Recommendation and Decision Technologies For Requirements Engineering

Alexander Felfernig
Applied Software Engineering
IST, TU Graz
felfernig@ist.tugraz.at

Walid Maalej
Applied Software Engineering
TU Munechen
maalejw@cs.tum.edu

Monika Schubert
Applied Software Engineering
IST, TU Graz
schubert@ist.tugraz.at

Monika Mandl
Applied Software Engineering
IST, TU Graz
mandl@ist.tugraz.at

Francesco Ricci
DB and Information Systems
University of Bolzano
Francesco.Ricci@unibz.it

ABSTRACT

Requirements engineering (RE) is considered as one of the most critical phases in the software life-cycle, and poorly implemented RE processes are among the major risks for project failure. Stakeholders are often faced with the challenge that the complexity of information outstrips their capability to survey it and to decide about which requirements should be taken into account. Additionally, preferences regarding a set of requirements are typically not known beforehand but constructed within the scope of a decision making process. In this paper we introduce a simple application scenario and discuss recommendation and decision technologies which can be exploited for proactively supporting stakeholders in their decision making.

1. INTRODUCTION

Requirements engineering (RE) is considered as one of the most critical phases in software projects [30], and poorly implemented RE constitutes a major risk for projects failure [17]. Today's software projects still have a high probability to be cancelled or to significantly exceed available resources [36]. Leffingwell [21] found that 40% of the total project budget is related to rework costs triggered by low-quality requirement documents. Requirements themselves are a verbalization of decision alternatives or already taken decisions that concern the functionality and quality of the software [4]. Stakeholders[1] engaged in RE processes typically have to deal with the following types of decisions [31]:

- *Quality decisions*, e.g., is the requirement non-redundant, concrete and understandable?
- *Preference decisions*, e.g., which requirements should be considered for the next release?
- *Classification decisions*, e.g., which topic, component, or team does this requirement belong to?
- *Property decisions*, e.g., is the effort estimation for this requirement realistic?

Individual as well as group decisions are made increasingly difficult due to the increasing size and complexity of software projects [1, 2, 7, 16, 28]. Most traditional requirements management tools fail to provide adequate support for such types of projects, even though many of them claim to support a collaborative process [10]. Although multiple users can work together to construct software requirements, existing tools provide limited support for managing large numbers of feature requests, resolving conflicts or helping to organize stakeholders into forums where they can explore their needs, generate requirements, and make common decisions [1, 7].

The major goal of our research is to address these problems by developing an intelligent recommendation & decision support system that assists stakeholders in their daily requirements engineering work. Thus the challenges of enormous information overload, continuously changing requirements and quickly aging requirements will be tackled. The major contribution of this paper is twofold. First, we introduce a basic scenario that motivates the application of recommendation and decision technologies in the context of requirements engineering. Second, we discuss different types of recommendation and decision technologies that can help to improve the overall quality of decision processes in RE.

The remainder of this paper is organised as follows. In Section 2 we introduce the mentioned application scenario. In Section 3 we discuss recommendation approaches that will serve as a foundation of our envisioned decision support system. Finally, we discuss related work in Section 4 and conclude the paper with Section 5.

2. EXAMPLE SCENARIO

We exemplify the envisioned functionality of a decision support system for RE by considering a simplified (!) RE scenario (see Figure 1). The basic functionalities proposed in this scenario have been derived from our experiences in industrial software projects we conducted, for example, in

[1]Stakeholders are persons with a specific interest on the outcome of a project: customers, programmers, analysts, designers, domain experts, decision makers, users, etc.

RSSE '10, May 4 2010, Cape Town, South Africa

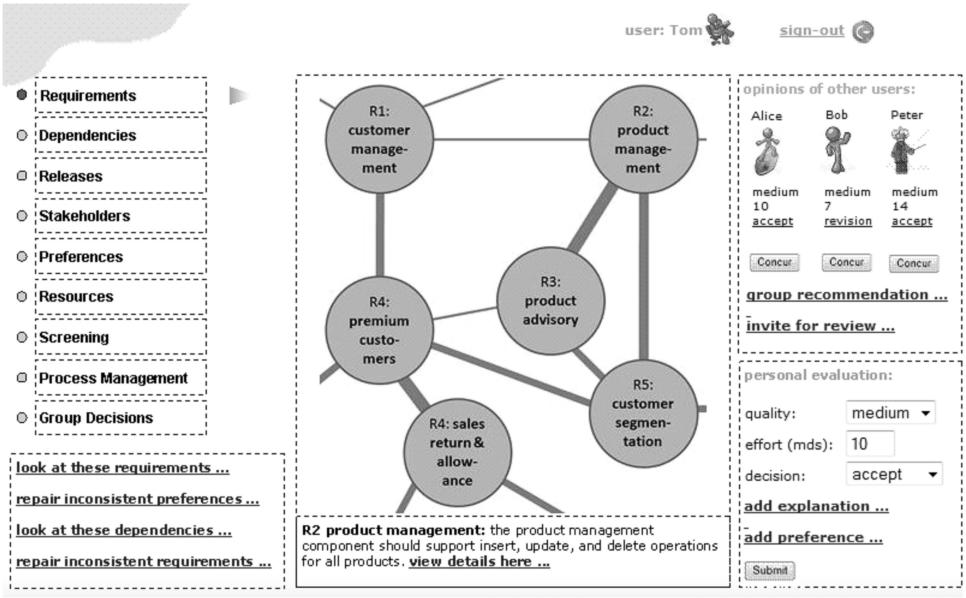

Figure 1: Recommendation & decision support for stakeholders engaged in RE processes.

the domain of financial services [13] . Although the project size was rather small (5-10 persons) we identified a couple of functionalities that have potential to improve the overall quality of decision processes in RE. Please note that we do not claim these functionalities to be complete (further analysis is needed in this context) but represent important aspects that have to be taken into account if the goal is to improve the quality of decision processes.

Figure 1 shows an example user interface of our envisioned RE decision support system. In the scenario of Figure 1 the stakeholders *Tom*, *Alice*, *Bob*, and *Peter* are cooperatively working on the requirement *R2*. Their task is to decide about the quality of the current version of *R2*. Note that in this scenario the stakeholders have specific tasks that are related to only a subset of the requirements. Decisions taken by this group of stakeholders have to be synchronized with decisions of other groups in order to achieve consensus.

The stakeholder *Tom* (current user) currently evaluates requirement *R2* regarding the dimensions quality and overall effort for implementation. He can either evaluate *R2* on his own or also take into account the opinions of other stakeholders that have already evaluated *R2* (in our case *Alice*, *Bob*, and *Peter*). *Tom* can therefore review and rate their evaluations (by clicking the corresponding link, for example, *accept* in case that he wants to review the evaluation of *Alice*). Furthermore, it is possible to invite additional stakeholders as participants of the decision process if needed (link *invite for review ...* on Figure 1).

Tom decides to concur in the evaluation of *Alice* and to provide explanations for his selection using the link *add explanation ...* (textual argumentation for the selected settings for quality, effort, and decision). Finally, *Tom* can define additional preferences on the given set of requirements in order to provide information regarding their importance. This can be done by activating *add preference ...*. *Tom* can activate the calculation of a proposal for the overall evaluation of *R2* that maximizes the agreement between *Tom*, *Alice*, *Bob*, and *Peter*. The calculation of such a proposal can be triggered by activating the link *group recommendation ...*.

On the left side of Figure 1 links to additional recommendation functionalities are depicted. The link *look at these requirements ...* recommends relevant requirements for the current task. For example those that have been developed independently by a different stakeholder but have high textual similarity with the requirements managed by *Tom*, *Alice*, *Bob*, and *Peter*. Another example of a recommendation is *look at these dependencies ...*. In this case a stakeholder is informed about potential dependencies between requirements that are not contained in the requirements model.

One further recommendation functionality is the recommendation of adaptations for stakeholder preferences with the goal to achieve maximum agreement (link *repair inconsistent preferences ...*) [14]. This functionality allows stakeholders to actively decide which preferences to change before proposals are generated for a group decision. Further examples for functionality not contained in the scenario depicted in Figure 1 are the recommendation of proposals for sets of requirements (e.g., release proposals) that help to achieve maximum consensus between stakeholders (represented by a link *look at proposed releases ...*), the recommendation of defaults (e.g., for attributes of a requirement – in our case quality, effort, and decision) implemented by a *propose value ...* functionality.

3. RECOMMENDATION & DECISION TECHNOLOGIES

In our envisioned system, recommendation and decision technologies will support two basic scenarios: *decision processes of individual stakeholders* and *group decision processes*. The differentiation is similar to the one presented in [18] where decision processes regarding travel destinations are separated into individual decisions of travel group members (construction of the individual preferences regarding

hotel type, sports facilities, etc.) and group decision processes (synchronizing the individual preferences with those of other travel group members).

3.1 Assisting individual stakeholders

We will exploit social network analysis [15] and integrate different methods for the selection of stakeholders to be included in requirements development and decision processes (especially useful in large and distributed projects [16]). Social networks [15] enable to determine the reputation of a stakeholder in certain contexts. This information can be used in contexts such as requirements selection, release planning, or selecting a stakeholder who should be responsible for the quality assurance of a set of requirements.

Stakeholders interacting with our system will be supported in two modalities: *push* (initiated by the system) and *pull* (initiated by the user) [24]. The second modality (the pull) is the more classical approach to deal with recommendations, i.e., knowledge-based, content-based, and collaborative filtering based [11, 29, 20, 32]; the stakeholder initiates the dialogue by querying the system. We will provide a set of standard queries, e.g., querying for new requirements, responsible stakeholders, or reusable requirements. According to the type of the query and the stakeholder activity this will initiate a conversation with the system. Simple recommendations will be delegated to collaborative [20] or content-based recommenders [29], recommendations based on a dialog with the stakeholder will be delegated to knowledge-based recommenders [11, 32].

For more complex items, e.g., those that cannot be selected without considering the structure and semantics of the dependencies or the preferences of other stakeholders, the request of the stakeholder will initiate a conversational recommendation session [12, 32]. Here we will use recommendation approaches based on critiques [32]. In these situations the stakeholder request will trigger a dialogue with candidate items, and with explanations for the suggestions. The stakeholder will have the possibility to critique either the recommendation itself, for instance "Stakeholder X has not enough experience to deal with requirement Y, please provide a more experienced developer for this J2ME project", or he might criticise the explanation of the system, e.g., "The dependency X is not a good reason for including requirement Y". When the recommendation deals with issues that must be negotiated with other peers, the system will activate the group assistance infrastructure (see the paragraph on *Assisting groups of stakeholders*) and the dialogue will involve other stakeholders.

Using the push modality, the system will activate the dialogue by alerting the user of an important issue that must be considered or negotiated with other stakeholders. The goal of the system in this case is to prioritize the recommendations and detect the right moment to interrupt the user. This functionality is based on context detection [3, 23]: the system detects the context, prioritizes the recommendations and selects the right timing for a user notification.

3.2 Assisting groups of stakeholders

Unlike the stakeholder assistance that focuses on a single stakeholder, the group assistance is responsible for actively supporting group decision processes [26, 18], for example, decisions in requirements screening (e.g., *is the requirement understandable enough*?), requirements negotiations (e.g., *what are acceptable trade-offs and what are the priorities of the others*?) or in release planning (*which requirements should be taken into account in the upcoming releases*?). The task of this recommendation functionality is to (a) collect stakeholder preferences, for example, regarding a specific requirement (see the simple scenario shown in Figure 1), and (b) in case of contradicting preferences, to moderate the decision process. Moderation means to propose minimal changes (repairs) to the given set of stakeholder preferences such that a solution (consistent release) can be found [12, 14]. The identification of such minimal preference sets is shown in the following example.

Example 1: repair recommendation for preferences.

Table 1 depicts a set of requirements R={r_1,r_2,r_3,r_4} and a set of stakeholders S={s_1,s_2,s_3}. For each requirement $r_i \in$ R each stakeholder specifies his/her preferences which can be 1 (*include*) and 0 (*exclude*), for example, c_{12}=1 denotes the fact that stakeholder s_1 wants to include requirement r_2 in the next software release. The set of stakeholder preferences is denoted as C=$\cup c_{ij}$. Note that inculusion and exclusion are example constraints (preferences), further types of constraints are possible (see, e.g., the RE ontology proposed by [22]) but not used in this example. For the given set of preferences shown in Table 1 there does not exist a solution, i.e, the stakeholder preferences are inconsistent.

	s_1 = Alice	s_2 = Bob	s_3 = Peter
r_1	1	1	1
r_2	1	0	1
r_3	0	0	1
r_4	1	1	1

Table 1: Inconsistent stakeholder preferences.

The first step to resolve this inconsistency is to figure out combinations of constraints (preferences) that are responsible for the inconistency, for example, the stakeholder preference c_{12} is inconsistent with the preference c_{22}. The complete set of such (minimal [19]) inconsistencies is CON = {con_1:{c_{12}, c_{22}}, con_2:{c_{22}, c_{32}}, con_3:{c_{13}, c_{33}}, con_4:{c_{23}, c_{33}}}. Such sets can be determined using the algorithm presented in [19]. We can now determine all possible repairs for the given set C of stakeholder preferences by simple deleting at least one element from each subset of CON (see [33]). The possible repair constraint sets rep_k for CON are elements of REP = {rep_1:{c_{22}, c_{33}}, rep_2:{c_{22}, c_{13}, c_{23}}, rep_3:{c_{12}, c_{32}, c_{33}}, rep_4:{c_{12}, c_{32}, c_{13}, c_{23}}} where a repair constraint set rep_k is defined as a *minimal set of stakeholder preferences* (see [14]) that have to be changed in order to make the resulting set of stakeholder preferences consistent.

For the given set REP we can identify the following set of concrete repair actions REP_c= {$repc_1$:{c_{22}=1, c_{33}=0}, $repc_2$:{c_{22}=1, c_{13}=1, c_{23}=1}, $repc_3$:{c_{12}=0, c_{32}=0, c_{33}=0}, $repc_4$:{c_{12}=0, c_{32}=0, c_{13}=1, c_{23}=1}. REP_c can now be considered as a set of alternative and minimal repairs for the original set of stakeholder preferences such that consistency between the preferences can be achieved.

The ranking and recommendation of such repair sets can be based on different approaches such as lowest *cardinality ranking* (in our case this would be the repair $repc_1$:{c_{22}=1, c_{33}=0}) or *multi attribute utility theory* (*MAUT* – see, e.g., [35]). If we want to apply MAUT we have to assign an *im-*

portance value to each of the given stakeholder preferences (see, e.g., Table 2) an then to apply, for example, the following utility formula (Formula 1) which determines for each repair candidate the corresponding utility value (see Table 3).

$$utility(repc_k) = \frac{1}{\sum_{c_{ij} \in repc_k} imp(c_{ij})} \quad (1)$$

In our simple example, the alternative repair actions would be recommended following the utilities shown in Table 3. Note that this is a recommendation forwarded to the stakeholders who are then responsible to restart the decision process. This way to recommend repair actions for inconsistent stakeholder requirements corresponds, for example, to a functionality that can be associated with the *repair inconsistent preferences ...* link in Figure 1.

	s_1	s_2	s_3
r_1	$imp(c_{11})$=0.5	$imp(c_{21})$=0.3	$imp(c_{31})$=0.4
r_2	$imp(c_{12})$=0.2	$imp(c_{22})$=0.3	$imp(c_{32})$=0.2
r_3	$imp(c_{13})$=0.2	$imp(c_{23})$=0.2	$imp(c_{33})$=0.2
r_4	$imp(c_{14})$=0.1	$imp(c_{24})$=0.2	$imp(c_{34})$=0.2

Table 2: Importance values for preferences.

$repc_k \in REP_c$	utility($repc_k$)
$repc_1$	2
$repc_2$	1.42
$repc_3$	1.66
$repc_4$	1.25

Table 3: Utility values for repair alternatives.

Example 2: group recommendations.

Let us assume that {r_1, r_2, r_3} are decision alternatives regarding the implementation of a requirement r , e.g., r: 'knowledge-based support for a car configurator application' (see Table 4). For example, {r_1: 'constraint solver' , r_2: 'description logics reasoner', r_3: 'prolog'} describe basic decision alternatives for a reasoning component that should support the implementation of a car configurator.

In this context a group recommendation functionality [18, 25, 26] could be applied which proposes a decision alternative that is acceptable for the stakeholders engaged in the decision process (in our case {s_1, s_2, s_3, s_4}). Different strategies for determining such a group recommendation are possible [18, 25, 26], for example, the so-called *least-misery* strategy [25] would propose requirements that are stable in the sense that none of the stakeholders has evaluated the alternative with a low rating. The evaluation of different recommendation strategies in such an similar scenarios is a major focus of future work. A detailed overview of such group decision strategies is provided in [25].

4. RELATED WORK

The application of recommender systems in RE contexts is a young and constantly evolving research field [1, 7, 24]. Currently the application of recommender technologies in RE is restricted to specific applications.

Castro-Herrera et al. [7] focus on the application of clustering methods for the grouping of coherent requirements. As an application scenario they introduce users of open source CRM environments who do not perform well in identifying appropriate discussion forums for their feature requests. Furthermore, basic collaborative filtering is used to recommend relevant (additional) requirements to the user.

Requirement grouping on the basis of clustering techniques is as well discussed in [9] – however, the focus of this work is different in the sense that the calculated clusters are then exploited by domain experts who are responsible for deciding about the priorities of the given requirements (without further decision support).

Chen et al. [8] apply clustering techniques to support the construction of feature models by an intelligent grouping of coherent requirements. Finally, Ruhe et al. [34] show how to apply AHP (Analytical Hierarchy Process) for determining a set of preferred requirements. This work is an important contribution to improve the quality of requirements selection. Compared to [34] our goal is to replace pure preference elicitation [27] (AHP-like utility-based approaches are an example [6]) with a preference construction paradigm [5] in requirements engineering decision-making.

5. CONCLUSIONS

In this paper we have summarized the current status of our work which focuses on the design and development of a system that supports decision processes in requirements engineering (RE) on the basis of different types of recommendation algorithms. Intelligent decision support in RE is an important issue since it can help to significantly improve the quality of software processes and to reduce the overall costs of software projects. The functionalities included in the presented application scenario have been identified within the scope of industrial software projects.

Our plan for future work is to further analyze requirements regarding decision support in different types of software projects (e.g., open source software development and large scale distributed software projects) and to implement a corresponding decision support system. We will evaluate this system within the scope of requirements engineering scenarios in university software courses as well as in large-scale industrial projects.

6. REFERENCES

[1] B. Alenljung and A. Persson. Decision-making activities in the requirements engineering decision processes: A case study. In *ISD 2005*, pages 707–718, Karlstad, Sweden, 2005.

[2] J. Alowibdi. Managing online requirements elicitation in ultra-large software systems, 2009.

[3] S. Anand and B. Mobasher. Contextual recommendation. *Discovering and Deploying User and Content Profiles*, pages 142–160, 2007.

[4] A. Aurum and C. Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

[5] J. Bettman, M. Luce, and J. Payne. Constructive consumer choice. *Journal of Consumer Research*, 25(3):187–217, 1998.

	s_1 = Alice	s_2 = Bob	s_3 = Peter	s_4 = Tom	rating of the group (least-misery)
r_1	9	7	9	10	7
r_2	2	3	9	1	1
r_3	10	10	2	6	2

Table 4: Group decisions in RE: recommending decision alternatives.

[6] R. Burke. Hybrid recommender systems: Survey and experiments. *UMUAI Journal*, 12(4):331–370, 2002.

[7] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In *16th IEEE Intl. Conf. on Req. Engineering (RE'08)*, pages 165–168, Barcelona, Spain, 2008.

[8] K. Chen, W. Zhang, H. Zhao, , and H. Mei. An approach to constructing feature models based on requirements clustering. In *13th IEEE Intl. Conf. on Req. Engineering (RE'05)*, pages 31–40, Paris, France, 2005.

[9] C. Duan. *Clustering and its application in requirements engineering*. Technical Report, DePaul University, Chicago, Illinois, 2008.

[10] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski. Towards automated requirements prioritization and triage. *Requirements Engineering*, 14:73–89, 2009.

[11] A. Felfernig and R. Burke. Constraint-based recommender systems: Technologies and research issues, 2008.

[12] A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan. Plausible repairs for inconsistent requirements, 2009.

[13] A. Felfernig, K. Isak, K. Szabo, and P. Zachar. The vita financial services sales support environment, 2007.

[14] A. Felfernig, M. Schubert, M. Mandl, and P. Ghiardini. Diagnosing inconsistent requirements preferences in distributed software projects. In *3rd International Workshop on Social Software Engineering*, pages 1–8, Paderborn, Germany, 2010.

[15] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):51–60, 1992.

[16] J. Herbsleb. Global software engineering: The future of socio-technical coordination, 2007.

[17] H. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. *IEEE Software*, 18(4):58–66, 2001.

[18] A. Jameson, S. Baldes, and T. Kleinbauer. Two methods for enhancing mutual awareness in a group recommender system, 2004.

[19] U. Junker. Quickxplain: Preferred explanations and relaxations for over-constrained problems, 2004.

[20] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news full text. *Communications of the ACM*, 40(3):77–87, 1997.

[21] D. Leffingwell. Calculating the return on investment from more effective requirements management. *American Programmer*, 10(4):13–16, 1997.

[22] S. Lohmann, T. Riechert, and S. Auer. Collaborative development of knowledge bases in distributed requirements elicitation, 2008.

[23] W. Maalej. Task-First or Context-First? Tool Integration Revisited. In *Proceedings of the ACM/IEEE International Conference on Automated Software Engineering*. IEEE Computer Society, May 2009.

[24] W. Maalej and A. K. Thurimella. Towards a research agenda for recommendation systems in requirements engineering. Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[25] J. Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85, 2004.

[26] K. McCarthy, M. Salamo, L. Coyle, L. McGinty, B. Smyth, and P. Nixon. Group recommender systems. In *11th ACM Intl. Conf. on Intelligent User Interfaces*, pages 267–269, Sydney, Australia, 2006.

[27] D. McFadden. Rationality for economists. *Journal of Risk and Uncertainty*, 19(1):73–105, 1999.

[28] L. Northrop, P. Feiler, R. Gabriel, J. Goodenought, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau. Ultra-large-scale systems: The software challenge of the future, 2006.

[29] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[30] K. Pohl. *Process-Centered Requirements Engineering*. John Wiley and Sons, 1996.

[31] B. Regnell, B. Paech, C. Aurum, C. Wohlin, A. Dutoit, and J. N. och Dag. Requirements means decision!, 2001.

[32] J. Reilly, J. Zhang, L. McGinty, P. Pu, and B. Smyth. Evaluating compound critiquing recommenders, 2007.

[33] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 23(1):57–95, 1987.

[34] G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, 13(4):354–366, 2003.

[35] D. Winterfeldt and W. Edwards. Decision analysis and behavioral research. *Cambridge University Press*, 1986.

[36] D. Yang, D. Wu, S. Koolmanojwong, A. Brown, and B. Boehm. Wikiwinwin: A wiki based system for collaborative requirements negotiation, 2008.