Palomares, C.; Franch, X.; Fucci, D. Personal recommendations in requirements engineering : the OpenReq approach. A: International Working Conference on Requirements Engineering: Foundation for Software Quality. "24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings". 2018, p. 297-304. The final authenticated version is available online at https://doi.org/10.1007/978-3-319-77243-1_19

Personal Recommendations in Requirements Engineering: The OpenReq Approach

Cristina Palomares[1], Xavier Franch[1], Davide Fucci[2]

[1] Universitat Politècnica de Catalunya, Spain
[2] University of Hamburg, Germany

Abstract. *[Context & motivation]* Requirements Engineering (RE) is considered as one of the most critical phases in software development but still many challenges remain open. *[Problem]* There is a growing trend of applying recommender systems to solve open RE challenges like requirements and stakeholder discovery; however, the existent proposals focus on specific RE tasks and do not give a general coverage for the RE process. *[Principal ideas/results]* In this research preview, we present the OpenReq approach to the development of intelligent recommendation and decision technologies that support different phases of RE in software projects. Specifically, we present the OpenReq part for personal recommendations for stakeholders. *[Contribution]* OpenReq aim is to improve and speed up RE processes, especially in large and distributed systems.

Keywords: Recommender systems, Personal recommendations, Requirements engineering

1 Introduction

High-quality Requirements Engineering (RE) is among the most critical phases for successful software development projects [1]. Due to the increasing size and complexity of these projects, we can observe a growing demand for recommender systems that can help to improve the overall quality of RE processes [2, 3].

Recommender systems help engineers to find information and to make decisions in situations where they lack experience or cannot consider all the data at hand. These systems proactively tailor suggestions that meet the particular information needs and preferences of users. In spite of the growing trend of using recommenders systems in RE, the current applications focus on specific RE tasks and do not address the RE process as a whole.

We propose the OpenReq approach to overcome such limitation. The overall goal of OpenReq [4] is to develop intelligent recommendation and decision technologies that support different phases of RE, specifically elicitation, specification, analysis, management and negotiation. The project focuses on using artificial intelligence-based techniques that proactively support stakeholders, both as individuals and as groups, within the scope of RE. This paper focuses on the recommendations of OpenReq for stakeholders as individuals, and explains the initial considerations about these recommendations and the technological approaches that will be used to develop such system.

In the following, we will provide an overview of the existent recommender systems for RE (Section 2). Section 3 will present the OpenReq's approach to personal recommendations. Finally, we conclude the paper in Section 4.

2 Recommender Systems in RE

Although recommender systems have been mostly applied to the Web (e.g., e-commerce, search engines), these systems have gained attention in different fields, one of them being Software Engineering (SE) [5]. Within SE, a prominent use case is related to bugs, from distinguish bugs from enhancements [6] to assign bugs to the right developer [7]. Other recommenders systems proposed for SE are used to predict defect priority [8], identify services that best suit the customer' needs [9], and give developers recommendations related to source code [10]. Accordingly, recommenders systems are also used in RE to help in the early stages of SE [2, 3]. Table 1 highlights some of the representative approaches of recommender systems for RE according to the RE stage they tackle, the main focus of this stage, the types of requirements they handle, and if there is an implementation available.

Recommenders systems have especially been applied to the elicitation and specification of requirements. Some approaches focus on recommending a specific type of requirement (e.g., sustainability requirements [11]) and, in a more general way, non-functional requirements [12], while others are applicable to any kind of requirement [13, 14]. Other approaches focus on identifying stakeholders that could help during the elicitation process [14, 15]. Consistency assurance is undertaken in [16].

Other RE activities where recommenders systems are used are requirements management and negotiation. Regarding requirements management, recommender systems complement feature requests systems in tasks such as grouping forums to avoid parallel discussions on the same topic [17] and managing changes [18]. In negotiation, recommendations are used to support prioritization and triaging of requirements [19], as well group decisions making [20].

As presented in Table 1, the proposed approaches focus on a specific RE task but do not support the needs of stakeholders in different requirements-related tasks through the different RE stages. OpenReq aims to do so by providing an open source tool that will be available for requirements analysts.

Table 1. Recommender systems for RE
E & S = Elicitation and Specification, NA = Not Applicable, NS = Not Stated

Id	RE stage	Focus	Req. types	Tool
[11]	E & S	Requirements discovery	Sustainability	Yes
[12]	E & S	Requirements discovery	Non-functional	NS
[13]	E & S	Requirements discovery	All	NS
[14]	E & S	Reqs. discovery, Stakeholders identification	All	NS
[15]	E & S	Stakeholders identification	NA	NS
[16]	E & S	Quality assurance	All	NS
[17]	Management	Feature requests clustering	All	Yes
[18]	Management	Feature requests changes	All	Yes
[19]	Negotiation	Prioritization, Triage	All	Yes
[20]	Negotiation	Group decision support	All	Yes
Open Req	E & S, Analysis, Management, Negotiation	Requirements discovery, Stakeholders identification, Quality assurance, Group decision support	All	Yes

3 OpenReq's Approach to Personal Recommendation in RE

One of the goals of the OpenReq system is assisting stakeholders with personal recommendations during the RE process. Specifically, the personal recommendations will take place during the requirements elicitation, specification and analysis stages. As shown in Figure 1, recommendations for individual stakeholders will be related to the screening and recommendation of relevant requirements, to the improvement of requirements quality, to the prediction of requirements properties, and to the identification of relevant stakeholders. These recommendations will be context-aware, meaning that the current context of the stakeholders will be taken into account when providing the recommendations.

The OpenReq approach will achieve each of the personal recommendation task, presented in Figure 1, as follows:

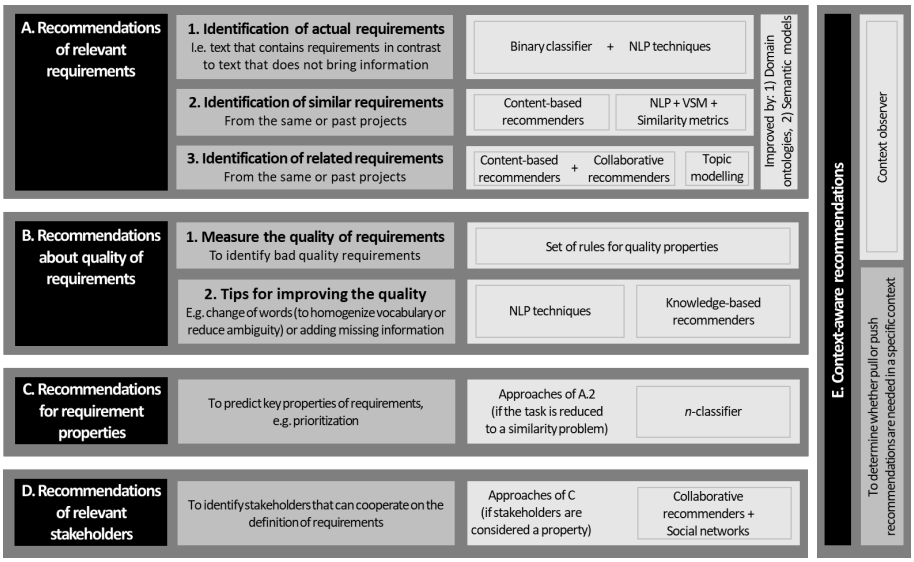

Fig. 1. Personal recommendations in OpenReq

A. Recommendations of relevant requirements. The recommendations in this group are related to the:

1. *Identification of actual requirements*, i.e. to recognize text that contains "actual" requirements in contrast to the one that does not bring valuable information. From the technical perspective, a binary classifier [21] will be used in combination with some basic Natural Language Processing (NLP) techniques [22] (to identify common words in requirements such as "must", "have/has to", etc.).
2. *Identification of similar requirements*, in the same project or from previous ones. For this purpose, different approaches will be investigated. The first one is based on using content-based recommenders [23] that, taking into account the requirements and current project metadata, will be able to recommend re-

quirements that have already been defined in previous projects. The second approach is based on using NLP techniques (such as tokenization, stemming, and stop words removal) to represent each requirement using a Vector Space Model (VSM) [24] (which represents a text, in this case a requirement, as a vector in a multi-dimensional space, where each dimension of the space corresponds to a term) and finally compute the similarity among two requirements using measures like Cosine, Dice or Jaccard [24].

3. *Identification of related requirements*. Here, content-based (based on semantic and text-based similarities) and collaborative recommendation approaches (based on context information) will be used [23] taking into account the available requirement metadata. Another approach is based on Topic Modelling [25], which can be used to associate a label (i.e., a topic) to a requirement or a subset of them, and then cluster the requirements in groups of related ones. The clustering can be done at different level of granularities (e.g., a hierarchy of topics), achieving different levels of relatedness.

The previous identification task can be improved by the use of: a) domain ontologies, especially to identify synonyms that are domain-specific, and b) semantic models, to specify how requirements are semantically related among each other.

B. Recommendations for improving the quality of requirements. In this case, the recommendations are related to:

1. *Measure the quality of requirements* to identify bad quality requirements. A set of rules reflecting quality properties will be identified from existing related work (e.g., [26]) and adapted to OpenReq. These rules can then be used against requirements to check their quality, compounding them to calculate a quality score.
2. *Tips for improving the quality of requirements*, for instance, changing some wording (to standardize the vocabulary or reduce the ambiguity of requirements) or adding missing information. Simple word lists can be used to identify weak words (e.g., terms that are considered ambiguous, such as "sometimes" or "usually") and thesauruses can be used to identify alternative terms. For more complex tips, knowledge-based recommenders [23] can point out open tasks on the basis of previously defined rules. These tips can be, for instance *"additional meta-information needed"*, "*text should be extended"*, or *"additional users should review this requirement"*.

C. Recommendations for requirement properties. The focus here is to predict key properties of requirements, such as priority. To this end, the recommender will match requirements at hand with those that have already been defined in past or ongoing projects. Therefore, this case is reduced to point A.2, where three approaches are possible: one based on content-based recommendations; one based on NLP techniques (VSM and similarity measures), and one based on topic modelling. Alternatively, the recommendation of each requirement property can be seen as a classification task with *n* classes where machine learning approaches can be used to assign a value to the specific property of a requirement.

D. Recommendations of relevant stakeholders. Here, the recommendations aim at detecting stakeholders who can cooperate on the definition of requirements. The first approach is based on the assumption that stakeholders can be seen as one property of a

requirement; therefore, the same approaches presented in C could be used. The second approach is based on using a collaborative filtering recommender that, by analysing existing social networks (e.g., typical roles of stakeholders in past software projects), individual strengths of stakeholders (e.g., topics the stakeholder has contributed to and related topics the stakeholder could be interested in), and personal availabilities, will create a user profile that will be used to match requirements to stakeholders. This collaborative filtering can be improved with weighting schemes [27] to requirements topics in which stakeholders have interest in or are experts on.

E. Context-aware recommendations. We aim to determine whether pull recommendations (automatically delivered to stakeholders) or push recommendations (stakeholders trigger them when needed) can be applied in a specific context. A *context observer* component will be integrated in OpenReq, which will take into account contextual information to decide in a personalised way when, what, and in which way recommendations will be delivered. For instance, we can use the history of the stakeholder activities within OpenReq to know if s/he is too busy to receive notifications on tips related to requirements quality.

4 Conclusions

In this paper, we provide an overview of the personal recommendations that will be supported by OpenReq to improve and speed-up the RE process, as well as the first considerations about how such recommendations can be implemented using a combination of state-of-the-art recommender systems and NLP techniques. Within the scope of OpenReq, we will focus on the development of a new RE solution for the systematic improvement of related development, maintenance, quality assurance, and decision processes, and also on integrating these improvements as extensions of existing RE tools. We expect to have a first prototype of the tool by January 2018.

Acknowledgments

The work presented in this paper has been conducted within the scope of the Horizon 2020 project OpenReq which is supported by the European Union under the Grant Nr. 732463.

References

1. Hofmann, H., Lehner., F.: Requirements engineering as a success factor in software projects. IEEE Software, 18(4). pp.58–66 (2001).
2. Mens, K., Lozano, A.: Source Code-Based Recommendation Systems. In: Robillard, M.P., Maalej, W., Walker, R.J., and Zimmermann, T. (eds.) Recommendation Systems in Software Engineering. pp. 93–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2014).
3. Mobasher, B., Cleland-Huang, J.: Recommender systems in requirements engineering. Ai Magazine, 32(3), pp. 81–89 (2011).
4. Intelligent Recommendation and Decision Technologies for Community-Driven Requirements Engineering (Horizon 2020 Project, https://www.openreq.org).

5. Robillard, M., Walker, R., Zimmermann, T.: Recommendation Systems for Software Engineering. IEEE Software. 27, 80–86 (2010).
6. Antoniol, G., Ayari, K., Di Penta, M., Khomh, F., Guéhéneuc, Y.-G.: Is It a Bug or an Enhancement?: A Text-based Approach to Classify Change Requests. CASCON 2008.
7. Nagwani, N.K., Verma, S.: Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes. ICT-KE 2011.
8. Yu, L., Tsai, W.-T., Zhao, W., Wu, F.: Predicting Defect Priority Based on Neural Networks. In: Cao, L., Zhong, J., and Feng, Y. ADMA 2010.
9. Tu, Z., et al.: Gig Services Recommendation Method for Fuzzy Requirement Description. ICWS 2017.
10. Felfernig, A., et al.: An Overview of Recommender Systems in Requirements Engineering. In: Maalej, W. and Thurimella, A.K. (eds.) Managing Requirements Knowledge. pp. 315–332. Springer Berlin Heidelberg, Berlin, Heidelberg (2013).
11. Roher, K., Richardson, D.: A proposed recommender system for eliciting software sustainability requirements. USER 2013.
12. Danylenko, A., Löwe, W.: Context-aware recommender systems for non-functional requirements. RSSE 2012.
13. Kumar, M., Ajmeri, N., Ghaisas, S.: Towards Knowledge Assisted Agile Requirements Evolution. RSSE 2010.
14. A. Finkelstein, Soo Ling Lim: StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. IEEE Transactions on Software Engineering. 38, 707–735 (2012).
15. Castro-Herrera, C., Cleland-Huang, J.: Utilizing Recommender Systems to Support Software Requirements Elicitation. RSSE 2010.
16. Felfernig, A., Friedrich, G., Schubert, M., Mandl, M., Mairitsch, M., Teppan, E.: Plausible Repairs for Inconsistent Requirements. IJCAI 2009.
17. Cleland-Huang, J., Dumitru, H., Duan, C., Castro-Herrera, C.: Automated Support for Managing Feature Requests in Open Forums. Commun. ACM. 52, 68–74 (2009).
18. Garcia, J.E., Paiva, A.C.R.: REQAnalytics: A recommender system for requirements maintenance. In: International Journal of Software Engineering and its Application. pp. 129-140 (2016).
19. Duan, C., Laurent, P., Cleland-Huang, J., Kwiatkowski, C.: Towards automated requirements prioritization and triage. Requirements Engineering. 14, 73–89 (2009).
20. Felfernig, A., Zehentner, C., Ninaus, G., Grabner, H., Maalej, W., Pagano, D. : Group Decision Support for Requirements Negotiation. UMAP 2012.
21. Winkler, J., Vogelsang, A.: Automatic Classification of Requirements Based on Convolutional Neural Networks. REW 2016.
22. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP Natural Language Processing Toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014).
23. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press, New York, NY, USA (2010).
24. Falessi, D., Cantone, G., Canfora, G.: A Comprehensive Characterization of NLP Techniques for Identifying Equivalent Requirements. ESEM 2010.
25. Chien, J.T.: Hierarchical Theme and Topic Modeling. IEEE Transactions on Neural Networks and Learning Systems. 27, 565–578 (2016).
26. Bucchiarone, A., Gnesi, S., Lami, G., Trentanni, G., Fantechi, A.: QuARS Express - A Tool Demonstration. ASE 2008.
27. Said, A., Jain, B.J., Albayrak, S.: Analyzing Weighting Schemes in Collaborative Filtering: Cold Start, Post Cold Start and Power Users. SAC 2012.