

INTELLIREQ: Intelligent Techniques for Software Requirements Engineering

Gerald Ninaus¹ and Alexander Felfernig¹ and Martin Stettinger¹
and Stefan Reiterer² and Gerhard Leitner³ and Leopold Weninger⁴ and Walter Schanil⁴

Abstract. Requirements Engineering is considered as one of the most critical phases of a software development project. Low-quality requirements are a major reason for the failure of a project. Consequently, techniques are needed that help to improve the support of stakeholders in the development of requirements models as well as in the process of deciding about the corresponding release plans. In this paper we introduce the INTELLIREQ Requirements Engineering environment. This environment is based on different recommendation approaches that support stakeholders in requirements-related activities such as definition, quality assurance, reuse, and release planning. We provide an overview of recommendation approaches integrated in INTELLIREQ and report results of empirical studies that show in which way recommenders can improve the quality of Requirements Engineering processes.

1 Introduction

Requirements Engineering can be defined as *the branch of systems engineering concerned with the desired properties and constraints of software-intensive systems, the goals to be achieved in the software's environment, and assumptions about the environment* – see [4]. Major phases of a Requirements Engineering process are elicitation & definition, quality assurance, negotiation, and release planning [32]. Requirements Engineering is a critical phase of a software development project since low-quality requirements are a major reason for the failure of a project [14]. The corresponding follow-up costs can add up to 40% of the overall project costs [19].

Due to the increasing size and complexity of software systems, there is a growing demand for intelligent approaches that can help to improve the quality of Requirements Engineering processes [8, 22, 24, 30]. Existing Requirements Engineering tools primarily support the definition and cataloging of requirements but fail to provide additional information such as hidden relationships between requirements and quality status of requirements. Furthermore, these tools do not support decision making scenarios where mediation support is needed (e.g., in the case of contradicting opinions and preferences of stakeholders). Finally, no mechanisms are integrated that help to increase user involvement although a low degree of involvement in many cases leads to project failure [20].

In this paper we introduce the INTELLIREQ environment⁵ which

exploits *recommendation technologies* [3, 18] to support Requirements Engineering tasks. INTELLIREQ supports *Early Requirements Engineering* where the major focus is to figure out and prioritize high-level requirements in software projects. The outcome of INTELLIREQ is a consistent set of high-level requirements with corresponding effort estimations and a release plan for the implementation of the identified requirements. All information units (e.g., requirements, dependencies, and release plan) are summarized in a high-level specification book. INTELLIREQ is applied by the industry and research partners of the Graz University of Technology.

The remainder of this paper is organized as follows. In Section 2 we show how recommender systems can support the development of requirements models. In Section 3 we present the INTELLIREQ user interface and discuss related functionalities. An overview of empirical studies related to the INTELLIREQ environment and the business benefits is given in Section 4. Section 5 contains a discussion of related and future work. The paper is concluded with Section 6.

2 INTELLIREQ Recommendation Technologies

In this section we provide an overview of recommendation technologies integrated in INTELLIREQ. A recommender system can be defined as *any system that guides a user in a personalized way to interesting or useful objects in a large space of possible options or that produces such objects as output* [1, 3]. Recommender systems support the identification of relevant items in situations where the complexity of an item assortment outstrips a user's capability to survey it and to reach a decision [2].

Recommendation Approaches. There are the following *four basic types* of recommendation approaches.

Collaborative Filtering [16, 21] is an implementation of word-of-mouth promotion where purchase decisions are taken on the basis of the opinion of relatives and friends: if users *A* and *B* rated similar items in a similar fashion in the past, Collaborative Filtering will propose new items to user *A* that *B* already rated positively.

Content-based Filtering [26] exploits features (e.g., keywords) of items a user liked in the past for the determination of recommendations. For example, if a customer of amazon.com bought books related to the Java programming language, similar books (related to Java) will be recommended in the future.

More complex items such as financial services or apartments are recommended by *knowledge-based recommenders* [1, 7]. In this case, constraints define the relationship between user requirements and the corresponding items and are thus responsible for the determination of recommendations.

Finally, *group recommenders* [13, 17, 23] recommend items for

¹ Graz University of Technology, Austria, email: {gerald.ninaus, alexander.felfernig, martin.stettinger}@ist.tugraz.at

² SelectionArts Ltd., Austria, email: stefan.reiterer@selectionarts.com

³ Alpen-Adria-Universität Klagenfurt, Austria, email: gerhard.leitner@aau.at

⁴ wsop Ltd., Vienna, Austria, email: {lweninger, wschanil}@wsop.at

⁵ www.intellireq.org

Table 1. Example of a content-based filtering recommendation scenario (REQ = set of requirements).

$r_i \in REQ$	category	release	effort	description
r_1	database	1	140 hours	store portfolio configuration in database
r_2	UI	1	300 hours	configurator UI with online help available
r_3	database	1	100 hours	Hibernate based database access
r_4	UI	2	200 hours	configurator UI with corporate identity

groups of users (e.g., recommendation of a hotel to a group of tourists who plan a common holiday trip).

Recommendation Approaches in INTELLIREQ. The following discussions are based on a simplified scenario which includes the four requirements depicted in Table 1. On the basis of this scenario we show how recommendation approaches can be exploited to support different types of Requirements Engineering activities (e.g., requirements definition, quality assurance, and release planning). In this context we want to emphasize that recommenders are key-supportive technologies, however, we do not claim that information gaps in general can be tackled by their application. For example, efficient Requirements Engineering (RE) processes heavily rely on the personal communication between stakeholders which cannot be substituted by recommenders. Based on our scenario we now exemplify the application of recommendation approaches in RE.

Content-based Filtering. Content-based Filtering [26] exploits similarities between user preferences and descriptions of items (items not known to the user up to now). User preferences are often represented in terms of keywords extracted from textual item descriptions – see also [28]. Alternatively, items can be described in terms of categories (semantic descriptions). Typical recommendations determined by Content-based Filtering are of type *item A is recommended since you purchased item B which is similar to A*.

When defining requirements, stakeholders can be supported, for example, by pointing out requirements defined by other stakeholders in the current project that are similar to the current one. Furthermore, requirements can be recommended for reuse, i.e., requirements already defined in previous projects can be more easily retrieved and reused in the current project. The similarity between two requirements in the set REQ of defined requirements ($\{r_a, r_b\} \subseteq REQ$) can be determined on the basis of Formula 1 (Dice coefficient which is a variation of the Jaccard coefficient “intensively” taking into account keyword commonalities – see also [18]).

$$sim(r_a, r_b) = \frac{2 * |keywords(r_a) \cap keywords(r_b)|}{|keywords(r_a)| + |keywords(r_b)|} \quad (1)$$

For example, $sim(r_1, r_3) = 0.33$, since $keywords(r_1) = \{store, portfolio, configuration, database\}$ and $keywords(r_3) = \{database, Hibernate\}$ (see Table 2). Let us assume that the active stakeholder (st) has already investigated the requirement r_1 . Now, Content-based Filtering would recommend requirement r_3 if r_3 has not been investigated by the active stakeholder up to now. If available, metadata can as well be exploited for determining the similarity between requirements – in this situation, keywords (see Formula 1) have to be substituted by category descriptions (see Table 1).

In INTELLIREQ, dependency detection is based on Content-based Filtering. Dependency detection is the task of identifying semantic relationships between requirements. Examples of relationships be-

Table 2. Keywords extracted from the textual requirement descriptions in Table 1 (REQ = set of requirements).

$r_i \in REQ$	extracted keywords
r_1	store, portfolio, configuration, database
r_2	configurator, UI, help
r_3	database, Hibernate
r_4	configurator, UI

tween two requirements (r_a and r_b) are r_a requires r_b , r_a is incompatible with r_b , r_a refines r_b , and r_a is part of r_b . INTELLIREQ does not identify relationships between requirements on a semantic level but on the level of similarities, i.e., the basic assumption is that similarity between requirements can be an indication of dependency. The assertion of a concrete dependency is the task of the stakeholders. In addition to the afore discussed Content-based Filtering approach, INTELLIREQ exploits semantic information extracted from OpenThesaurus⁶ instead of keywords. This OpenThesaurus enhanced version is integrated in the current INTELLIREQ version (www.intellireq.org).⁷

Group Recommendation. The major goal of group recommendation technologies [13, 17, 23] is to foster consensus among group members. Group recommenders can support group decision making by taking into account the fact that individual decisions depend on factors such as own evaluation of an alternative, beliefs about group member opinions, and information about the individual motivations (e.g., egocentric or cooperative motivation [17]). Group recommenders include heuristics [23] that can be used for identifying alternatives that will be accepted by all or at least a majority.

Requirements evaluation & negotiation are basic application scenarios for group recommenders since stakeholders have to cooperatively decide about the quality of requirements and also to figure out in which way requirements should be taken into account in the release plan. For demonstration purposes we assume that the requirement r_1 has already been evaluated by the four stakeholders $\{st_a, st_b, st_c, st_d\}$ – evaluations are depicted in Table 3.

Table 3. Example of a decision problem: deciding about the group evaluation of requirement r_1 (MAJ = majority voting as decision heuristic).

r_1	st_a	st_b	st_c	st_d	MAJ
quality	medium	medium	medium	high	medium
priority	high	high	medium	high	high
decision	accept	revision	accept	accept	accept

⁶ www.openthesaurus.de

⁷ The inclusion of English thesauri such as WordNet (wordnet.princeton.edu) is within the scope of future work.

Meta Data

Description - Bluetooth

The watch needs a bluetooth connection. This is necessary to connect the watch to other devices. Example: heart rate chest strap

Meta Data	Adjust	Info
Risk: 4 (Average)	<input type="range"/>	●
Feasibility: 8 (High)	<input type="range"/>	●
Cost: 5 (Average)	<input type="range"/>	●
Relevance: 4 (Average)	<input type="range"/>	●
Priority: 1 (Low)	<input type="range"/>	●
Duration: h	<input type="text" value="20"/>	●
Preferred Release:	<input type="text" value="R2 - 2014-03-13"/>	●

Select Meta Data Type to view

Priority

All ratings for Priority

Picture	Name	Rating
	IRManagement	4
	IRKunde	7
	AlexanderFelfernig	1

Recommendation

Majority Recommendation: 1

Average Recommendation: 4

Save

Figure 1. INTELLIREQ: details regarding a single requirement. The three stakeholders provided inconsistent ratings for the property *priority* which is indicated by the traffic light feedback mechanism.

In order to determine a group recommendation we can apply group decision heuristics [23]. For example, the *majority voting* strategy (see Table 3) recommends a value that represents the majority in the set of individual votes. Another example of such a group decision scenario is the assignment of requirements to software releases. In this context as well stakeholders can have different preferences regarding the assignment of a requirement to a specific release. When applying majority voting (MAJ), the release with the highest number of votes would be assigned to the corresponding requirement. As a result of a couple of empirical studies [9], majority voting (MAJ) has been selected and integrated as the primary decision heuristic of the INTELLIREQ environment. In addition to the analysis of individual decision heuristics, [9] also introduce a meta-heuristic that combines individual heuristics into an ensemble. Ensemble-based heuristics showed to outperform individual heuristics [25] and therefore will be integrated and evaluated in new versions of INTELLIREQ.

Knowledge-based Recommendation. Knowledge-based recommendation [1, 7] exploits deep knowledge about items, user requirements and preferences, and their relationships. Recommendation knowledge is represented in terms of constraints (rules) which indicate the relationship between user requirements/preferences and the given item set. This type of knowledge representation supports the generation of explanations as to why items are recommended or no solution could be found [12].

In the Requirements Engineering context, knowledge-based recommenders can be used, for example, for the recommendation of open issues. In Figure 1 the three stakeholders have diverging estimates regarding the *priority* of the requirement – this situation can be automatically detected by constraints that indicate open issues to be solved (using the traffic light semantics).

Furthermore, knowledge-based recommenders can be applied in the context of release planning. In INTELLIREQ release plans are manually defined by stakeholders. Inconsistencies between stake-

holder preferences can be repaired on the basis of heuristic search based diagnosis. In addition, we have developed concepts that allow a model-based diagnosis (MBD) of inconsistencies [12, 29]. MBD identifies a minimal set of changes in the requirements model such that consistency can be restored. In the case of incomplete release plans (some of the requirements do not have an assigned release), INTELLIREQ can propose completions that are based on recommendations of group recommendation algorithms [23].

3 INTELLIREQ User Interface

Figure 1 and Figure 2 provide an impression of the way in which users can define and manage their requirements in INTELLIREQ. Each requirement has a textual description and is associated with a set of properties (metadata) that describe specific characteristics of a requirement, for example, associated risk, feasibility, and costs. Each stakeholder is encouraged to evaluate requirements with regard to the given set of properties (metadata). For each requirement, INTELLIREQ provides group recommendations that support a group of stakeholders in deciding about the evaluation of the requirement.

INTELLIREQ automatically identifies potential dependencies between requirements and determines recommendations that are ranked conform to the degree of similarity between the requirements (see “support value” in Figure 2). In the current version, dependency recommendations can be selected and (manually) transformed into corresponding formal dependencies (e.g., *requires* and *incompatible*) that are taken into account as constraints [33] in release planning.

An important functionality are traffic lights which summarize open issues in an requirements model. For example, if stakeholders evaluate requirement properties differently (e.g., requirement *r* is considered as infeasible by stakeholder *A* but completely feasible by stakeholder *B*), then the corresponding traffic light is red which points out that additional evaluations are needed. In the current ver-

Activated Unrelated Recommendation				
Dependencies				
	Requirement A	Support Value	Requirement B	
	Ideal BMI	0.83	Size, Weight, Bod...	Evaluate
	Time Synchronisat...	0.66	Radio Modul	Evaluate
	Charge Function	0.33	Charge Mode	Evaluate
	Energy Consumptio...	0.33	Size, Weight, Bod...	Evaluate
	Energy Consumptio...	0.33	Ideal BMI	Evaluate
Recalculate				

Figure 2. INTELLIREQ: recommendation of dependencies; dependency recommendation is based on OpenThesaurus (www.openthesaurus.de), i.e., INTELLIREQ currently supports German, the English descriptions used in this paper have been included for reasons of understandability.

sion of INTELLIREQ, a red light is displayed if the corresponding user evaluation exceeds the standard deviation, an orange light is used to point out a low number of stakeholders (less than two) who took a look at the requirement, otherwise a green light is shown.

In INTELLIREQ, traffic lights are included on different *levels*: (1) contradicting evaluations on the level of requirement properties, (2) neglected requirements in the context of quality assurance (e.g., a requirement has never been evaluated by a stakeholder), (3) unexplained decisions for release plans, and (4) effort-related inconsistencies in the current release plan (e.g., due to too many requirements in a specific release). If the overall implementation effort of requirements assigned to a release is too low or too high, this situation is reflected in terms of red or yellow lights (see Table 4).

Table 4. Constraints related to the allowed implementation effort of requirements assigned to a release ($RS = \frac{\text{actual effort}}{\text{allowed effort}}$).

RS	green	yellow	red
<90%		x	
90-100%	x		
>100%			x

4 User Studies and Benefits

In order to analyze improvements that can be achieved by INTELLIREQ, we conducted different system evaluations that will be discussed in the following. First, we analyzed the usability of the INTELLIREQ user interface. Second, we evaluated different INTELLIREQ recommendation approaches.

Usability. This study has been conducted at the Graz University of Technology. N=20 subjects (85% male and 15% female) interacted with the INTELLIREQ environment and developed a requirements model (set of requirements) for an application domain they could choose on their own. In a second step the subjects had to switch to a predefined example set of requirements (digital watch) and to complete a predefined set of tasks such as defining dependencies between the given requirements (with the support of the INTELLIREQ dependency detection) and evaluating meta-properties (e.g., risk level, feasibility, and costs) of requirements. After having completed these tasks, the participants had to fill out a questionnaire based on the system usability scale (SUS) and to answer further questions regarding the applicability of the INTELLIREQ environment. The subjects

of the study agreed on the applicability of the system. INTELLIREQ is easy to use and the majority of the subjects stated that they are willing to use the system on a regular basis (see Figure 3).

Recommendation Support. Further feedback provided by the subjects of the usability study was the following. Content-based dependency recommendations were appropriate and helped to increase the quality of requirement models (average evaluation 4.35).⁸ Content-based recommendation algorithms also alleviated the search for and the reuse of requirements (4.26). Recommendations regarding quality assurance tasks (in terms of a traffic light signal) are helpful and should be constantly shown to the user (4.22).

The outcome of previous evaluations (see [13]) was that group recommendation increases the perceived system usability and quality of decision support. In this context it is important to not disclose individual preferences of group members in early phases of a decision process. The reason for this is that the knowledge about the preferences of other group members leads to an insufficient exchange of decision-relevant information. In future INTELLIREQ versions we will make this property configurable, i.e., if the administrator prefers to disclose the preferences (evaluations) of different users from the very beginning, (s)he will be able to do so. Finally, recommendations to groups intensify discussions between group members which itself has a positive impact on the quality of the decision outcome [13]. The reason for this is that discussions between group members increase information sharing which itself increases requirements-related knowledge of group members and thus improves the quality of the information needed for taking a decision.

In addition to these evaluation results we were interested in the impact of recommendations (traffic light support) on quality assurance practices. For example, if less than two other stakeholders (not the originator of the requirement) took a look at a specific requirement, a yellow traffic light is shown (a red traffic light is shown if no other stakeholder took a look at the requirement). This is a kind of knowledge-based recommendation where a constraint specifies the status of the traffic-light. From the psychological point of view people prefer situations where things are complete and they do not need to think about these things any further (completion directly leads to a sense of having achieved closure). Contrary to this, incomplete things

⁸ 1 = I do not agree, 2 = I partially agree, 3 = I rather agree, 4 = I agree, 5 = I totally agree.

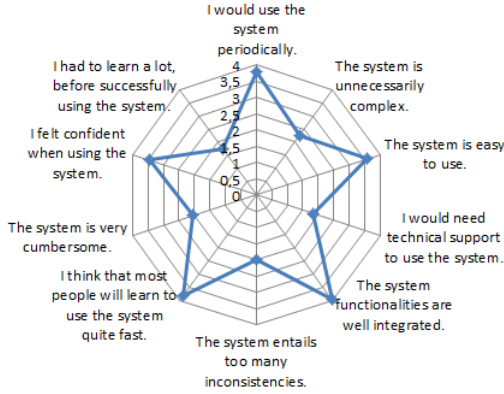


Figure 3. SUS usability evaluation: average ratings, N=20 (1 = I do not agree, 2 = I partially agree, 3 = I rather agree, 4 = I agree, 5 = I totally agree).

leave us unsatisfied and we seek to resolve the existing incompleteness.

In order to analyze the impact of traffic lights based user guidance we conducted an empirical study with N=32 computer science students (22% female and 78% male).⁹ In the role of a release manager their task was to analyze an existing requirements model, resolve inconsistencies in the model, and to generate a corresponding release plan. The outcome of this study was the following. When supported by traffic light based recommendations regarding quality assurance tasks (50% of the subjects received such recommendations), users needed significantly less interaction steps (e.g., in terms of the number of changes of the requirements view) ($p < 0.01$) and less time ($p < 0.02$) to successfully complete the given task. Traffic light based indication of tasks also persuaded subjects to document their decisions regarding a release plan ($p < 0.01$).

The semi-automated detection of dependencies between requirements in INTELLIREQ is based on content-based filtering where the requirements most similar to the requirement currently under investigation are presented to the user. The underlying assumption is that similarity between requirements is an indicator for dependencies. In order to evaluate the quality of the current dependency detection approach (see Section 2), we measured precision as an indicator of prediction quality (see Formula 2). The average precision (stakeholders *accepted* this recommendation as a dependency) measured in the current projects is 0.692 for the 10 top-ranked requirements (those with the highest support value). Note that even for very small projects with about 100 requirements, the theoretical number of pairs to be analyzed with regard to dependencies is 4.950.

$$precision = \frac{|accepted(req_i)|}{|recommended(req_i)|} \quad (req_i \in REQ) \quad (2)$$

Summarization of Benefits. The major benefits of the INTELLIREQ Requirements Engineering environment are the following. *Time efforts* related to the development and quality assurance of requirements can be reduced due to a more *systematic approach of quality assurance* (traffic light based indication of open issues) and due to a *group recommendation support* that helps to mediate between different stakeholders (e.g., in the case of contradicting evaluations of requirements). A further reason for time savings is the *recommendation of potential dependencies* between requirements which otherwise would have to be figured out manually. A more systematic analysis of dependencies can also cause a reduction of inconsistent

definitions in the requirements model. In the same line, the support of *requirements reuse* can help to avoid the definition of redundant requirements. From the psychological standpoint, the traffic light based indication of open issues exploits the phenomenon *need for completion* and thus increases individual *user engagement*.

5 Related and Future Work

Recommender Systems in Software Engineering. The application of recommendation technologies in Software Engineering is manifold and ranges from method call recommendations in software development [34] to the recommendation of effort estimation methods in project management [27]. An overview of the application of recommendation technologies in Software Engineering can be found in [31]. A detailed overview of the application of recommendation technologies in Requirements Engineering can be found in [10]. In the remainder of this section we focus on the topics related to recommender systems in Requirements Engineering.

Stakeholder Recommendation. Crucial for the success of a project is the inclusion of the right representatives of a group. The StakeNet approach [20] supports stakeholder identification on the basis of the concepts of social network analysis. StakeNet social networks are build from individual stakeholder recommendations (e.g., *A* recommends *B* to be part of the project). An example of a network analysis operation in this context is *betweenness centrality* which counts for a specific stakeholder *st* the number of shortest paths between other stakeholders in which *st* is included. A corresponding high value indicates a person's capability of acting as a broker between groups. The inclusion of stakeholder recommendation mechanisms into the INTELLIREQ environment is an issue for future work.

Recommendation of Requirements. An approach to requirements reuse is presented in Dumitru et al. [5]. Reuse support is implemented on the basis of content-based filtering where keywords extracted from the description of the new project are matched with keywords extracted from requirements descriptions of already completed projects. In contrast to INTELLIREQ no Thesaurus information is used when determining content-based recommendations. Furthermore, in contrast to existing approaches to include recommendation techniques in Requirements Engineering processes, no group recommendations (e.g., in the context of requirements definition and release planning) are supported.

Consistency Management. Especially for informal requirements an automated consistency management is unrealistic [15]. However, semi-automated approaches as implemented in INTELLIREQ can help to reduce related efforts. INTELLIREQ provides a couple of techniques that help to improve consistency management processes. Inconsistencies can, for example, be resolved on the basis of the concepts of model-based diagnosis [29] combined with corresponding repair algorithms [7]. A discussion of the automated diagnosis of inconsistent requirement models can be found in [11].

Requirements Prioritization. Restrictions regarding available resources often require prioritization decisions regarding the set of requirements that should be implemented [4]. In disaster scenarios victims are categorized into three types: those who will die, those who will survive, and those whose survival depends on the medication (also known as *triage*). Requirements prioritization is similar: requirements that *must not be included* in the next release, those that are *optional* for the next release, and those that *must be included*. In INTELLIREQ, requirements prioritization is implemented as a group decision process where for each requirement the group as a whole has to develop a consensus regarding the prioritization (not included,

⁹ The subjects of this study did not participate in the usability study.

optional, must be included).

Future Work. (1) Requirements Engineering environments are often based on the assumption of stable stakeholder preferences (e.g., regarding the prioritization of requirements). In fact, decision processes in most cases follow a process of incremental preference construction and are subject to different types of biasing effects. Being able to take into account related decision psychological theories requires a strongly interdisciplinary research approach. (2) In order to further improve the quality of dependency detection mechanisms in INTELLIREQ, approaches from natural language processing [6] and text mining [35] have to be combined with content-based approaches currently included in INTELLIREQ. (3) The INTELLIREQ user interface will be improved in terms of integrating functionalities to automatically annotate and group requirements. (4) In future versions of INTELLIREQ we intend to provide interfaces to existing Requirements Engineering tools such as IBM Doors.

6 Conclusion

Existing Requirements Engineering tools primarily support the definition and cataloging of requirements but fail to provide additional information such as similarity of requirements, dependencies between requirements, and quality status of requirements. In this paper we presented the INTELLIREQ environment which focuses on the integration of recommendation technologies with the goal to make Requirements Engineering environments more proactive. Among the major advantages that can be expected from the application of recommendation technologies in Requirements Engineering are an increased reuse of requirements, active guidance of stakeholders, increased consistency in requirements models, and reduced time efforts needed for the construction of requirement models.

ACKNOWLEDGEMENTS

The presented work has been conducted in the project INTELLIREQ (funded by the Austrian Research Promotion Agency – 829626).

REFERENCES

- [1] R. Burke, 'Knowledge-based recommender systems', *Encyclopedia of Library and Information Systems*, **69**(32), 180–200, (2000).
- [2] R. Burke, 'Hybrid recommender systems: Survey and experiments', *UMUAI Journal*, **12**(4), 331–370, (2002).
- [3] R. Burke, A. Felfernig, and M. Goeker, 'Recommender systems: An overview', *AI Magazine*, **32**(3), 13–18, (2011).
- [4] A. Davis, 'The art of requirements triage', *IEEE Computer*, **36**(3), 42–49, (2003).
- [5] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, and C. Castro-Herrera, 'On-demand feature recommendations derived from mining public product descriptions', pp. 181–190, Waikiki, Honolulu, Hawaii, (2011). ACM/IEEE.
- [6] A. Fantechi and E. Spinicci, 'A content analysis technique for inconsistency detection in software requirements documents', in *WER2005*, pp. 245–256, Porto, Portugal, (2005).
- [7] A. Felfernig and R. Burke, 'Constraint-based recommender systems: Technologies and research issues', in *IEEE ICEC'08*, pp. 17–26, Innsbruck, Austria, (2008).
- [8] A. Felfernig, W. Maalej, M. Mandl, M. Schubert, and F. Ricci, 'Recommendation and decision technologies for requirements engineering', in *ICSE 2010 Workshop on Recommender Systems in Software Engineering*, pp. 1–5, Cape Town, South Africa.
- [9] A. Felfernig and G. Ninaus, 'Group recommendation algorithms for requirements prioritization', in *ICSE 2012 Workshop on Recommender Systems for Software Engineering (RSSE 2012)*, pp. 1–4, Zürich, Switzerland, (2012).
- [10] A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej, 'An overview of recommender systems in requirements engineering', in *Managing Requirements Knowledge Book*, pp. 315–332, Berlin Heidelberg, (2013). Springer.
- [11] A. Felfernig, M. Schubert, M. Mandl, and P. Ghirardini, 'Diagnosing inconsistent requirements preferences in distributed software projects', in *3rd International Workshop on Social Software Engineering*, pp. 1–8, Paderborn, Germany, (2010).
- [12] A. Felfernig, M. Schubert, and S. Reiterer, 'Personalized diagnosis for over-constrained problems', in *23rd International Conference on Artificial Intelligence (IJCAI 2013)*, pp. 1990–1996, Peking, China.
- [13] A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maalej, D. Pagano, L. Weninger, and F. Reinfrank, 'Group decision support for requirements negotiation', *Springer Lecture Notes in Computer Science*, (7138), 1–12, (2011).
- [14] H. Hofmann and F. Lehner, 'Requirements engineering as a success factor in software projects', *IEEE Software*, **18**(4), 58–66, (2001).
- [15] J. Iyer and D. Richards, 'Evaluation framework for tools that manage requirements inconsistency', (2004).
- [16] L. Terveen, J. Herlocker, J. Konstan, and J. Riedl, 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems*, **22**(1), 5–53, (2004).
- [17] A. Jameson, S. Baldes, and T. Kleinbauer, 'Two methods for enhancing mutual awareness in a group recommender system', in *ACM Intl. Working Conference on Advanced Visual Interfaces*, pp. 48–54, Gallipoli, Italy, (2004).
- [18] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems – An Introduction*, Cambridge University Press, 2010.
- [19] D. Leffingwell, 'Calculating the return on investment from more effective requirements management', *American Programmer*, **10**(4), 13–16, (1997).
- [20] S. Lim, D. Quercia, and A. Finkelstein, 'Stakenet: Using social networks to analyse the stakeholders of large-scale software projects', in *32nd ACM/IEEE International Conference on Software Engineering*, pp. 295–304, Cape Town, South Africa, (2010). ACM/IEEE.
- [21] G. Linden, B. Smith, and J. York, 'Amazon.com recommendations: Item-to-item collaborative filtering', *IEEE Internet Computing*, **7**(1), 76–80, (2003).
- [22] W. Maalej and A. Thurimella, 'Towards a research agenda for recommendation systems in requirements engineering', in *2nd International Workshop on Managing Requirements Knowledge*, Atlanta, USA, (2009).
- [23] J. Masthoff, 'Group recommender systems', *Recommender Systems Handbook*, 677–702, (2011).
- [24] B. Mobasher and J. Cleland-Huang, 'Recommender systems in requirements engineering', *AI Magazine*, **32**(3), 81–89, (2011).
- [25] G. Ninaus, 'Using group recommendation heuristics for the prioritization of requirements', in *Proceedings of the 6th ACM Conference on Recommender Systems*, pp. 329–332. ACM, (2012).
- [26] M. Pazzani and D. Billsus, 'Learning and revising user profiles: The identification of interesting web sites', *Machine Learning*, **27**, 313–331, (1997).
- [27] B. Peischl, M. Zanker, M. Nica, and W. Schmid, 'Constraint-based Recommendation for Software Project Effort Estimation', *Journal of Emerging Technologies in Web Intelligence*, **2**(4), 282–290, (2010).
- [28] L. Roy R. Mooney, 'Content-based book recommending using learning for text categorization', *User Modeling and User-Adapted Interaction*, **14**(1), 37–85, (2004).
- [29] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).
- [30] D. Renzel, M. Behrendt, R. Klammer, and M. Jarke, 'Requirements bazar: Social requirements engineering for community-driven innovation', in *RE 2013*, pp. 326–327, Rio de Janeiro, Brazil, (2013).
- [31] M. Robillard, R. Walker, and T. Zimmermann, 'Recommendation Systems for Software Engineering', *IEEE Software*, **27**(4), 80–86, (2010).
- [32] I. Sommerville, *Software Engineering*, Pearson, 2007.
- [33] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.
- [34] M. Tsunoda, T. Kakimoto, N. Ohsugi, A. Monden, and K. Matsumoto, 'Javawock: A java class recommender system based on collaborative filtering', in *SEKE 2005*, pp. 491–497, Taipei, Taiwan, (2005).
- [35] I. Witten and E. Frank, *Data Mining*, Elsevier, 2005.