alternatives can counteract serial position (primacy/recency) effects. Figures 7.7 and 7.8 show the results of one decision alternative (*Zeus*) based on a 5-star scale (Figure 7.7) and on a MAUT dimension evaluation (Figure 7.8).

After the participants finished their evaluation we asked them whether or not they felt comfortable during the evaluation process on the basis of the systems usability scale (SUS) (Bangor et al. (2008)). This is an important information for us because the MAUT-based evaluation needs more effort and time because all dimensions have to be evaluated separately (see Figure 7.4).

The usability feedback was very positive and provides a good motivation to continue our work on the extension and improvement of the CHOICLA functionalities. A detailed description of the results and the questionnaire for the evaluation can be found in Figure 7.9. We did not detect significant differences in the SUS results between the 5-star version and the MAUT-based version. The results indicate that participants spend more time for evaluating alternatives and thus are less susceptible to serial position effects.

7.6. Conclusions and Future Work

In this chapter we gave an overview of the CHOICLA environment which supports different types of group decision scenarios. Decision apps can be defined on the basis of a feature model that supports the flexible configuration (definition) of decision apps. Within the scope of a user study we detected primacy/recency effects in the evaluation of decision alternatives and figured out that a MAUT-based item evaluation approach can help to counteract these effects. Our future work will include the development of group decision technologies for complex products and services, the development of further group decision heuristics which especially focus on fostering consensus among group members, and the investigation of further decision biases that play a role in the context of group decision making.

Counteracting Anchoring Effects in Group Decision Making

This chapter is based on the results documented in Stettinger et al. (2015a). The author of this thesis provided major parts of this chapter in terms of literature research, de-biasing strategies as well as the user study and wrote major parts of this chapter. For this work we received the James Chen Best Student Paper Award on the 23rd Conference on User Modelling, Adaptation and Personalization (UMAP 2015).

8.1. Abstract

Similar to single user decisions, group decisions can be affected by decision biases. In this chapter we analyze anchoring effects as a specific type of decision bias in the context of group decision scenarios. On the basis of the results of a user study in the domain of software requirements prioritization we discuss results regarding the optimal time when preference information of other users should be disclosed to the current user. Furthermore, we show that explanations can increase the satisfaction of group members with various aspects of a group decision process (e.g., satisfaction with the decision and decision support quality).

8.2. Introduction

Many decisions in everyday life occur in the context of groups, for example, a decision regarding the restaurant to choose for a dinner with friends or a decision regarding the next years' conference or workshop location. A major objective of the CHOICLA¹ group decision support environment is to

¹www.choicla.com.

support different types of group decision scenarios in an efficient fashion. CHOICLA includes functionalities that determine recommendations on the basis of individual preferences of group members. When dealing with group decisions, one has to cope with different types of *decision biases* which can deteriorate decision quality. We will first provide a short overview of such biases and then focus on the aspect of how to counteract anchoring effects in group decision making. For a more detailed overview of such biases we refer to Felfernig (2014).

Serial position effects occur in situations where items at the beginning and the end of a list are evaluated more often (behavioural aspect) and also recalled (cognitive aspect) more often (Felfernig et al. (2007a); Murphy et al. (2012)) than items in the middle of a list. Such items can be argumentations in product descriptions (Stettinger et al. (2015b)), products and their attributes (Felfernig et al. (2007a)), and lists of links (Murphy et al. (2012)). Such effects can occur independent of the popularity of an attribute or item, for example, item properties presented at the beginning and the end of a recommendation dialog are recalled more often independent of their popularity (Felfernig et al. (2007a)). A possibility to counteract serial position effects in group-based recommendation is to change the preference acquisition interface, for example, from a star-based rating to a utility-based rating (items are evaluated with regard to a predefined set of interest dimensions) which encourages users to analyze item descriptions in more detail (Stettinger et al. (2015b)).

Decoy effects cause shifts in preference construction since decisions are taken depending on the context in which alternatives are presented to the user (Tversky and Simonson (1993)). For example, including a completely inferior alternative (e.g., with the lowest overall utility compared to all other alternatives in a list of recommended items) can change a user's evaluation of the remaining items in the list. In the context of recommenders, such effects have been analyzed by Teppan et al. (Teppan and Felfernig (2012)) who showed the existence of decoy effects on the basis of real-world financial services datasets. Counteracting decoy effects can be based on predictive models that predict decoy items which could be eliminated from a result set (Teppan and Felfernig (2012)).

Explanations can have a significant impact on the way that items are perceived/evaluated and – as a consequence – on the corresponding decision. Thus, explanations play an important role in recommender systems (Gkika and Lekakos (2014); Tintarev and Masthoff (2007)), for example, a digital camera will be purchased or not, a movie will be watched or not, a car feature will we included or not, a project proposal will be accepted or not, and a software requirement will be regarded as important or not. Stettinger et al. (Stettinger et al. (2015b)) analyze the impact of argument orderings of item explanations on the decision outcome, Felfernig et al. (Felfernig et al. (2006)) and Pu et al. (Pu and Chen (2007)) show the (positive) influence of explanations on a user's trust in recommender systems, and Herlocker et al. (Herlocker et al. (2000)) discuss different explanation-relevant dimensions in recommender systems where beside *justification*, *user involvement*, and *education*, *acceptance* is mentioned as a major relevant factor.

Anchoring effects cause decisions which are influenced by the group member who first articulated

his/her preferences (Adomavicius et al. (2011); Jacowitz and Kahneman (1995)) – these results in the context of decision support environments are confirmed by social-psychological studies that point out the relationship between decision quality and the visibility of individual preferences for other group members (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). Interestingly, hidden preferences in early phases of group decision scenarios can increase the overall amount of information exchange between group members and the higher the amount of information exchange the higher the quality of the decision outcome. In collaborative filtering scenarios, anchoring effects can be triggered by disclosing, for example, the average rating of other (similar) users. An adaptation of the preference acquisition interface (e.g., a rating scale adapted from a 5-star to a binary one) can help to counteract such biases in collaborative filtering (Adomavicius et al. (2011, 2014)).

The existence of anchoring effects in group decision scenarios has also been shown in Felfernig et al. (Felfernig et al. (2012b)) who analyzed bias-induced preference shifts in the context of requirements engineering. In this scenario, the task of the project team was to make decisions regarding different technical and organizational aspects of their software project. Examples of such decisions are the way in which their software project should be evaluated and the type of technology that should be used for implementing the requirements. Masthoff and Gatt (Masthoff and Gatt (2006)) discuss algorithmic approaches to satisfaction prediction in group decision scenarios where *conformity* (judgments are influenced by the judgements already articulated by other group members) and *emotional contagion* (influence of an individual's affective state on that of other group members) are mentioned as influence factors. Compared to Masthoff and Gatt (Masthoff and Gatt (2006)), we did not analyze emotional states of group members and focused on the impacts of different degrees of judgement visibility. An analysis of intra-group dynamics in CHOICLA decision scenarios is within the scope of future work. Our major focus in this chapter is to show in which way anchoring effects can be counteracted in the context of group decision making. In this context, we focus on a *requirements prioritization scenario* where groups of students (teams) had to agree on the set of additional requirements (and their priority) they are willing to implement in their software project. In addition, we investigated the impact of explanations in group decision scenarios (explanations textually entered after a final decision has been taken). In this context we were interested on the impact that explanations can have on the overall acceptance of a group decision by individual group members.

As a basis for completing the requirements prioritization task the teams of our study used the CHOICLA group decision support environment. Example CHOICLA scenarios for industrial settings are the selection of new employees, the selection of conference locations, and the evaluation of project proposals. In the private context, CHOICLA can, for example, support the selection of a restaurant for a dinner with friends, the selection of a hotel for a holiday trip, and the selection of a cinema movie to watch with friends. In addition to CHOICLA, there exist many other group decision support environments. DOODLE² focuses primarily on the aspect of coordinating meetings and does not include additional mechanisms to determine recommendations for groups of users. Similarly, VERN (Yardi et al.

²doodle.com.

(2005)) is a tool that supports the identification of meeting times based on the idea of unconstrained democracy where individuals are enabled to freely propose alternative dates themselves. SMARTOCRACY provides support for voting scenarios in social network contexts where information from the social network is applied to rank recommendations (Rodriguez et al. (2007)). DOTMOCRACY³ deals with larger groups of users and provides a method for collecting and visualizing group preferences. The system is based on the idea of participatory decision making – it’s major outcome is a graph type visualization of the group-immanent preferences. Compared to CHOICLA, these tools focus on specific domains and do not offer the possibility for a flexible definition of domain-independent decision scenarios.

The contributions of this chapter are the following: (1) we provide a short overview of the CHOICLA group decision support environment on the basis of a working example from the area of software requirements prioritization, (2) we show (a) the existence of anchoring effects and (b) possibilities of counteracting these effects in the context of group decision making, and (3) we show that explanations in group decision scenarios can have a positive impact on the overall acceptance of group decisions. The remainder of this chapter is organized as follows. In Section 8.3 we provide an overview of the CHOICLA decision support environment. In Section 8.4 we report the results of an empirical study which focused on (a) anchoring effects within group decision scenarios and (b) the impact of explanations. The chapter is concluded with Section 8.5.

8.3. The CHOICLA Environment

Decision tasks often differ in their basic properties, for example, *decision heuristics* (Masthoff (2011)) such as *majority voting* or *least misery* should be preselected or not, *alternatives* can only be defined by the administrator (also denoted as *creator*) of a decision task (app), *preferences of other group members* should be visible (or not), and decisions should be *explained* or not (by the creator of a decision task). Due to the many existing options, decision tasks must be configured before being provided to a group of users – for details see Stettinger et al. (Stettinger et al. (2014)). An example of a definition (configuration) of a CHOICLA decision app is depicted in Figure 8.1. In this example, a group of users (stakeholders) should *decide about the priority of requirements that should be additionally implemented in a software project*. In this context, all group members are allowed to add their own alternatives (software requirements), to add additional material (links and files), and to see the preferences of other users (regarding the prioritization of requirements). Making the process design of decision tasks configurable introduces the flexibility that is needed due to the heterogeneity of decision problems. The achieved flexibility provides the basis for organizing the CHOICLA components in a kind of a software product line that is open in terms of the generation (implementation) of problem-specific decision applications.

³dotmocracy.org.

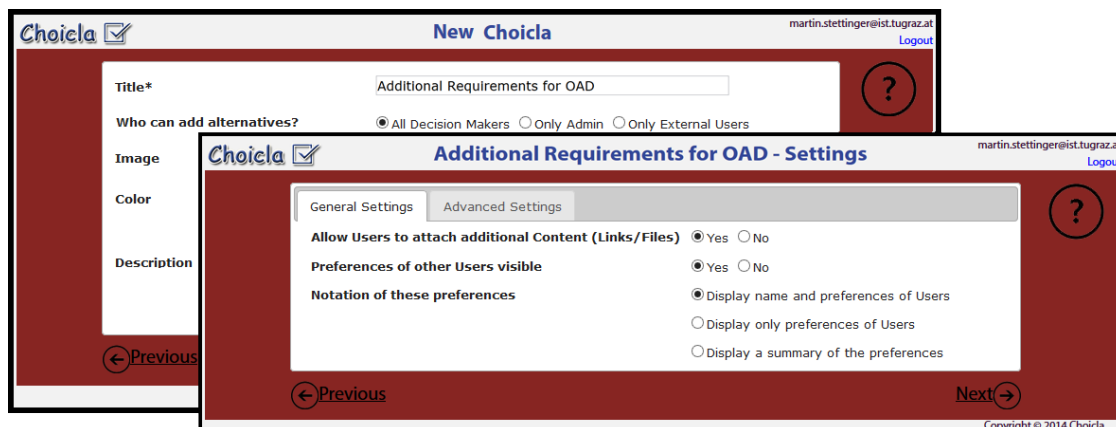


Figure 8.1.: Interface for configuring CHOICLA decision apps.

After a CHOICLA *decision app configuration* has been completed, the corresponding *decision app* is automatically generated and installed on the home screen of the decision app creator. The creator can now invite relevant users (in our case stakeholders) to participate in the decision process – this is currently possible via email. Figure 8.2 depicts examples of already configured and generated CHOICLA decision apps: *requirements prioritization* (our working example), *appointment scheduling*, *hardware procurement*, and *personnel decision*⁴.

Figure 8.2 includes two more tabs which are denoted as *DecisionApp Store* and *Create Decision-App*. The former can be used for searching and installing new decision apps (this is only possible if a decision app has been defined as *public* and therefore been made *reusable* by the app creator), the latter can be used for creating (configuring) your own decision app (for details see Stettinger et al. (Stettinger et al. (2014))). CHOICLA decision apps can entail an arbitrary number of decision *instances*, for example, if a requirements prioritization decision has to be taken for a new project or a new set of requirements, the same decision app can be used by simply creating a new instance inside the given decision app. Also after completion of the decision process, each individual instance of the decision app is accessible in a *decision history* (documentation).

8.4. User Study

As already mentioned in Section 8.2, our major goal is to analyze anchoring effects in group decision scenarios. In a requirements prioritization scenario (team members had to select *additional requirements they had to implement within the scope of their project*) we wanted to investigate the existence of anchoring effects and also to figure out when to best disclose individual preferences (evaluations) to other users (in our case stakeholders). In this context we were also interested in the impact of preference invisibility on the degree of information exchange between individual stakeholders. Finally,

⁴The CHOICLA *personnel decision app* is already applied by an Austrian university.

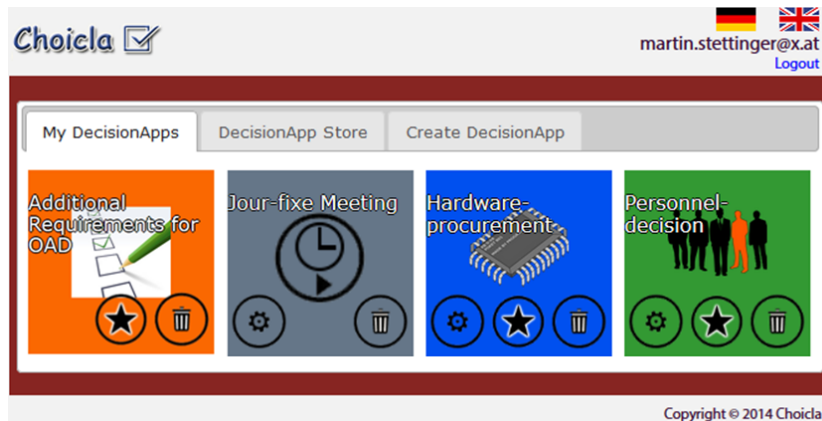


Figure 8.2.: Examples of defined (configured) and generated CHOICLA decision apps.

we wanted to investigate factors such as the impact of the existence of explanations for group decisions on the degree of satisfaction with the decision support and the perceived understandability of the group decision. In the remainder of this chapter we will first present the CHOICLA decision app generated for the purposes of requirements prioritization (software requirements for an online game) and then discuss the design of our user study and the corresponding study results in detail.

The generated requirements prioritization decision app supports the prioritization of requirements on the basis of a multi-utility based evaluation scheme (Dyer (2005)). Team members (subjects of the study) were enabled to evaluate each requirement with regard to the dimensions *Risk*, *Effort*, and *Profit*. Note that such dimensions are freely definable in CHOICLA if a MAUT-based aggregation function (group recommendation heuristic) has been selected. An example of the evaluation of the requirement *Change Background* is depicted in Figure 8.3.

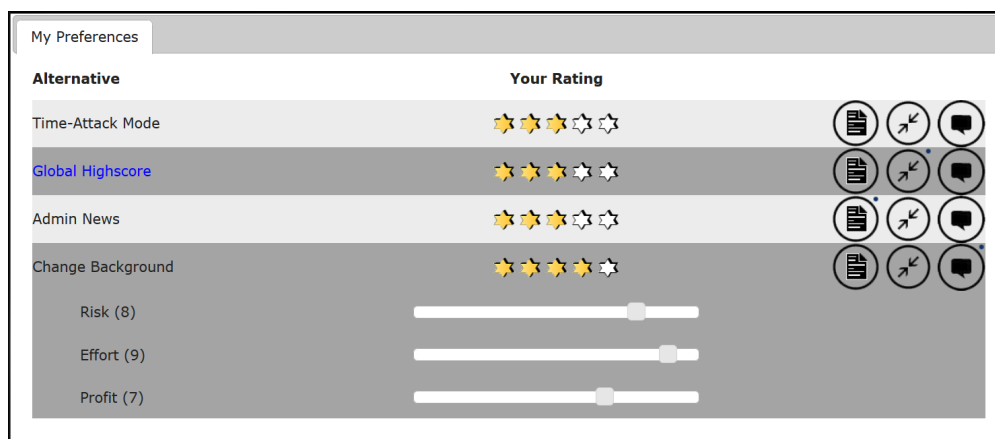


Figure 8.3.: Evaluation interface of the requirements in CHOICLA. Participants can enter their ratings by selecting a value for the dimensions *Risk*, *Effort*, and *Profit*.

This requirement is linked to a detailed textual description – in our case, the background style

should be changeable in an online game. Note that in utility-based scenarios CHOICLA supports a group-based MAUT approach, where individual ratings defined for interest dimensions are aggregated using arithmetic mean and then added up (for details see Stettinger et al. (Stettinger et al. (2015b))). The utility of each individual alternative (*requirement*) is then transposed to a five-star rating scale as depicted in Figure 8.3. Since the goal of our study was to investigate anchoring effects in the context of group decision scenarios, the *visibility of the preferences of other group members* was one of the major variation points in the user study.

Figure 8.3 includes a CHOICLA user interface version where the preferences of other users are not disclosed to the current user. In contrast, Figure 8.4 depicts an interface version where the preferences (priorities) of the individual stakeholders are visible (the height of each bar corresponds to the corresponding MAUT value (Dyer (2005); Stettinger et al. (2015b)) of a requirement, individual preferences are visible when moving the mouse pointer over the corresponding bar). If all stakeholders have articulated their requirements, the creator of a decision app can close the decision process, i.e., no further changes/adaptations of the individual user preferences are possible from that time on. Closing the decision process means that one or more options are selected by the administrator and these alternatives altogether then represent the final decision. The selected alternatives may not correspond with the alternatives proposed by the aggregation heuristic (in our case MAUT).

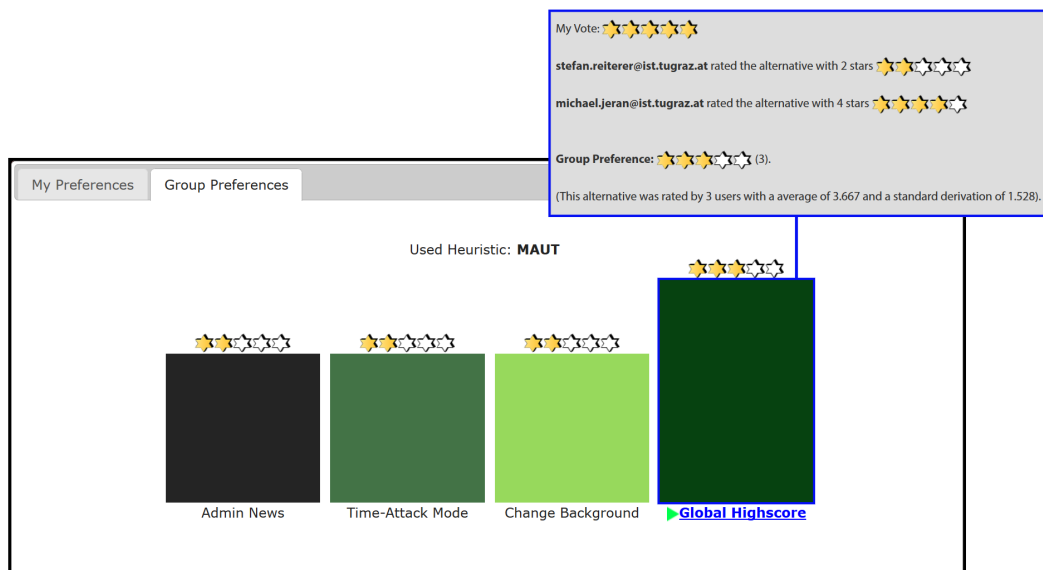


Figure 8.4.: Group recommendation (on the basis of MAUT values) for the prioritization of requirements within CHOICLA. Preferences of individual stakeholders are disclosed when moving the mouse pointer over the bar.

In our working example, the creator of the decision app selected only one requirement (*Global Highscore*) as an additional requirement to be implemented in the project (see Figure 8.5). In this case, the creator follows the group recommendation and also explains the reason for the final decision. Note that the possibility of explaining final decisions is another major variation point in the user study,

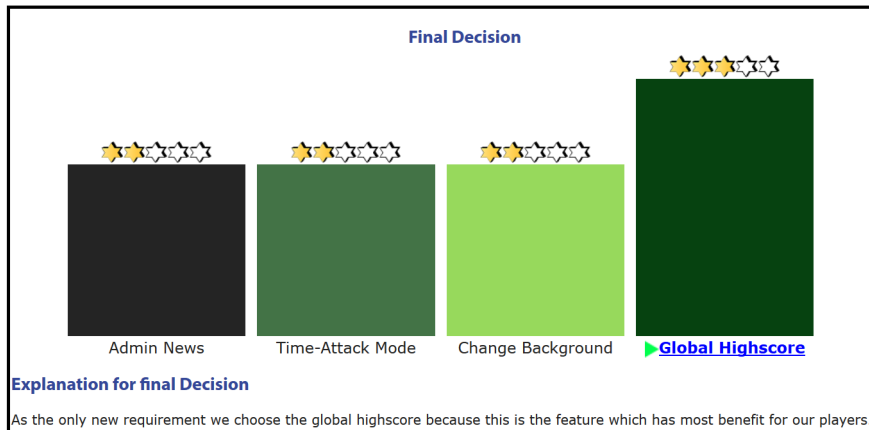


Figure 8.5.: Representation of a final decision in CHOICLA in terms of a bar chart.

i.e., some versions included this option, some versions not.

We conducted a user study with computer science students at the Graz University of Technology (N=229 participants, 16% female, and 84% male) who took a course on *object oriented analysis and design*. Students formed software teams with 5–6 participants (in total 45 teams) who had then to implement an online game environment. Each team had to develop the same set of basic requirements but could choose 5 out of a set of 10 additional requirements using CHOICLA as the sole decision and communication platform.⁵ The 45 software teams (groups) were assigned to different categories as follows (see also Table 8.1). First, 23 groups were confronted with a CHOICLA user interface which enforced the explanation of final decisions, the remaining 22 groups had the option to explain their decisions but this was not mandatory. Second, the individual CHOICLA versions differed in terms as of when individual preferences are made public to all group members (after one, two, three, or all group member(s) has(have) articulated his/her(their) preferences).

explanation mandatory				explanation not mandatory			
after 1.	after 2.	after 3.	after all	after 1.	after 2.	after 3.	after all
6 groups	6 groups	5 groups	6 groups	6 groups	6 groups	5 groups	5 groups

Table 8.1.: Assignment of versions to groups in the user study, for example, "explanation mandatory+after 1." denotes a CHOICLA version with mandatory explanations and individual preferences were disclosed after one group member defined his/her preferences.

The hypotheses as input for our user study were the following. First, we assumed that anchoring effects occur especially in cases where preference information of individual users is disclosed although this information has not been provided by all group members, i.e., the lower the number of completed preference definitions the higher the probability of anchoring effects (H1).

Second, we assumed that the best time to disclose individual preferences is a situation where each

⁵Due to space limitations we limited our example set to 3 requirements.

group member has already articulated his/her requirements (H2). This strategy should lead to the best results regarding (a) *the satisfaction with the final group decision* as well as (b) *the perceived degree of decision support*, (c) *perceived understandability of the final group decision*, and (d) *consideration of one's personal preferences*. In our study, data to answer (a)–(d) were collected in a post-decision questionnaire. The rating scale for questions (a)–(b) was [very satisfied (5) .. very unsatisfied (1)], for question (c) it was [understood immediately (5) .. no chance to understand without asking a couple of times (1)], and for (d) it was [excellent (5) .. very bad (1)].

In the line of decision psychological experiments (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)) we assume that the later individual preferences are disclosed the higher will be the number of comments in the CHOICLA forum (H3). A higher degree of information exchange also has a direct positive impact on decision quality – see also (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). Hypothesis H3 is related to the fact that groups tend to focus on the preferences of other group members if this information is available but otherwise focus on information exchange to gain a better understanding of the problem setting (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)).

With hypothesis H4 we want to express the assumption that the explanation of a final decision can increase (a) *the satisfaction with the final group decision* as well as (b) *the perceived degree of decision support*, (c) *perceived understandability of the final group decision*, and (d) *consideration of one's personal preferences*.

The results of our user study were the following. We can confirm hypothesis H1, i.e., anchoring effects are triggered by an earlier disclosure of preference information to other group members. In this context, we analyzed the standard deviations of the individual user ratings (i.e., we used the standard deviation of ratings as an indicator of anchoring effects) depending on the time of the disclosure of the ratings (preferences) of individual group members. Figure 8.6 depicts the standard deviations of user ratings depending on the time of preference disclosure; standard deviations increase monotonously in the number of anonymously articulated preferences. The series of standard deviations related to versions *after 1.* and *after 2.* (and above) significantly differ in terms of their mean values ($p < 0.05$, t-test).

We can also confirm hypothesis H2. The later the time of preference disclosure (the more group members have articulated their preferences without viewing the preferences of other users), the higher the evaluation with regard to the dimensions (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one's personal preferences. Figure 8.7 depicts, for example, the user evaluations with regard to (a) satisfaction with final group decision and (b) perceived degree of decision support. The average evaluations of all dimensions, i.e., (a) .. (d), are depicted in Table 8.2.

T-tests also confirm significant user evaluation improvements with an increasing number of defined but undisclosed preferences. The average user evaluations regarding (a) and (b) related to versions

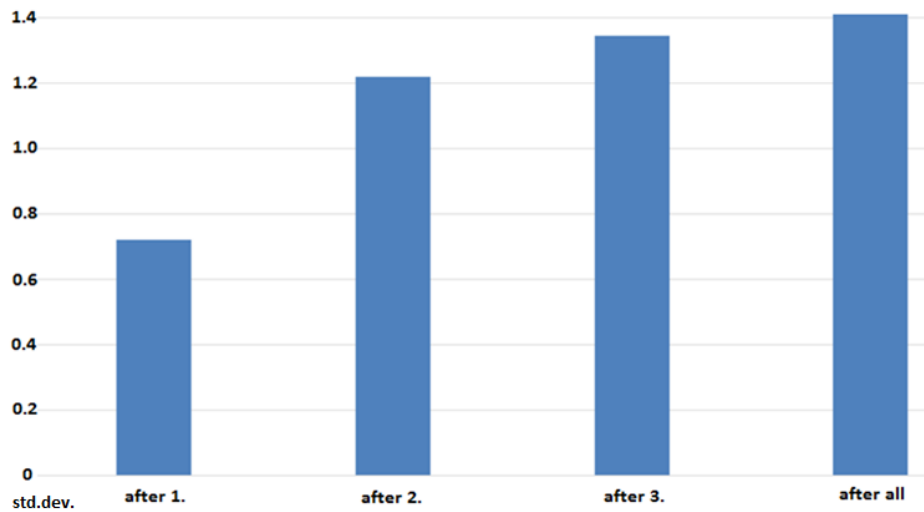


Figure 8.6.: Standard deviations of user ratings of alternatives (requirements) depending on preference disclosure time (after 1..3, or all users articulated preferences).

after 1. and *after 3.* (and above) differ in terms of their mean value ($p < 0.05$, t-test, see also Figure 8.7). Significant results ($p < 0.05$, t-test) could also be observed for average user evaluations regarding (c) and (d) related to versions *after 1.* and *after all.*

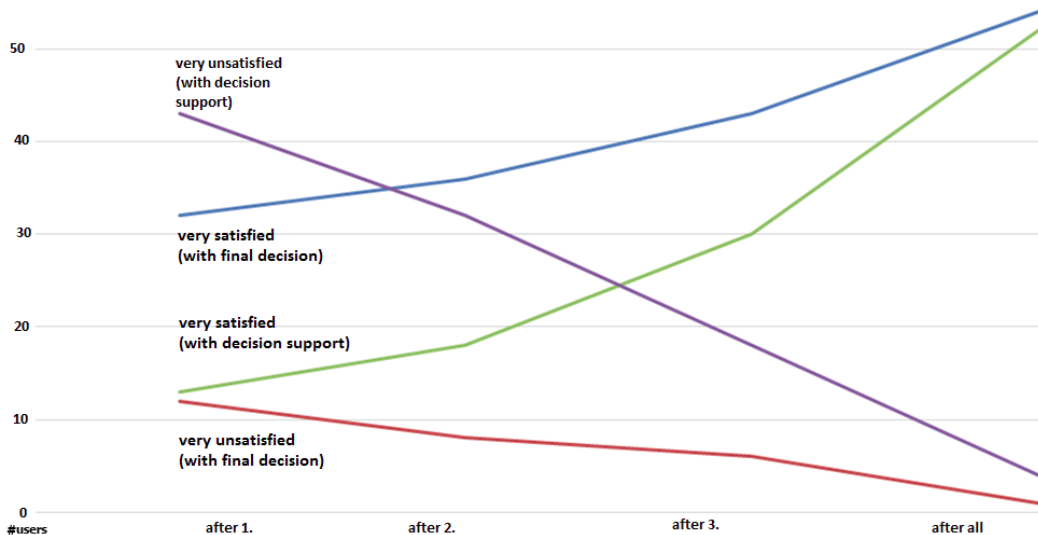


Figure 8.7.: Satisfaction with final group decision and perceived degree of decision support depending on preference disclosure time (after 1..3, or all users articulated preferences).

We can confirm hypothesis H3: the later individual preferences are disclosed to other users, the higher the amount of comments/discussions in the CHOICLA forum. The number of comments depending on the degree of already available preference definitions not disclosed to other users is shown

	All			
	after 1.	after 2.	after 3.	after all
a	2.87(1.67)	3.01(1.5)	3.31(1.19)	3.73(0.73)
b	1.75(1.77)	2.2(1.62)	2.83(1.59)	3.72(1.02)
c	3.44(1.65)	3.54(1.54)	3.79(1.22)	4.04(0.81)
d	3.02(1.77)	3.55(1.75)	3.91(1.46)	4.16(1.04)

Table 8.2.: Avg. evaluations and std.dev. regarding (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one's personal preferences.

in Figure 8.8.

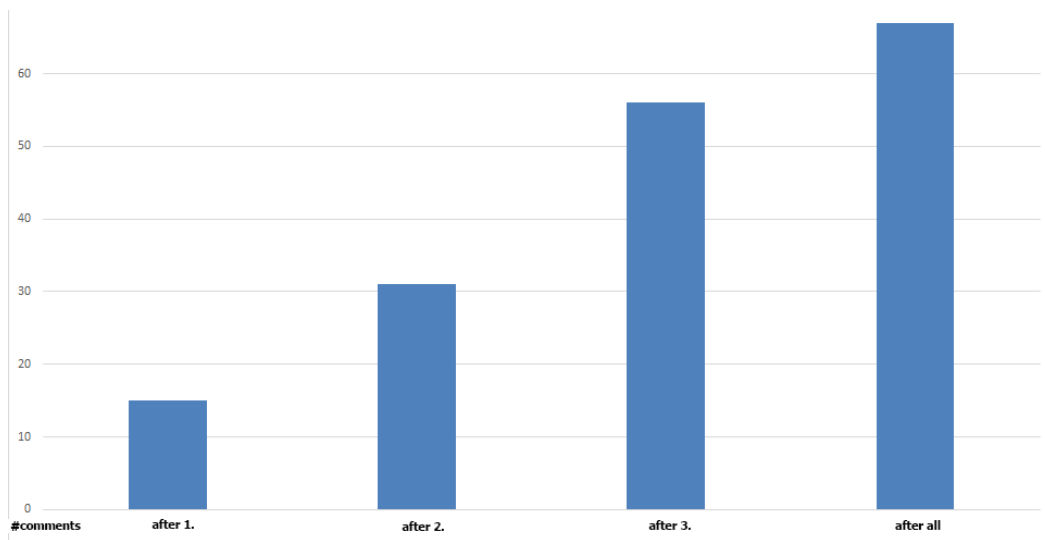


Figure 8.8.: Number of comments in the CHOICLA discussion forum depending on preference disclosure time (after 1..3, or all users articulated preferences).

Finally, we can also confirm hypothesis H4: groups with (enforced) explanation support for group decisions have significantly higher evaluations in terms of the dimensions (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one's personal preferences. This is confirmed by corresponding t-tests ($p < 0.05$) when comparing groups with and without (enforced) explanation support (average evaluations are depicted in Table 8.3).

8.5. Conclusions and Future Work

With the work presented in this chapter we have shown the existence of anchoring effects in group decision scenarios: the earlier individual user preferences are disclosed to other group members, the higher the probability of the occurrence of anchoring effects. The time of preference disclosure

	Explanations Enforced			
	after 1.	after 2.	after 3.	after all
a	3.67(1.27)	4.01(1.15)	4.31(0.89)	4.93(0.23)
b	1.95(1.66)	2.5(1.51)	3.45(1.19)	4.69(0.82)
c	3.84(1.35)	3.94(1.24)	4.29(0.92)	4.87(0.41)
d	3.62(1.37)	3.95(1.15)	4.41(0.66)	4.86(0.31)
	Explanations Not Enforced			
	after 1.	after 2.	after 3.	after all
a	2.4(1.76)	2.87(1.66)	3.17(1.47)	3.22(1.33)
b	1.63(1.94)	2.02(1.82)	2.67(1.63)	3.18(1.47)
c	3.12(1.72)	3.25(1.78)	3.57(1.58)	3.73(1.22)
d	2.88(1.96)	3.17(1.81)	3.29(1.73)	3.87(1.4)

Table 8.3.: Avg. evaluations and std.dev. regarding (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one’s personal preferences.

also has a direct impact on the perceived quality of the decision outcome and the perceived decision support. Furthermore, late preference disclosure can lead to a higher discussion intensity inside a group which can have a direct positive impact on the quality of the decision outcome. It is important to take into account these aspects in application development; especially one has to analyze the need of preference disclosure since non-disclosed preferences can help to significantly improve decision quality. The analysis of further decision biases and their impact on group decision making is within the major focus of our future work since this will help to further advance the quality of group decision support in the CHOICLA environment. With regard to anchoring effects we want to analyze in further detail the impact of different representation types of user preferences (e.g. aggregated representations vs. user-specific representations) on evaluation dimensions such as perceived decision quality and quality of decision support. Finally, we are also interested in a deeper understanding of intra-group dynamics that can potentially help to further improve the quality of group decisions.

Conclusions & Future Work

To offer decision support with the goal to make group decisions in general more efficient and increase the quality of decision outcomes it is essential to consider factors such as the quality of the recommendation algorithms, intelligent preference acquisition interfaces as well as results from latest decision psychology research. This thesis presents CHOICLA, a novel domain-independent group decision support environment in which groups of users are supported in making high quality decisions. In this chapter we reflect on our research questions and contributions as well as the limitations of the approaches. An outlook for future research concludes this chapter.

9.1. Conclusions

In the following we present a summary of the answers to the research questions we defined in Section 1.2.

Research Question Q1:

How to design a domain-independent decision support environment for groups of users?

At the time when we started focusing on the investigation of group decision support technologies we did not find any work where an integrated tool for supporting groups of users in a domain-independent fashion was presented. In Section 6.3 we present our group decision support environment called CHOICLA which advances the state-of-the-art by providing decision support for groups of users independent from the domain of the decision task. In order to evaluate the pilot version of CHOICLA we conducted a first small-scale user study where we made the software accessible to a total number of 48 persons. Section 6.5 presents the results of our empirical study where we could gain first evidence that people consider the system suitable for a wide range of applications and can imagine to apply the system in various domains.

Research Question Q2:

How to design the creation process of a group decision task as a configuration problem?

To create a group decision support environment which is independent from the domain of the decision task it is essential to offer possibilities for customizing the decision process (on the basis of features such as recommendation heuristic, preference visibility options as well as settings concerning the management of decision alternatives). In the context of this work we show how to represent the creation of a group decision task as a configuration task. In Section 5.3 we introduce a feature model that expresses the diverse features (properties) of decision tasks. By taking into account the dependencies among the features (for a detailed discussion see Section 5.4) we show how group decision tasks can be configured in the CHOICLA environment (see Section 5.5).

Research Question Q3:

How to best support groups of users in the context of personnel decisions?

First impressions, interpersonal attraction or the appearance of job applicants are often responsible for subjective evaluations which can lead to suboptimal decision outcomes due to the fact that in such cases often no concrete structure is followed and the assessment criteria are not stable over time (Kobryniewicz and Biernat (1997); Dougherty et al. (1994); Graves and Powell (1988)). In this thesis we present first steps towards a more structured and objective hiring procedure. Section 6.4.1 presents the evaluation technique integrated in the CHOICLA group decision support environment which shows the ability to counteract subjective evaluations by providing pre-defined evaluation dimensions which are tailored towards the current job position. In addition we introduce a group recommendation approach which is based on a modified MAUT (Stettinger and Felfernig (2014)) approach.

Research Question Q4:

How to achieve long term fairness in group decision tasks?

Many group decision tasks such as a decision regarding the location of next year's annual company excursion or a decision regarding which restaurant to choose for a dinner with friends, occur repetitively. Available decision support tools do not offer functionalities where past decision outcomes can be used to extract knowledge for future scenarios. In this thesis we present an intelligent approach where past decision outcomes can directly influence future group recommendations. With the help of the knowledge gained out of past decision outcomes we are able to introduce an algorithmic approach (see Section 6.3.1) which facilitates long term fairness (and thus satisfaction) among group participants.

Research Question Q5.1:**How to counteract serial position effects in group decision support environments?**

Suboptimal decision outcomes are often caused by decision biases which occur before and during the decision process. Serial position effects (primacy/recency) (Felfernig et al. (2007a); Murphy et al. (2012)) occur in situations where users have to evaluate large item sets and lead to more popularity of the items which are presented at the beginning and at the end of such item sets (Bar-Hillel (2015)). In Section 7.5 we focus on the investigation of serial position effects in descriptive texts of decision alternatives. Section 7.5 presents the results of our empirical study which clearly point out that the individual preferences are strongly influenced by the description of decision alternatives. Additionally we compared different preference acquisition interfaces (see Section 7.3.4) for evaluating the decision alternatives (star-based rating scales vs. utility-based rating scales). As key result of our user study we could introduce an intelligent utility-based preference acquisition interface with the ability to counteract serial position effects. In addition to that we also present the results of the feedback concerning the system's usability which was very positive and provided the motivation to continue our work to further improve the CHOICLA technologies (Stettinger et al. (2015b)).

Research Question Q5.2:**How to counteract anchoring effects in context of group decision scenarios?**

Anchoring effects represent another frequently occurring decision bias which leads to suboptimal decision outcomes by triggering decisions which are biased by the first person who articulated his/her preferences (Adomavicius et al. (2011); Jacowitz and Kahneman (1995)). In this thesis we present the results of our empirical study which show that anchoring effects – comparable to single user decision scenarios – also occur in the context of group decision making. In Section 8.4 we discuss the optimal time for disclosing the preferences of the other group members to counteract anchoring effects. In addition to that we also introduce the results of a user study which show that communication behaviour is also affected by the time when individual preferences are disclosed to the other group members. Less discussions are the result of presenting the individual preferences of the other group members in early stages of the decision process. A detailed discussion on the communication behaviour can be found in Section 8.4. We see our contribution as an important one towards intelligent decision support systems for groups of users for different reasons. First, we offer innovative techniques that take into account that counteracting decision biases such as anchoring effects and serial position effects can significantly increase the quality of decision outcomes. In addition to that the quality of decision outcomes can be further increased by providing techniques that trigger a higher amount of communication among the group members during the decision process.

Limitations

The usability studies as well as the developed recommendation heuristics presented in this thesis have limitations. The results of the conducted usability studies give a first impression of the perceived usability of the system but show limitations in terms of the small number of participants. The introduced recommendation heuristic for achieving long term fairness only considers the individual distance to the final decision and does not take other dimensions (e.g., individual tastes or individual domain-specific preferences) into account which is a limitation of this approach. Compared to the large number of possible decision biases (Felfernig (2014)) the small number of investigated decision biases (anchoring effects (Adomavicius et al. (2011); Jacowitz and Kahneman (1995)) and serial position effects (Felfernig et al. (2007a); Murphy et al. (2012))) represent another limitation of this thesis. Further limitations of the CHOICLA decision support environment are (1) no support for complex alternatives, (2) no support for strategic decisions as well as (3) no support for combined decisions (e.g., time and restaurant).

9.2. Future Work

One major focus for future work is the continuous extension of the CHOICLA technologies where we focus on the detailed analysis of further application domains. We focus on both (1) the design (implementation) of group decision tasks with the aim to make the creation process as simple and straightforward as possible and on (2) the resulting group decision task with the aim to raise the quality of decision outcomes and make group decisions in general more efficient. To further advance the quality of decision outcomes in the CHOICLA group decision support environment, a deeper understanding of further decision biases as well as their impact on group decision making is a key factor. In order to achieve the vision of raising group decision making to the next level, one essential future research direction is the implementation and development of further group recommendation heuristics (for already implemented group recommendation heuristics see Section 7.3.2). Another area of future research will include the development of group decision technologies for complex products and services. In the following we discuss additionally relevant topics for future research.

Further decision biases in the context of group decision scenarios

Further decision biases are induced if a system is open in the sense that there exists a functionality which allows decision makers to add new decision alternatives during the decision process. In cases where not all decision alternatives are available/known during the creation process of a decision task it is essential that new decision alternatives can be added during the decision process itself. If new decision alternatives arise during the decision process, additional decision biases can be induced due to the fact that members of a group will perceive already rated (selected) decision alternatives more

attractive than new ones (Neale et al. (2004)). Research in the field of decision psychology states that if consensus about alternatives is formed out of discussion in early stages of a decision process, this consensus is cognitive resistant to changes. It is very unlikely that by adding additional information during an ongoing decision task a different alternative will be chosen because such additional information will be adapted to already defined consensus (Lind et al. (2001)). This effect can be explained by the *dissonance theory* and the underlying *assimilating effect* (Festinger (1957)). Due to the *assimilating effect* people try to minimize psychological incongruities which are the result of adding additional information to a present perception (Neale et al. (2004)). In groups with a high cohesion this effect is intensified because in such groups the fear from exclusion is higher (see Lind et al. (2001)). To counteract such effects in future versions of the CHOICLA environment, intelligent presentations of the individual preferences of the other group members will be developed. One possible approach to do that is, for example, be a mechanism which only discloses the preferences of the other group participants at a stage of the decision process where no new alternatives could be added anymore and the decision makers have already articulated their individual preferences. Out of this discussion the question if such intelligent preference presentations can increase the willingness of group participants to articulate their real preferences will be within the scope of future research.

By taking into account the results of our study concerning anchoring effects (see Section 8.4) we want to analyze in more detail the consequence of changing the representation types of individual user preferences on the evaluation dimensions such as, perceived quality of decision support as well as decision quality. One example of changing the individual preference representation types is the comparison of user-specific representations versus aggregated representations.

An in-depth research of further decision phenomena within the scope of group decision processes is another future research issue. For single-user cases there exist several investigations on decision phenomena such as decoy effects (Huber et al. (1982), Teppan and Felfernig (2009)) but in group decision contexts nearly no studies have been conducted. Group polarization (Zuber et al. (1992)) as well as cognitive dissonance reduction (Festinger (1957)) are essential aspects in the context of group decision scenarios which have to be investigated in more detail. Finally, a detailed understanding of intra-group dynamics (Masthoff and Gatt (2006)) seems promising for achieving higher quality group decisions.

Further recommendation approaches

Up to now our research focused more on the consequences of different degrees of judgement visibility and did not include an analysis of emotional states of the group members. Masthoff and Gatt (Masthoff and Gatt (2006)) present algorithmic approaches which are able to predict the satisfaction in group decision scenarios. Mentioned influence factors in this context are *Emotional Contagion* as well as *Conformity* (Masthoff and Gatt (2006)). Emotional contagion describes a phenomenon in which

an individual's emotional state influences the emotional states of other group members. Conformity explains situations in which the judgements of a group participant are influenced by judgements which were already articulated by other group members (Masthoff and Gatt (2006)). A research issue of future work in this context is an investigation of intra-group dynamics in CHOICLA decision scenarios.

A different approach of applying advanced hybrid recommendation technologies in future versions of the CHOICLA technologies could be the inclusion of collaborative filtering (see Section 2.3) as well as content-based filtering (see Section 2.4) approaches into group decision tasks. On the basis of the entered decision task title, the knowledge of the individual preferences of the group members (e.g. gained from former group decision processes) and an optional location information (e.g. GPS position on a mobile device) the system could present precise recommendations for relevant decision alternatives.

Long term fairness

The support of repeating decisions (Ariely and Zakay (2001)) in CHOICLA is another area of future research. Fairness is an essential aspect in this context since it has a direct impact on the conflict resolution procedure. The degree of perceived fairness of the participants has an influence on both the trust in other group members as well as the willingness to accept compromises in the resolution of conflicts of opinions (Lind et al. (2001)). Further detailed investigations are needed to improve the algorithmic approach we presented in Section 6.3.1 with the aim of achieving long term fairness among group members in recurring decision scenarios.

Preference acquisition interface

One essential part of the vision of the CHOICLA group decision support environment is an intuitive user interface. To meet this requirement we are currently in the planning phase of a user study where we focus on the investigation of different preference acquisition interfaces on mobile devices. One research question of this study is whether the presented rating technique (star-evaluation vs. smiley-evaluation vs. drag & drop list ordering) has an influence on the perceived user experience on a mobile device. Another research question we want to address in this study is whether the reduced granularity of the preference information we have to accept when using drag & drop list ordering compared to a five-point scale still leads to an accurate recommendation. Figure 9.1 sketches the preliminary interface versions we are going to use in the mentioned future user studies.

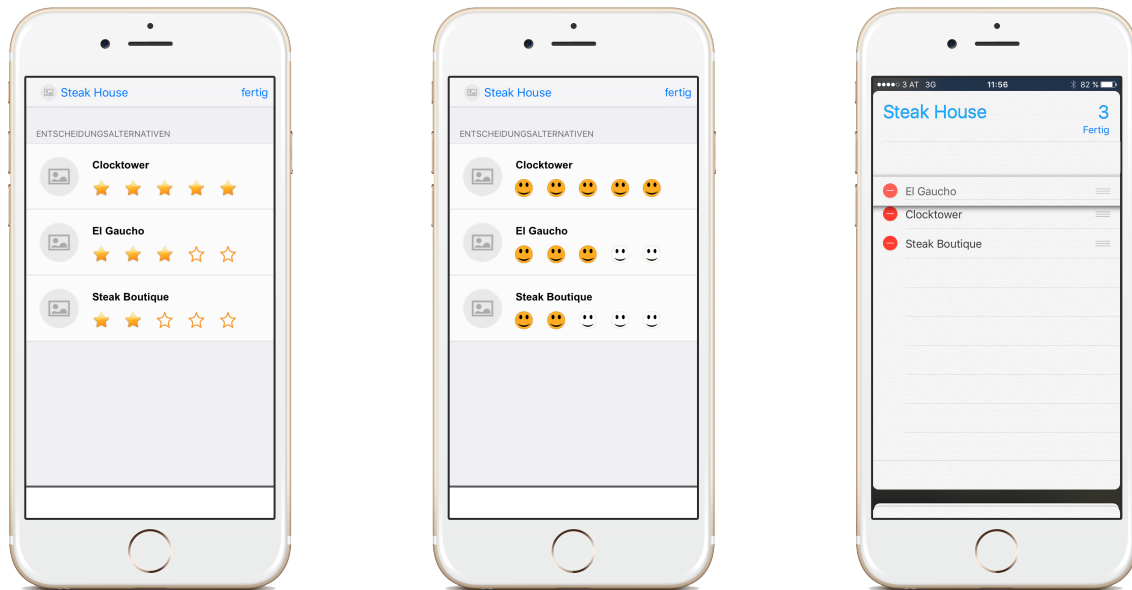


Figure 9.1.: Example of different preference acquisition interfaces on a mobile device (star-evaluation, smiley-evaluation, and drag & drop list ordering). The star-evaluation as well as the smiley-evaluation assigns a value (like resp. dislike) to each alternative whereas the drag & drop list ordering puts the alternatives in relation to each other (most preferred, second most preferred, ...).

Communication behaviour

Research from decision psychology points out that a higher degree of information exchange during the decision process has a direct positive impact on the decision quality (see, for instance, Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)). By taking into account the results we already got from our empirical study (see Section 8.4) another research focus for future work is the development and evaluation of further group recommendation heuristics which increase information exchange among the group members during the decision process. A higher degree of information interchange during the decision process helps group members to gain a better understanding of the decision problem and thus can increase the quality of decision outcomes (Greitemeyer and Schulz-Hardt (2003); Mojzisch and Schulz-Hardt (2010)).

List of Figures

2.1. Collaborative filtering (CF) dataflow: users are rating items and receive recommendations for items based on the ratings of users with a similar rating behavior – the nearest neighbors (NN).	18
2.2. Content-based filtering (CBF) dataflow: users are rating items and receive recommendations of items similar to those that have received a good evaluation from the current user (U_a).	20
2.3. Knowledge-based recommendation (KBR) dataflow: users are entering their preferences and receive recommendations based on the interpretation of a set of rules (constraints).	23
2.4. Determination of the complete set of diagnoses (hitting sets) Δ_i for the given conflict sets $CS_1 = \{r_1, r_2\}$ and $CS_2 = \{r_2, r_3\}$: $\Delta_1 = \{r_2\}$ and $\Delta_2 = \{r_1, r_3\}$	26
2.5. Example of a critiquing scenario: an entry item (l_7) is shown to the user. The user specifies the critique "less time effort". The new entry item is l_9 since it is consistent with the critique and the item most similar to l_7	32
3.1. PEOPLEVIEWS homescreen – the current version of the user interface is provided in German. The homescreen explains the basic functionalities of the system (development, maintenance, and execution of recommender applications).	44
3.2. PEOPLEVIEWS: example user attributes.	44
3.3. PEOPLEVIEWS: example of an item list.	45
3.4. PEOPLEVIEWS: example of an item evaluation user interface (evaluation of item <i>Building Loan</i> with regard to the user attribute <i>goal</i>).	45
3.5. PEOPLEVIEWS: example of a recommender application (Financial Services).	46
3.6. STUDYBATTLE "Assign Properties" learning application. The task of the user is to relate items with corresponding attribute values.	47
3.7. Results of a SUS-based usability study Bangor et al. (2008) of the PEOPLEVIEWS environment.	48

4.1.	Sketch of a user interface for game-based knowledge acquisition. The overall goal of the game is that both players agree on the set of incompatible value combinations of a given set of variables. This user interface can be regarded as a micro task template for the acquisition of incompatibility constraints.	56
4.2.	Group-based diagnosis of a faulty configuration knowledge base. Diagnoses are selected by taking into account the expertise of users/knowledge engineers: the higher the <i>personal score</i> (value derived from his/her personal contributions), the higher the weight given to his/her opinion.	57
5.1.	Fragment of the CHOICLA feature model. In this model, f_i are used as abbreviation for the individual features, for example, f_1 is the short notation for feature <i>Decision Task (Application)</i>	67
5.2.	CHOICLA: definition of a decision task. Basic settings & further configurable features in case the decision makers are allowed to contribute own alternatives during the decision process.	71
5.3.	CHOICLA: user interface for addition of decision alternatives. The dots in the upper right corner of every symbol indicate whether there is an information in this category available or not. The meaning of the used symbols is (from left to right): <i>edit, delete, geographical information, files, links</i> and <i>comments</i>	72
5.4.	CHOICLA: user interface for individual voting of decision alternatives. Each alternative can be voted by a five-star scale. The tab <i>Places</i> shows the geographical distribution of the decision alternatives (if available). In tab <i>Group Preferences</i> the actual group recommendation as well as the individual preferences of the other users (if feature f_{14} is set) is presented to the users. The process where the "final decision" can be set is triggered by the button <i>Finalize Choicla</i>	73
6.1.	<i>Choicla</i> : definition of a decision task. Basic settings & further configurable features.	77
6.2.	<i>Choicla</i> : Home screen of a registered user. The symbols within the tiles trigger actions which can be performed in the current state of the decision app. Possible actions are (from left to right): configuration, evaluation (only possible if the decision app is publicly available over the store), and delete.	80
6.3.	<i>Choicla</i> : example of individual ratings. Each user can take a look at the current recommendation and adapt his/her preferences if needed.	81
6.4.	<i>Choicla</i> : example of the entering of application data. Each applicant can insert his/her personal data needed for the advertised job position.	82
7.1.	Fragment of the CHOICLA feature model. f_i is used as abbreviation for the individual features, for example, f_3 is the short notation for feature <i>Recommendation Support</i>	89
7.2.	Example set of parameters which can be selected when creating a new CHOICLA decision app.	90

7.3.	Evaluation of the decision alternatives based on a 5-star scale.	93
7.4.	Dimension-based evaluation of the decision alternatives in CHOICLA.	93
7.5.	Actual group recommendation based on the <i>Group Distance</i> recommendation function (see Formula 7.4). The group recommendation is highlighted with bold font as well as an green arrow.	95
7.6.	CHOICLA: personal home screen of an registered user. In this setting, the user has installed four different CHOICLA decision apps which are the selection of a location, a meeting time for a jour-fixe meeting, new hardware as well as personnel decisions.	96
7.7.	Evaluations of the decision alternative <i>Zeus</i> on basis of a 5-star scale. The dotted lines show the average voting value of the alternative. Description 1 states the negative aspects at the very beginning and the end (negative salient) and Description 2 states the positive aspects at the very beginning and the end (positive salient).	97
7.8.	Evaluations (scale: [1..5]) of the decision alternative <i>Zeus</i> on basis of MAUT dimensions. The dotted lines show the average voting value of the alternative. Description 1 states the negative aspects at the very beginning and the end (negative salient) and Description 2 states the positive aspects at the very beginning and the end (positive salient).	98
7.9.	Results of the SUS-based usability study of the second group (MAUT-based evaluation of alternatives) (N=21).	100
8.1.	Interface for configuring CHOICLA decision apps.	107
8.2.	Examples of defined (configured) and generated CHOICLA decision apps.	108
8.3.	Evaluation interface of the requirements in CHOICLA. Participants can enter their ratings by selecting a value for the dimensions <i>Risk</i> , <i>Effort</i> , and <i>Profit</i>	108
8.4.	Group recommendation (on the basis of MAUT values) for the prioritization of requirements within CHOICLA. Preferences of individual stakeholders are disclosed when moving the mouse pointer over the bar.	109
8.5.	Representation of a final decision in CHOICLA in terms of a bar chart.	110
8.6.	Standard deviations of user ratings of alternatives (requirements) depending on preference disclosure time (after 1..3, or all users articulated preferences).	112
8.7.	<i>Satisfaction with final group decision</i> and <i>perceived degree of decision support</i> depending on preference disclosure time (after 1..3, or all users articulated preferences).	112
8.8.	Number of comments in the CHOICLA discussion forum depending on preference disclosure time (after 1..3, or all users articulated preferences).	113

9.1. Example of different preference acquisition interfaces on a mobile device (star-evaluation, smiley-evaluation, and drag & drop list ordering). The star-evaluation as well as the smiley-evaluation assigns a value (like resp. dislike) to each alternative whereas the drag & drop list ordering puts the alternatives in relation to each other (most preferred, second most preferred, ...). 121

List of Tables

1.1.	Contributions of this thesis with regard to the research questions.	11
2.1.	Example set of software engineering related learning units (LU) – this set will be exploited for demonstration purposes throughout this chapter. Each of the learning units is additionally characterized by a set of categories (<i>Java</i> , <i>UML</i> , <i>Management</i> , <i>Quality</i>), for example, the learning unit l_1 is assigned to the categories <i>Java</i> and <i>UML</i> .	17
2.2.	Example collaborative filtering data structure (rating matrix): learning units (LU) and related user ratings (we assume a rating scale of 1–5).	17
2.3.	Similarity between user U_a and the users $U_j \neq U_a$ determined based on Formula 2.1. If the number of commonly rated items is below 2, no similarity between the two users is calculated.	18
2.4.	User-based collaborative filtering based recommendations (predictions) for items that have not been rated by user U_a up to now.	19
2.5.	Degree of interest in different categories, for example, user U_1 accessed a learning unit related to the category <i>Java</i> three times. If a user accessed an item at least once, it is inferred that the user is interested in this item.	21
2.6.	Example of content-based filtering: <i>user</i> U_a has already consumed the items l_4 , l_6 , and l_{10} (see Table 2.2). The item most similar (see Formula 2.5) to the preferences of U_a is l_8 and is now the best recommendation candidate for the current user.	21
2.7.	Software engineering learning units (LU) described based on <i>deep knowledge</i> , for example, <i>obligatory</i> vs. <i>non-obligatory</i> , <i>duration</i> of consumption, recommended <i>semester</i> , <i>complexity</i> of the learning unit, associated <i>topics</i> , and average user rating (<i>eval</i>).	22
2.8.	Contributions of item properties to the dimensions <i>time effort</i> and <i>quality</i>	25
2.9.	Item-specific utility for user U_a ($utility(U_a, l_i)$) assuming the personal preferences for <i>time effort</i> = 0.2 and <i>quality</i> = 0.8. In this scenario, item l_4 has the highest utility for user U_a	26

2.10. Examples of hybrid recommendation approaches (RECS = set of recommenders, s = recommender-individual prediction, score = item score).	27
2.11. Example of <i>weighted</i> hybrid recommendation: individual predictions are integrated into one score. Item l_8 receives the best overall score (9.0).	28
2.12. Example of <i>mixed</i> hybrid recommendation: individual predictions are integrated into one score conform the zipper principle (best collaborative filtering prediction receives score=10.0, best content-based filtering prediction receives score=9.0 and so forth).	28
2.13. Summarization of the strengths and weaknesses of collaborative filtering (CF), content-based filtering (CBF), and knowledge-based recommendation (KBR).	30
2.14. Example of group recommendation: selection of a learning unit for a group. The recommendation (l_7) is based on the <i>least misery</i> heuristic.	33
3.1. Example users of PEOPLEVIEWS environment.	38
3.2. Example set of items used in working example.	38
3.3. User attributes $u \in U$ of example financial services recommender.	39
3.4. Example of <i>user-specific filter constraints</i> (= micro contributions).	40
3.5. Example of <i>recommendation-relevant filter constraints</i> which are the result of integrating user-specific filter constraints (see Table 3.4).	40
3.6. Support values (see Formula 3.2) derived from user-specific filter constraints (see Table 3.4).	42
3.7. Example set of user requirements ($req_i \in REQ$).	43
3.8. Utility-based ranking of items in the recommendation set.	43
3.9. Example list of micro tasks to be integrated in PEOPLEVIEWS.	49
4.1. Community-based knowledge engineering: example micro tasks.	55
4.2. Application scenarios for <i>group-based configuration</i> identified within the scope of a study with $N=25$ companies applying configuration systems.	59
4.3. Example set of tourist destinations (in the Alps-Adriatic area). The assumption in this example is that each person is allowed to articulate at most two preferences and the trip must include at least two destinations.	60
5.1. Examples of user-specific ratings with regard to the available decision alternatives (restaurants).	69
5.2. Results of applying the aggregation functions to the user preferences shown in Table 5.1. MAJ = Majority Voting; LMIS = Least Misery; MPLS = Most Pleasure; GDIS = Lowest Group Distance; ENS = Ensemble Voting. This example is based on the preference information in Table 5.1.	70

6.1. User-specific ratings with regard to the available decision alternatives (restaurants) in recurring decision situations. The group recommendation is marked bold in the corresponding decision situation.	79
7.1. Examples of individual user ratings with regard to the available decision alternatives (restaurants).	91
7.2. Results of applying the aggregation functions to the individual preferences shown in Table 7.1. MAJ = Majority Voting; LMIS = Least Misery; MPLS = Most Pleasure; GDIS = Lowest Group Distance; ENS = Ensemble Voting. This example recommendation values are based on the preference information in Table 7.1.	92
7.3. Study setting: assignment of groups to different types of preference definition support (5-star scale vs. MAUT dimensions) and descriptions of alternatives (<i>ns</i> : negative salient = <i>Description 1</i> vs. <i>ps</i> : positive salient = <i>Description 2</i>).	98
7.4. Positive salient description (= <i>Description 2</i>) of a restaurant.	99
7.5. Negative salient description (= <i>Description 1</i>) of a restaurant.	99
8.1. Assignment of versions to groups in the user study, for example, " <i>explanation mandatory+after 1</i> ." denotes a CHOICLA version with <i>mandatory explanations</i> and <i>individual preferences were disclosed</i> after one group member defined his/her preferences.	110
8.2. Avg. evaluations and std.dev. regarding (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one's personal preferences.	113
8.3. Avg. evaluations and std.dev. regarding (a) satisfaction with the final group decision, (b) perceived degree of decision support, (c) perceived understandability of the final group decision, and (d) consideration of one's personal preferences.	114

Bibliography

- ADOMAVICIUS, G., BOCKSTEDT, J., CURLEY, S., AND ZHANG, J. 2011. Recommender systems, consumer preferences, and anchoring effects. In *RecSys 2011 Workshop on Human Decision Making in Recommender Systems*. 35–42. (Cited on pages 2, 4, 5, 7, 11, 88, 105, 117, and 118.)
- ADOMAVICIUS, G., BOCKSTEDT, J., CURLEY, S., AND ZHANG, J. 2014. De-Biasing User Preference Ratings in Recommender Systems. In *RecSys 2014 Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS 2014)*. Foster City, CA, USA, 2–9. (Cited on pages 3, 88, and 105.)
- ADOMAVICIUS, G., BOCKSTEDT, J. C., CURLEY, S. P., AND ZHANG, J. 2013. Do recommender systems manipulate consumer preferences? a study of anchoring effects. *Information Systems Research* 24, 4, 956–975. (Cited on page 2.)
- ARDISSONO, L., FELFERNIG, A., FRIEDRICH, G., GOY, A., JANNACH, D., PETRONE, G., SCHÄFER, R., AND ZANKER, M. 2003. A framework for the development of personalized, distributed web-based configuration systems. *AI Magazine* 24, 3, 93–108. (Cited on page 24.)
- ARIELY, D. AND ZAKAY, D. 2001. A timely account of the role of duration in decision making. *Acta Psychologica* 108, 2, 187 – 207. Time, Judgement and Decision Making. (Cited on pages 6, 84, and 120.)
- BACHWANI, R. 2012. Preventing and diagnosing software upgrade failures. *PhD Thesis, Rutgers University*. (Cited on page 15.)
- BANGOR, A., KORTUM, P., AND MILLER, J. 2008. An empirical evaluation of the System Usability Scale (SUS). *International Journal of Human-Computer Interaction* 24, 6, 574–594. (Cited on pages 47, 48, 100, and 123.)
- BAPTISTE, P., PAPE, C. L., AND NUIJTEN, W. 2001. *Constraint-based Scheduling*. Kluwer. (Cited on page 63.)
- BAR-HILLEL, M. 2015. Position effects in choice from simultaneous displays: A conundrum solved. *Perspectives on Psychological Science* 10, 4, 419–433. (Cited on pages 3, 5, 6, 10, and 117.)

- BATORY, D. 2005. Feature models, grammars, and propositional formulas. In *Proceedings of the 9th International Conference on Software Product Lines*. SPLC'05. Springer-Verlag, Berlin, Heidelberg, 7–20. (Cited on page 66.)
- BENAVIDES, D., FELFERNIG, A., GALINDO, J., AND REINFRANK, F. 2013. Automated Analysis in Feature Modeling and Product Configuration. In *13th International Conference on Software Reuse*. Number 7925 in LNCS. Springer, Pisa, Italy, 160–175. (Cited on page 90.)
- BENAVIDES, D., SEGURA, S., AND RUIZ-CORTÉS, A. 2010. Automated analysis of feature models 20 years later: A literature review. *Information Systems* 35, 6, 615 – 636. (Cited on page 66.)
- BESSIERE, C., COLETTA, R., O'SULLIVAN, B., AND PAULIN, M. 2007. Query-driven constraint acquisition. In *21st International Joint Conference on Artificial Intelligence (IJCAI'07)*. Hyderabad, India, 50–55. (Cited on page 62.)
- BETTMAN, J., LUCE, M., AND PAYNE, J. 1998. Constructive consumer choice processes. *Journal of Consumer Research* 25, 3, 187–217. (Cited on pages 6 and 62.)
- BILLSUS, D. AND PAZZANI, M. 1998. Learning collaborative information filters. In *15th International Conference on Machine Learning (ICML'98)*. 46–54. (Cited on page 17.)
- BURKE, R. 2000. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems* 69, 32, 180–200. (Cited on pages 2, 16, 22, and 35.)
- BURKE, R. 2002. Hybrid recommender systems: survey and experiments. *User Modeling and User-Adapted Interaction* 12, 4, 331–370. (Cited on pages 27 and 29.)
- BURKE, R., FELFERNIG, A., AND GOEKER, M. 2011. Recommender systems: An overview. *AI Magazine* 32, 3, 13–18. (Cited on pages 15 and 35.)
- BURKE, R., HAMMOND, K., AND YOUND, B. 1997. The findme approach to assisted browsing. *IEEE Expert* 12, 4, 32–40. (Cited on pages 32 and 35.)
- BURKE, R. AND RAMEZANI, M. 2010. Matching recommendation technologies and domains. *Recommender Systems Handbook*, 367–386. (Cited on pages 30 and 37.)
- CHEN, L. AND PU, P. 2012. Critiquing-based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction* 22, 1–2, 125—150. (Cited on page 32.)
- CHESBROUGH, H. 2003. *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press, Boston, MA. (Cited on pages 54 and 61.)
- COSLEY, D., LAM, S., ALBERT, I., KONSTAN, J., AND RIEDL, J. 2003. Is seeing believing – how recommender system interfaces affect users' opinions. In *CHI03*. 585–592. (Cited on page 2.)
- DE KLEER, J., MACKWORTH, A., AND REITER, R. 1992. Characterizing diagnoses and systems. *AI Journal* 56, 197–222, 57–95. (Cited on page 46.)

-
- DOUGHERTY, T. W., TURBAN, D. B., AND CALLENDER, J. C. Oct 1994. Confirming first impressions in the employment interview: A field study of interviewer behavior. *Journal of Applied Psychology* 79 Issue 5, 659 – 665. (Cited on pages 6, 7, 9, and 116.)
- DYER, J. 2005. Maut — multiattribute utility theory. In *Multiple Criteria Decision Analysis: State of the Art Surveys*. International Series in Operations Research & Management Science, vol. 78. Springer New York, 265–292. (Cited on pages 7, 9, 82, 91, 92, 98, 108, and 109.)
- EKSTRAND, M., LUDWIG, M., KOLB, J., AND RIEDL, J. 2011a. LensKit: a modular recommender framework. In *ACM Recommender Systems 2011*. 349–350. (Cited on page 31.)
- EKSTRAND, M., RIEDL, J., AND KONSTAN, J. 2011b. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction* 4, 2, 1–94. (Cited on page 16.)
- FALKNER, A., FELFERNIG, A., AND HAAG, A. 2011. Recommendation Technologies for Configurable Products. *AI Magazine* 32, 3, 99–108. (Cited on page 25.)
- FANO, A. AND KURTH, S. 2003. Personal choice point: helping users visualize what it means to buy a BMW. In *8th Intl. Conference on Intelligent User Interfaces (IUI 2003)*. Miami, FL, USA, 46–52. (Cited on page 36.)
- FELFERNIG, A. 2014. Biases in Decision Making. In *Intl. Worksh. on Decision Making and Recommender Systems*. Vol. 1278. CEUR Proceedings, 32–37. (Cited on pages 2, 104, and 118.)
- FELFERNIG, A., BENAVIDES, D., GALINDO, J., AND REINFRANK, F. 2013a. Towards Anomaly Explanation in Feature Models. *Workshop on Configuration, Vienna, Austria*, 117–124. (Cited on page 66.)
- FELFERNIG, A. AND BURKE, R. 2008. Constraint-based Recommender Systems: Technologies and Research Issues. In *10th ACM Intl. Conference on Electronic Commerce (ICEC'08)*. Innsbruck, Austria, 17–26. (Cited on pages 2 and 35.)
- FELFERNIG, A., FRIEDRICH, G., GULA, B., HITZ, M., KRUGGEL, T., LEITNER, G., MELCHER, R., RIEPAN, D., STRAUSS, S., TEPPAN, E., AND VITOUCH, O. 2007a. Persuasive recommendation: Serial position effects in knowledge-based recommender systems. In *Persuasive Technology*, Y. Kort, W. IJsselsteijn, C. Midden, B. Eggen, and B. Fogg, Eds. Lecture Notes in Computer Science, vol. 4744. Springer Berlin Heidelberg, 283–294. (Cited on pages 3, 4, 6, 7, 8, 10, 22, 88, 104, 117, and 118.)
- FELFERNIG, A., FRIEDRICH, G., JANNACH, D., AND STUMPTNER, M. 2004. Consistency-based diagnosis of configuration knowledge bases. *Artificial Intelligence* 152, 2 (February), 213–234. (Cited on pages 25, 54, 56, 58, and 62.)
- FELFERNIG, A., FRIEDRICH, G., JANNACH, D., AND ZANKER, M. 2006. An Integrated Environment for the Development of Knowledge-based Recommender Applications. *Intl. Journal of Electronic Commerce (IJECE)* 11, 2, 11–34. (Cited on pages 3, 4, 5, 16, 22, 23, 24, 30, 31, 36, 62, 70, 76, 77, 89, 95, and 104.)
-

- FELFERNIG, A., FRIEDRICH, G., SCHUBERT, M., MANDL, M., MAIRITSCH, M., AND TEPPAN, E. 2009. Plausible Repairs for Inconsistent Requirements. In *IJCAI'09*. Pasadena, CA, 791–796. (Cited on pages 23, 26, and 37.)
- FELFERNIG, A., FRIEDRICH, G. E., AND JANNACH, D. 2000a. UML as Domain Specific Language for the Construction of Knowledge-based Configuration Systems. *International Journal of Software Engineering and Knowledge Engineering* 10, 4, 449–469. (Cited on page 54.)
- FELFERNIG, A., GULA, B., LEITNER, G., MAIER, M., MELCHER, R., AND TEPPAN, E. 2008. Persuasion in knowledge-based recommendation. In *3rd Intl. Conf. on Persuasive Technology*. LNCS. Springer, 71–82. (Cited on page 24.)
- FELFERNIG, A., HAAS, S., NINAUS, G., SCHWARZ, M., ULZ, T., STETTINGER, M., ISAK, K., JERAN, M., AND REITERER, S. 2014a. RecTurk: Constraint-based Recommendation based on Human Computation. In *RecSys 2014 CrowdRec Workshop*. Foster City, CA, USA, 1–6. (Cited on pages 38, 39, and 47.)
- FELFERNIG, A., HOTZ, L., BAGLEY, C., AND TIIHONEN, J. 2014b. *Knowledge-based Configuration – From Research to Business Cases*. Elsevier/Morgan Kaufmann Publishers. (Cited on pages 53 and 66.)
- FELFERNIG, A., ISAK, K., SZABO, K., AND ZACHAR, P. 2007b. The VITA Financial Services Sales Support Environment. In *AAAI/IAAI 2007*. 1692–1699. (Cited on pages 15, 36, and 53.)
- FELFERNIG, A., JANNACH, D., AND ZANKER, M. 2000b. Contextual Diagrams as structuring mechanisms for designing configuration knowledge bases in UML. In *3rd International Conference on the Unified Modeling Language (UML2000)*. Number 1939 in LNCS. 240–254. (Cited on page 62.)
- FELFERNIG, A., JERAN, M., NINAUS, G., REINFRANK, F., AND REITERER, S. 2013b. Toward the Next Generation of Recommender Systems. In *Multimedia Services in Intelligent Environments: Recommendation Services*. Springer, 81–98. (Cited on page 30.)
- FELFERNIG, A., JERAN, M., NINAUS, G., REINFRANK, F., REITERER, S., AND STETTINGER, M. 2014c. Basic approaches in recommender systems. In *Recommendation Systems in Software Engineering*, M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, Eds. Springer Verlag, 15–37. (Cited on page 15.)
- FELFERNIG, A., JERAN, M., STETTINGER, M., ABSENGER, T., GRUBER, T., HAAS, S., KIRCHENGAST, E., SCHWARZ, M., SKOFITSCH, L., AND ULZ, T. 2015. Human computation based acquisition of financial service advisory practices. In *Proceedings of the 1st International Workshop on Personalization & Recommender Systems in Financial Services, Graz, Austria, April 16, 2015*. 27–34. (Cited on page 35.)
- FELFERNIG, A. AND KIENER, A. 2005. Knowledge-based Interactive Selling of Financial Services with FSAdvisor. In *17th Innovative Applications of Artificial Intelligence Conference (IAAI05)*. Pittsburgh, Pennsylvania, 1475–1482. (Cited on page 36.)

- FELFERNIG, A., MAALEJ, W., MANDL, M., RICCI, F., AND SCHUBERT, M. 2010. Recommendation and decision technologies for requirements engineering. In *ICSE 2010 Workshop on Recommender Systems in Software Engineering*. Cape Town, South Africa, 1–5. (Cited on page 62.)
- FELFERNIG, A., REINFRANK, F., AND NINAUS, G. 2012a. Resolving Anomalies in Feature Models. In *20th Intl. Symposium on Methodologies for Intelligent Systems*. Macau, China, 1–10. (Cited on page 54.)
- FELFERNIG, A., REITERER, S., STETTINGER, M., REINFRANK, F., JERAN, M., AND NINAUS, G. 2013c. Recommender Systems for Configuration Knowledge Engineering. In *Workshop on Configuration*. Vienna, Austria, 51–54. (Cited on page 62.)
- FELFERNIG, A., SCHIPPEL, S., LEITNER, G., REINFRANK, F., ISAK, K., MANDL, M., BLAZEK, P., AND NINAUS, G. 2013d. Automated Repair of Scoring Rules in Constraint-based Recommender Systems. *AI Communications* 26, 2, 15–27. (Cited on pages 29 and 63.)
- FELFERNIG, A., SCHUBERT, M., AND REITERER, S. 2013e. Personalized Diagnosis for Over-Constrained Problems. In *23rd International Conference on Artificial Intelligence*. Peking, China. (Cited on pages 23 and 37.)
- FELFERNIG, A., SCHUBERT, M., AND ZEHENTNER, C. 2011. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)* 25, 2, 175–184. (Cited on page 50.)
- FELFERNIG, A. AND SHCHEKOTYKHIN, K. 2006. Debugging User Interface Descriptions of Knowledge-based Recommender Applications. In *ACM International Conference on Intelligent User Interfaces (IUI'06)*. Sydney, Australia, 234–241. (Cited on page 22.)
- FELFERNIG, A., STETTINGER, M., NINAUS, G., JERAN, M., REITERER, S., FALKNER, A. A., LEITNER, G., AND TIIHONEN, J. 2014d. Towards open configuration. In *Proceedings of the 16th International Configuration Workshop, Novi Sad, Serbia, September 25-26, 2014*. 89–94. (Cited on pages 5, 9, and 53.)
- FELFERNIG, A., ZEHENTNER, C., AND BLAZEK, P. 2011. CoreDiag: Eliminating Redundancy in Constraint Sets. In *22nd Intl. Workshop on Principles of Diagnosis*. Munich, Germany. (Cited on page 50.)
- FELFERNIG, A., ZEHENTNER, C., NINAUS, G., GRABNER, H., MAALEJ, W., PAGANO, D., WENINGER, L., AND REINFRANK, F. 2012b. Group decision support for requirements negotiation. In *Advances in User Modeling*, L. Ardissono and T. Kuflik, Eds. Lecture Notes in Computer Science, vol. 7138. Springer Berlin Heidelberg, 105–116. (Cited on pages 1, 2, 15, 33, 54, 68, 69, 73, 78, 88, 89, 91, and 105.)
- FESTINGER, L. 1957. *A Theory of Cognitive Dissonance*. Stanford University Press. (Cited on pages 84 and 119.)

- FLEISCHANDERL, G., FRIEDRICH, G. E., HASELBÖCK, A., SCHREINER, H., AND STUMPTNER, M. 1998. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems* 13, 4, 59–68. (Cited on page 53.)
- FORZA, C., SALVADOR, F., AND TRENTIN, A. 2008. Form postponement effects on operational performance: a typological theory. *International Journal of Operations and Production Management* 28, 1067–1094. (Cited on pages 54 and 61.)
- FRIEDRICH, G. 2004. Elimination of spurious explanations. In *European Conference on Artificial Intelligence (ECAI 2004)*. Valencia, Spain, 813–817. (Cited on page 37.)
- GKIKI, S. AND LEKAKOS, G. 2014. The Persuasive Role of Explanations in Recommender Systems. In *2nd Intl. Workshop on Behavior Change Support Systems (BCSS 2014)*. Vol. 1153. CEUR Proceedings, Padua, Italy, 59–68. (Cited on pages 3 and 104.)
- GOLDBERG, D., NICHOLS, D., OKI, B., AND TERRY, D. 1992. Using Collaborative Filtering to weave an information Tapestry. *Communications of the ACM* 35, 12, 61–70. (Cited on page 16.)
- GRASCH, P., FELFERNIG, A., AND REINFRANK, F. 2013. ReComment: Towards Critiquing-based Recommendation with Speech Interaction. In *7th ACM Conference on Recommender Systems*. ACM, 157–164. (Cited on page 32.)
- GRAVES, L. M. AND POWELL, G. N. Feb 1988. An investigation of sex discrimination in recruiters' evaluations of actual applicants. *Journal of Applied Psychology* 73(1), 20 – 29. (Cited on pages 6, 7, 9, and 116.)
- GREITEMEYER, T. AND SCHULZ-HARDT, S. 2003. Preference-consistent evaluation of information in the hidden profile paradigm: Beyond group-level explanations for the dominance of shared information in group decisions. *Journal of Personality & Social Psychology* 84(2), 332–339. (Cited on pages 2, 3, 4, 5, 7, 11, 105, 111, and 121.)
- GUNAWARDANA, A. AND SHANI, G. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research* 10, 2935–2962. (Cited on page 27.)
- HAAG, A. 1998. Sales Configuration in Business Processes. *IEEE Intelligent Systems* 13, 4, 78–85. (Cited on page 53.)
- HACKER, S. AND VONAHN, L. 2009. Matchin: Eliciting User Preferences with an Online Game. In *CHI'09*. 1207–1216. (Cited on page 37.)
- HAYES-ROTH, F., WATERMAN, D., AND LENAT, D. 1983. *Building Expert Systems*. Addison-Wesley. (Cited on pages 5 and 54.)
- HENNIG-THURAU, T., MARCHAND, A., AND MARX, P. 2012. Can automated group recommender systems help consumers make better choices? *Journal of Marketing* 76, 5, 89–109. (Cited on page 32.)

-
- HERLOCKER, J., KONSTAN, J., BORCHERS, A., AND RIEDL, J. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Conference on Research and Development in Information Retrieval*. Berkeley, CA, USA, 230–237. (Cited on page 19.)
- HERLOCKER, J., KONSTAN, J., AND RIEDL, J. 2000. Explaining Collaborative Filtering Recommendations. In *CSCW 2000*. ACM, Philadelphia, PA, USA, 241–250. (Cited on page 104.)
- HOPPENBROUWERS, S., LUCAS, P., ROMANO, D., AND MOFFAT, D. 2009. Attacking the knowledge acquisition bottleneck through Games-For-Modelling. In *AISB Symposium*. 81–86. (Cited on pages 5 and 54.)
- HOTZ, L., FELFERNIG, A., STUMPTNER, M., RYABOKON, A., BAGLEY, C., AND WOLTER, K. 2013. Configuration Knowledge Representation & Reasoning. In *Knowledge-based Configuration – From Research to Business Cases*, A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, Eds. Morgan Kaufmann Publishers, Chapter 6, 59–96. (Cited on page 54.)
- HUANG, Y., LIU, H., NG, W., LU, W., SONG, B., AND LI, X. 2008. Automating knowledge acquisition for constraint-based product configuration. *Journal of Manufacturing Technology Management* 19, 6, 744–754. (Cited on pages 54 and 62.)
- HUBER, J., PAYNE, J., AND PUTO, C. 1982. Adding Asymmetrically Dominated Alternatives: Violations of Regularity and the Similarity Hypothesis. *The Journal of Consumer Research* 9, 1, 90–98. (Cited on pages 74, 84, and 119.)
- HVAM, L., MORTENSEN, N., AND RIIS, J. 2007. *Product Customization*. Springer. (Cited on pages 53, 54, and 55.)
- JACOWITZ, K. AND KAHNEMAN, D. 1995. Measures of Anchoring in Estimation Tasks. *Personality and Social Psychology Bulletin* 21, 1, 1161–1166. (Cited on pages 2, 4, 5, 7, 11, 74, 76, 84, 87, 105, 117, and 118.)
- JAMESON, A. 2004. More than the sum of its members: challenges for group recommender systems. In *Proceedings of the working conference on Advanced visual interfaces*. AVI '04. ACM, New York, NY, USA, 48–54. (Cited on pages 1, 5, 11, 33, 59, 62, 68, 73, 77, 78, 94, and 96.)
- JAMESON, A. AND SMYTH, B. 2007. Recommendation to groups. In *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Lecture Notes in Computer Science, vol. 4321. Springer Berlin Heidelberg, 596–627. (Cited on pages 33 and 68.)
- JANNACH, D. AND BUNDGAARD-JOERGENSEN, U. 2007. SAT: A Web-Based Interactive Advisor for Investor-Ready Business Plans. In *Intl. Conference on e-Business (ICE-B 2007)*. 99–106. (Cited on page 37.)
- JANNACH, D., ZANKER, M., FELFERNIG, A., AND FRIEDRICH, G. 2010. *Recommender Systems – An Introduction*. Cambridge University Press. (Cited on pages 1, 2, 15, 18, 19, 21, 22, 27, 29, 35, and 63.)

- JOHN, I., KNODEL, J., LEHNER, T., AND MUTHIG, D. 2006. A practical guide to product line scoping. In *Software Product Line Conference 2006 (SPLC2006)*. 3–12. (Cited on page 61.)
- JUNKER, U. 2004. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *19th Intl. Conference on Artificial Intelligence. AAAI'04*. AAAI Press, 167–172. (Cited on page 25.)
- KNIJNENBURG, B., REIJMER, N., AND WILLEMSSEN, M. 2011. Each to his own: how different users call for different interaction methods in recommender systems. In *RecSys 2011*. Chicago, IL, USA, 141–148. (Cited on page 32.)
- KOBYRNOWICZ, D. AND BIERNAT, M. 1997. Decoding subjective evaluations: How stereotypes provide shifting standards. *Journal of Experimental Social Psychology* 33, 6, 579 – 601. (Cited on pages 6, 7, 9, and 116.)
- KONSTAN, J., MILLER, B., MALTZ, D., HERLOCKER, J., GORDON, L., AND RIEDL, J. 1997. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM* 40, 3, 77–87. (Cited on pages 1, 16, and 17.)
- KOREN, Y., BELL, R., AND VOLINSKY, C. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8, 42–49. (Cited on page 20.)
- KUTTY, S., CHEN, L., AND NAYAK, R. 2012. A people-to-people recommendation system using tensor space models. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing. SAC '12*. ACM, New York, NY, USA, 187–192. (Cited on pages 83 and 84.)
- LEITNER, G., FERCHER, A., FELFERNIG, A., AND HITZ, M. 2012. Reducing the Entry Threshold of AAL Systems: Preliminary Results from Casa Vecchia. In *13th Intl. Conference on Computers Helping People with Special Needs*. Linz, Austria, 709–715. (Cited on page 37.)
- LIM, S., QUERCIA, D., AND FINKELSTEIN, A. 2010. Stakenet: Using social networks to analyse the stakeholders of large-scale software projects. *32nd ACM/IEEE International Conference on Software Engineering*, 295–304. (Cited on page 15.)
- LIND, E., KRAY, L., AND THOMPSON, L. 2001. Primacy effects in justice judgments: Testing predictions from fairness heuristic theory. *Organizational Behavior and Human Decision Processes* 85, 2, 189 – 210. (Cited on pages 6, 84, 119, and 120.)
- LINDEN, G., SMITH, B., AND YORK, J. 2003. Amazon.com Recommendations – Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1, 76–80. (Cited on pages 15 and 16.)
- MACKWORTH, A. 1977. Consistency in networks of relations. *Artificial Intelligence* 8, 1, 99–118. Reprinted in *Readings in Artificial Intelligence*. (Cited on pages 55 and 66.)
- MALINOWSKI, J., WEITZEL, T., AND KEIM, T. 2008. Decision support for team staffing: An automated relational recommendation approach. *Decision Support Systems* 45, 3, 429 – 447. Special Issue Clusters. (Cited on page 84.)

-
- MANDL, M. AND FELFERNIG, A. 2012. Improving the Performance of Unit Critiquing. In *20th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2012)*. Montreal, Canada, 176–187. (Cited on page 32.)
- MANDL, M., FELFERNIG, A., TEPPAN, E., AND SCHUBERT, M. 2010. Consumer Decision Making in Knowledge-based Recommendation. *Journal of Intelligent Information Systems (JIIS)* 37, 1, 1–22. (Cited on pages 22, 62, and 87.)
- MASTHOFF, J. 2004. Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. *User Modeling and User-Adapted Interaction (UMUAI)* 14, 1, 37–85. (Cited on pages 1, 5, 32, and 73.)
- MASTHOFF, J. 2011. Group recommender systems: Combining individual models. *Recommender Systems Handbook*, 677–702. (Cited on pages 1, 4, 5, 11, 32, 33, 60, 61, 69, 73, 76, 78, 79, 91, and 106.)
- MASTHOFF, J. AND GATT, A. 2006. In Pursuit of Satisfaction and the Prevention of Embarrassment: Affective State in Group Recommender Systems. *User Modeling and User-Adapted Interaction* 16, 3–4, 281–319. (Cited on pages 105, 119, and 120.)
- MCCAREY, F., CINNEIDE, M., AND KUSHMERICK, N. 2005. Rascal – A Recommender Agent for Agile Reuse. *Artificial Intelligence Review* 24, 3–4, 253–273. (Cited on page 16.)
- MCCARTHY, K., REILLY, J., MCGINTY, L., AND SMYTH, B. 2004. On the dynamic generation of compound critiques in conversational recommender systems. In *Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer. (Cited on page 32.)
- MCCARTHY, K., SALAMO, M., COYLE, L., MCGINTY, L., SMYTH, B., AND NIXON, P. 2006. Group recommender systems: a critiquing based approach. In *2006 International Conference on Intelligent User Interfaces (IUI 2006)*. ACM, Sydney, Australia, 282–284. (Cited on pages 1, 5, 32, and 73.)
- MOJZISCH, A. AND SCHULZ-HARDT, S. 2010. Knowing other’s preferences degrades the quality of group decisions. *Journal of Personality & Social Psychology* 98, 5, 794–808. (Cited on pages 2, 3, 4, 5, 7, 11, 75, 88, 105, 111, and 121.)
- MOLIN, E., OPPEWAL, H., AND TIMMERMANS, H. 1997. Modeling Group Preferences Using a Decompositional Preference Approach. *Group Decision and Negotiation* 6, 339–350. (Cited on page 76.)
- MURPHY, J., HOFACKER, C., AND MIZERSKI, R. 2012. Primacy and Recency Effects on Clicking Behavior. *Computer-Mediated Communication* 11, 522–535. (Cited on pages 3, 4, 6, 8, 10, 88, 104, 117, and 118.)
- MUSAT, C., GHASEMI, A., A., AND FALTINGS, B. 2012. Sentiment Analysis Using a Novel Human Computation Game. In *3rd Workshop on the People’s Web Meets NLP*. 1–9. (Cited on page 62.)

- NEALE, M., ROSS, L., AND CURHAN, J. 2004. Dynamic Valuation: Preference Changes in the Context of Face-to-face Negotiation. *Journal of Experimental Social Psychology* 40, 2, 142–151. (Cited on pages 84 and 119.)
- NINAUS, G., FELFERNIG, A., STETTINGER, M., REITERER, S., LEITNER, G., WENINGER, L., AND SCHANIL, W. 2014. IntelliReq: Intelligent Techniques for Software Requirements Engineering. In *21st European Conference on Artificial Intelligence / Prestigious Applications of Intelligent Systems (PAIS 2014)*. Prague, Czech Republic, to appear. (Cited on pages 59, 62, and 81.)
- NUNAMAKER, J. F., DENNIS, A. R., VALACICH, J. S., AND VOGEL, D. R. 1991. Information Technology for Negotiating Groups: Generating Options for Mutual Gain. *Management Science* 37, 10, 1325–1346. (Cited on pages 5 and 96.)
- O’CONNOR, M., COSLEY, D., KONSTAN, J., AND RIEDL, J. 2001. PolyLens: a recommender system for groups of users. In *7th European Conference on Computer Supported Cooperative Work*. 199–218. (Cited on page 33.)
- PAZZANI, M. AND BILLSUS, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27, 313–331. (Cited on pages 15, 16, and 20.)
- PAZZANI, M. J. AND BILLSUS, D. 2007. *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter Content-Based Recommendation Systems, 325–341. (Cited on page 1.)
- PEISCHL, B., ZANKER, M., NICA, M., AND SCHMID, W. 2010. Constraint-based Recommendation for Software Project Effort Estimation. *Journal of Emerging Technologies in Web Intelligence* 2, 4, 282–290. (Cited on pages 16 and 36.)
- PEREZ, I., CABRERIZO, F., AND HERRERA-VIEDMA, E. 2010. A Mobile Decision Support System for Dynamic Group Decision-Making Problems. *IEEE Transactions on Systems, Man, and Cybernetics* 40, 6, 1244–1256. (Cited on pages 1, 5, and 73.)
- PRIBIK, I. AND FELFERNIG, A. 2012. Towards Persuasive Technology for Software Development Environments: An Empirical Study. In *Persuasive Technology Conference (Persuasive 2012)*. 227–238. (Cited on page 37.)
- PU, P. AND CHEN, L. 2007. Trust-inspiring Explanation Interfaces for Recommender Systems. *Knowledge-Based Systems*, 542–556. (Cited on pages 3, 4, and 104.)
- REITER, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32, 1 (April), 57–95. (Cited on pages 25, 26, 46, 58, and 62.)
- REITERER, S., FELFERNIG, A., BLAZEK, P., LEITNER, G., REINFRANK, F., AND NINAUS, G. 2013. WeeVis. In *Knowledge-based Configuration – From Research to Business Cases*, A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, Eds. Morgan Kaufmann Publishers, Chapter 25, 365–376. (Cited on page 37.)

- RICCI, F. AND NGUYEN, Q. 2007. Acquiring and revising preferences in a critiquing-based mobile recommender systems. *IEEE Intelligent Systems* 22, 3, 22–29. (Cited on page 32.)
- RICHARDSON, M. AND DOMINGOS, P. 2003. Building Large Knowledge Bases by Mass Collaboration. In *2nd Intl. Conference on Knowledge Capture (K-CAP'03)*. 129–137. (Cited on pages 5 and 54.)
- ROBILLARD, M., WALKER, R., AND ZIMMERMANN, T. 2010. Recommendation Systems for Software Engineering. *IEEE Software* 27, 4, 80–86. (Cited on page 16.)
- RODRIGUEZ, M., STEINBOCK, D., WATKINS, J., GERSHENSON, C., BOLLEN, J., GREY, V., AND DEGRAF, B. 2007. Smartocracy: Social networks for collective decision making. In *HICSS 2007*. IEEE, Waikoloa, Big Island, HI, USA, 90. (Cited on pages 73, 83, 88, and 106.)
- SAHM, A. AND MAALEJ, W. 2010. Switch! recommending artifacts needed next based on personal and shared context. In *Software Engineering (Workshops)*. 473–484. (Cited on page 16.)
- SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web*. 285–295. (Cited on pages 17 and 20.)
- SCHRIJVERS, T., TACK, G., WUILLE, P., SAMULOWITZ, H., AND STUCKEY, P. 2013. Search combinatorics. *Constraint Journal* 18, 2, 269–305. (Cited on pages 60 and 62.)
- SCHUBERT, M., FELFERNIG, A., AND MANDL, M. 2010. FastXPlain: Conflict Detection for Constraint-Based Recommendation Problems. In *Trends in Applied Intelligent Systems (proc. of 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2010)*, N. García-Pedrajas, F. Herrera, C. Fyfe, J. Benítez, and M. Ali, Eds. Lecture Notes in Computer Science, vol. 6096. Springer, Cordoba, Spain, 621–630. (Cited on page 54.)
- SIORPAES, K. AND HEPP, M. 2008. Games with a Purpose for the Semantic Web. *IEEE Intelligent Systems* 23, 3, 50–60. (Cited on pages 57 and 62.)
- STETTINGER, M. 2014. Choicla: Towards domain-independent decision support for groups of users. In *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys '14. ACM, New York, NY, USA, 425–428. (Cited on pages 8, 10, and 75.)
- STETTINGER, M. AND FELFERNIG, A. 2014. Choicla: Intelligent Decision Support for Groups of Users in the Context of Personnel Decisions. *Joint Workshop on Interfaces and Human Decision Making for Recommender Systems co-located with ACM Conference on Recommender Systems (RecSys 2014)*, 28–32. (Cited on pages 7, 8, 9, 12, 75, 94, and 116.)
- STETTINGER, M., FELFERNIG, A., LEITNER, G., AND REITERER, S. 2015a. Counteracting anchoring effects in group decision making. In *User Modeling, Adaptation and Personalization*, F. Ricci, K. Bontcheva, O. Conlan, and S. Lawless, Eds. Lecture Notes in Computer Science, vol. 9146. Springer International Publishing, 118–130. (Cited on pages 3, 7, 11, and 103.)

- STETTINGER, M., FELFERNIG, A., LEITNER, G., REITERER, S., AND JERAN, M. 2015b. Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment. In *20th ACM Conference on Intelligent User Interfaces (IUI2015)*. ACM, Atlanta, Georgia, USA, 148–157. (Cited on pages 3, 10, 87, 104, 109, and 117.)
- STETTINGER, M., FELFERNIG, A., NINAUS, G., JERAN, M., REITERER, S., AND LEITNER, G. 2014. Configuring Decision Tasks. *Workshop on Configuration, Novi Sad*, 17–21. (Cited on pages 1, 4, 7, 9, 65, 77, 90, 106, and 107.)
- STETTINGER, M., NINAUS, G., JERAN, M., REINFRANK, F., AND REITERER, S. 2013. We-decide: A decision support environment for groups of users. In *Recent Trends in Applied Artificial Intelligence*, M. Ali, T. Bosse, K. Hindriks, M. Hoogendoorn, C. Jonker, and J. Treur, Eds. Lecture Notes in Computer Science, vol. 7906. Springer Berlin Heidelberg, 382–391. (Cited on pages 8, 33, 66, 75, 83, and 91.)
- STUMPTNER, M. 1997. An Overview of Knowledge-based Configuration. *AI Communications* 10, 2, 111–126. (Cited on pages 53 and 90.)
- TAKACS, G., PILASZY, I., NEMETH, B., AND TIKK, D. 2009. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research* 10, 623–656. (Cited on page 16.)
- TEPPAN, E. AND FELFERNIG, A. 2009. Asymmetric Dominance- and Compromise Effects in the Financial Services Domain. In *IEEE International Conference on E-Commerce and Enterprise Computing (CEC/EEE 2009)*. Vienna, Austria, 57–64. (Cited on pages 62, 84, and 119.)
- TEPPAN, E. AND FELFERNIG, A. 2012. Minimization of Decoy Effects in Recommender Result Sets. *Web Intelligence and Agent Systems* 10, 4, 385–395. (Cited on pages 88 and 104.)
- TIIHONEN, J. AND FELFERNIG, A. 2010. Towards recommending configurable offerings. *International Journal of Mass Customization* 3, 4, 389–406. (Cited on page 31.)
- TINTAREV, N. AND MASTHOFF, J. 2007. Explanations of Recommendations. In *ACM Conf. on Recommender Systems 2007*. ACM, Minneapolis, MN, USA, 203–206. (Cited on pages 3 and 104.)
- TSUNODA, M., KAKIMOTO, T., OHSUGI, N., MONDEN, A., AND MATSUMOTO, K. 2005. Javawock: A java class recommender system based on collaborative filtering. In *SEKE 2005*. Taipei, Taiwan, 491–497. (Cited on page 16.)
- TVERSKY, A. AND SIMONSON, I. 1993. Context-dependent Preferences. *Management Science* 39, 10, 1179–1189. (Cited on pages 88 and 104.)
- VON AHN, L. 2005. Human Computation. In *Technical Report CMU-CS-05-193*. Carnegie Mellon University, School of Computer Science. (Cited on pages 12, 37, 47, and 62.)
- VON AHN, L. 2006. Games with a Purpose. *IEEE Computer* 39, 6, 92–94. (Cited on page 57.)

- WAGNER, C. 2006. Breaking the Knowledge Acquisition Bottleneck through Conversational Knowledge Management. *Information Resources Management Journal* 19, 1, 70–83. (Cited on pages 5 and 54.)
- YANG, B. AND BURNS, N. D. 2003. Implications of postponement for the supply chain. *International Journal of Production Research* 41, 9, 2075–2090. (Cited on pages 54 and 61.)
- YARDI, S., HILL, B., AND CHAN, S. 2005. VERN: Facilitating Democratic group Decision Making Online. In *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2005)*. ACM, Sanibel Island, Florida, USA, 116–119. (Cited on pages 73, 83, 88, and 105.)
- ZUBER, J., CROTT, H. W., AND WERNER, J. 1992. Choice shift and group polarization: An analysis of the status of arguments and social decision schemes. *Journal of Personality and Social Psychology* 62, 1, 50–61. (Cited on page 119.)