Recommender Systems in Requirements Engineering

Bamshad Mobasher and Jane Cleland-Huang

■ *Requirements engineering in large-scale industrial, government, and international projects can be a highly complex process involving thousands or even hundreds of thousands of potentially distributed stakeholders. The process can result in massive amounts of noisy and semistructured data that must be analyzed and distilled in order to extract useful requirements. As a result, many human-intensive tasks in requirements elicitation, analysis, and management processes can be augmented and supported through the use of recommender system and machine-learning techniques. In this article we describe several areas in which recommendation technologies have been applied to the requirements engineering domain, namely stakeholder identification, domain analysis, requirements elicitation, and decision support across several requirements analysis and prioritization tasks. We also highlight ongoing challenges and opportunities for applying recommender systems in the requirements engineering domain.*

Requirements engineering is the practice of eliciting, analyzing, prioritizing, negotiating, and specifying the requirements for a software-intensive system (Robertson and Robertson 1999). These activities engage various stakeholders in the task of identifying and producing an agreed-upon set of requirements that clearly specify the functionality, behavior, and constraints of the proposed system. The importance of the requirements engineering task is illustrated by several studies, which have shown that requirements-related issues, such as poorly specified requirements (Leffingwell 1997) and incomplete and changing requirements and specifications, are the root cause of many failed projects. To address these problems, researchers and practitioners have developed processes for identifying relevant stakeholders; discovering their needs, wants, and desires for the system; prioritizing and negotiating requirements; and specifying them in understandable, measurable, and testable ways (Robertson and Robertson 1999). In this article we focus particularly on recent efforts to use machine learning and recommender systems technologies for automating requirements engineering processes and enabling stakeholder and designer decision support.

In general, the task of a recommender system in any domain is to identify items of interest to a given user of which that user may otherwise be unaware. The recommendation problem is typically formulated as a prediction task in which a predictive model is built according to prior observations and then used in conjunction with the dynamic profile of a new user to predict the level of interest by that user on a target item. Recommender systems generally fall into the three broad categories of content-based systems, which make recommendations based on content similarity between user profiles and the semantic descriptions of items (Pazzani and Billsus 2007); collaborative-filtering systems, which make recommendations by examining implicitly or explicitly derived preferences of a user on items and identifying other users with similar interests (Schafer et al. 2007); and knowledge-based systems, which make recommendations based on heuristic rules that depend on prior knowledge of the users or the domain (Felfernig et al. 2006). The output of a recommender system is usually a set of items that were previously unknown to the user and that scored the highest predicted interest values.

Maalej and Thurimella (2009) outlined a preliminary research agenda for recommender systems within the requirements engineering domain. They proposed several ways in which recommender technologies could help stakeholders create, analyze, specify, and manage requirements. These included using a recommender system to identify relevant background information for a given requirement, to recommend suitable templates for a given task, to suggest opportunities for saving time and effort by reusing requirements, for identifying pertinent design rationales related to specific types of requirements, and for proposing traceability relationships between various requirements and design elements. Although many of these proposed methods are based on heuristic approaches, more recent work in the area has focused on recommender systems that go beyond heuristic rules or search mechanisms and focus on more traditional machine-learning techniques.

Recommender systems have been used in the requirements engineering domain to address three specific kinds of problems. The first is to identify potential stakeholders for a given project. The second is to discover user requirements or features for a system, and the third is to provide support for requirements-related decision making such as requirements prioritization. Although current work is still in the early experimental stage, various approaches have already been evaluated on large-scale simulations using data from open-source forums, reenactments of large-scale projects with thousands of stakeholders, and small-scale proof-of-concept software development projects. Results from these studies show significant promise for transference to real industrial projects in the future.

Stakeholder Identification and Analysis

Stakeholder analysis is a critical component of every requirements engineering process. It involves identifying a set of stakeholders capable of providing a relatively complete and accurate description of the product under development. Although it is fairly straightforward to identify an initial group of stakeholders, it is far more difficult to make sure that all important stakeholder groups are represented by collaborative, responsible, authorized, committed, and knowledgeable stakeholders (Boehm and Turner 2004). There are two primary recommendation techniques that have been used effectively in this area. The first utilizes social networking and crowd sourcing to identify and recommend stakeholders for a project. The second is a hybrid recommender system that recommends stakeholders for particular discussion threads in online forums and that recommends discussion threads to individual stakeholders.

Discovering Project Stakeholders

Lim and others developed a recommender tool named Stakesource, which harnesses the power of crowd sourcing to discover project stakeholders (Lim, Quercia, and Finkelstein 2010a and 2010b). First an initial set of stakeholders is identified and asked to refer other stakeholders whom they consider important for the project. These stakeholders in turn are asked to refer additional people. The process continues until few additional stakeholders are discovered. At that time, a social network graph is constructed in which nodes represent candidate stakeholders and arcs represent referrals. Social network analysis techniques are then used to compute salience measures and to rank stakeholders according to their potential importance to the project.

Lim, Quercia, and Finkelstein also evaluated several different salience metrics; however, four of them stand out as most useful for ranking the potential importance of stakeholders. *Betweenness centrality* measures a stakeholder *S*'s ability to serve as a broker between other groups of stakeholders by summing the shortest paths between pairs of stakeholders that pass through *S* (Brandes 2001). *Load centrality* measures the influence stakeholder *S* has on the transfer of information between each pair of stakeholders. It is computed by passing one unit of information between each pair of stakeholders along the shortest path. When *n* paths are equal to the shortest length, the unit is divided by *n* and passed equally along each of the paths. Load centrality is then computed for each stakeholder *S* by summing the units, or fractions of units, that pass through it during stakeholder communication. These two metrics are used to identify core

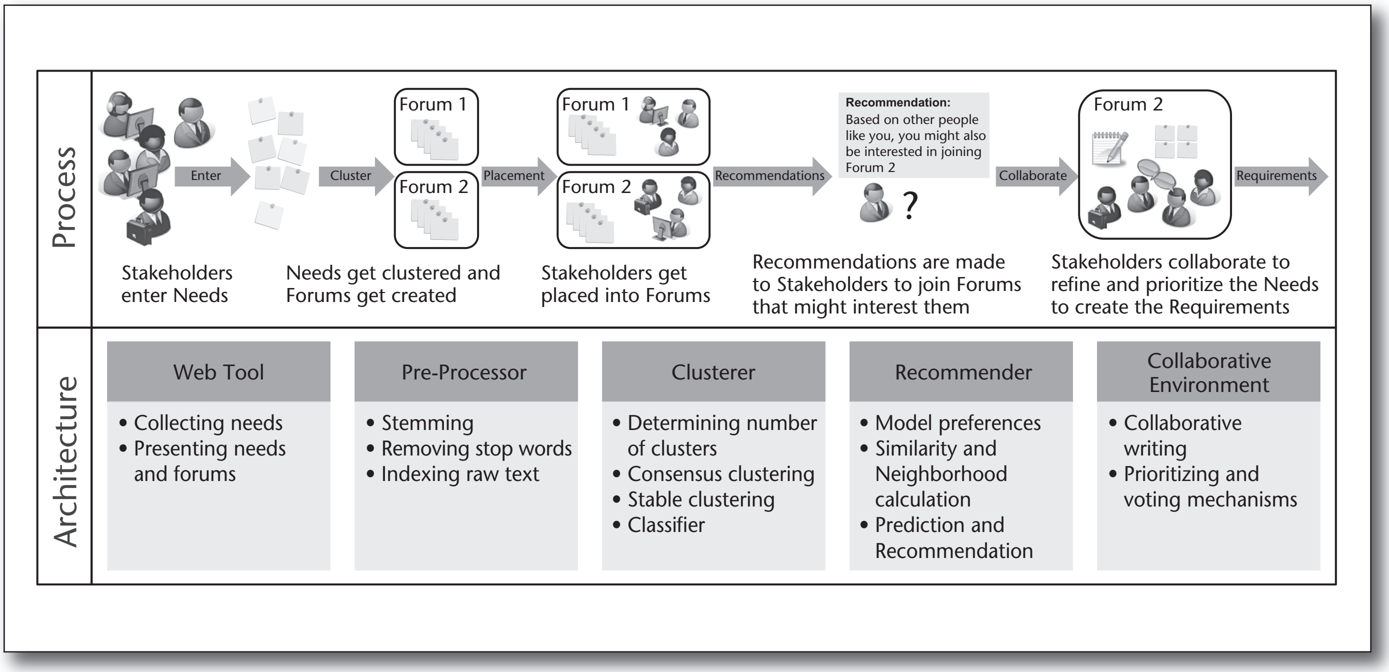

Process
Enter
Cluster
Forum 1
Forum 2
Placement
Recommendations
Recommendation: Based on other people like you, you might also be interested in joining Forum 2
?
Collaborate
Requirements
Stakeholders enter Needs
Needs get clustered and Forums get created
Stakeholders get placed into Forums
Recommendations are made to Stakeholders to join Forums that might interest them
Stakeholders collaborate to refine and prioritize the Needs to create the Requirements
Architecture
Web Tool
• Collecting needs
• Presenting needs and forums
Pre-Processor
• Stemming
• Removing stop words
• Indexing raw text
Clusterer
• Determining number of clusters
• Consensus clustering
• Stable clustering
• Classifier
Recommender
• Model preferences
• Similarity and Neighborhood calculation
• Prediction and Recommendation
Collaborative Environment
• Collaborative writing
• Prioritizing and voting mechanisms

Figure 1. Forum Recommendation Process (Castro-Herrera et al. 2009).

project stakeholders. Two other metrics are used to identify and recommend stakeholders who are significantly influenced by a project. These are *PageRank* and *in-degree* metrics, both of which measure the degree to which a stakeholder is authoritative (locally in the case of in-degree and globally in the case of PageRank). Salience measures are computed for each stakeholder, and recommendations are made for any stakeholders scoring above a certain salience threshold.

Recommending Stakeholders in Forums

In distributed projects, it is often infeasible to bring stakeholders together on a regular basis, and as a result, numerous organizations are adopting the practice of using wikis and forums to gather an initial set of feature requests and then to extract a more formal set of requirements from those requests (Cleland-Huang et al. 2009). Although such tools allow more stakeholders to engage in the requirements elicitation process, they also introduce challenges related to information overload, noisy and redundant data, incomplete discussions, dissenting opinions, and frustrated stakeholders who often receive no feedback for their contributions. Fortunately, many of these problems can be addressed through using data-mining techniques to keep feature requests organized in cohesive and nonredundant threads (Cleland-Huang et al. 2009) and also through using recommender systems to keep stakeholders informed of interesting topics and to help project managers find appropriate stakeholders to engage in open discussions.

The organizer and promoter of collaborative ideas (OPCI) approach, developed by Castro-Herrera, Cleland-Huang, and Mobasher (2009) and Castro-Herrera, Duan, Cleland-Huang, and Mobasher (2009) is depicted in figure 1. While OPCI can be used to support any online forum, it was originally designed to support the requirements engineering process (Castro-Herrera et al. 2009). OPCI's recommender system serves two purposes: (1) recommending specific topics to individual stakeholders in order fully to engage each stakeholder in the requirements elicitation process, and (2) recommending stakeholders for specific topics in order to ensure that each topic has sufficient coverage.

In OPCI, the initial feature requests are represented as weighted term vectors. Specifically, each feature request, *x,* is represented as a vector of terms $(x_1, \cdots, x_W)$ where x_i is the normalized weight associated with term t_i in *x,* and *W* represents the total number of unique terms in the entire set of feature requests. This normalized weight is calculated using a standard information-retrieval technique known as *tf-idf*, where *tf* represents the frequency of term *t* in the feature request and *idf* represents the inverse document frequency. The Cosine similarity between two feature request vectors $a = (a_i, \cdots, a_W)$ and $b = (b_i, \cdots, b_W)$ is then computed as:

Users	Known Data				Threads			
	K_1	K_2	…	K_i	T_1	T_2	…	T_j
U_1		1			1	1		
U_2	1				1			
U_3	1			1		1		1
…								
U_k		1		1	1			1

Table 1. Users' Profiles with Additional Known Data.

$$sim(a,b) = \frac{\sum_{i=1}^{W} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{W} a_i^2 \cdot \sum_{i=1}^{W} b_i^2}} \quad .$$

Intuitively this formula returns high similarity scores between two feature requests that share a significant number of relatively rare terms.

OPCI uses a consensus-based clustering approach to merge similar discussion threads and to extract topics that are dispersed across multiple threads into new discussion themes (Castro-Herrera et al. 2009; Duan, Cleland-Huang, and Mobasher 2008). Stakeholders who have contributed ideas to the initial threads or the reconstructed threads are assumed to be interested in the topics of those threads.

An initial stakeholder profile is created for each stakeholder according to the discussion threads that the stakeholder participates in, and also according to known information such as the role of the stakeholder in the project or the stakeholder's subject matter expertise. These profiles form a profiles matrix $M := (m_{i,j})_{U \times T}$, where U is the number of stakeholders, T is the number of discussion threads, and $M[i, j]$ is the weight representing the stakeholder's interest in the discussion thread (derived as part of the clustering process that generates the initial discussion forums). The profiles are used to feed a collaborative recommender system (Schafer et al. 2007) based on the standard *k*-nearest-neighbor (*k*NN) strategy (Castro-Herrera et al. 2009). It computes similarities among stakeholder profiles in order to create a "neighborhood" for a targeted stakeholder. The resulting neighborhood is then used to predict the interest that the targeted stakeholder will have in any particular topic. These recommendations serve to foster the cross-pollination of ideas, prevent the problem of missed or stagnant discussions, and promote topics that address key areas of the system that have so far not been explored.

Two variations of the standard *k*NN algorithm were explored (Castro-Herrera, Cleland-Huang, and Mobasher 2009). In the first variation, user profiles were modeled using a binary matrix in which a 1 represented membership in a thread, and a 0 represented nonmembership. This representation was shown experimentally to be a good fit for the forum environment since users do not normally specify degrees of interest in forums or posts. The second variation involved extending the profiles with features corresponding to semantic knowledge about the domain or prior knowledge about the users. Examples of additional known data available for the requirements elicitation domain include: roles that the stakeholders play in the project, interests in key features of the system, and interests in cross-cutting concerns such as security or usability. Formally, the profiles matrix M := $(m_{i,j})_{U \times (K+T)}$ now includes K columns that capture a user's interest in a known metadata element. Table 1 shows a schematic example of the enhanced matrix.

The binary recommender was shown to perform relatively well, and in all four of the datasets demonstrated the ability to recommend back approximately 25 percent to 50 percent of the known interests in the top 10 recommendations (Castro-Herrera, Cleland-Huang, and Mobasher 2009). The experiments also clearly demonstrated the usefulness of incorporating prior knowledge into the users' profiles in order to enrich knowledge of the users' interests in early stages of the forum. Recommendations that incorporated known data were able to recommend back approximately 70 percent to 90 percent of known interests in the top 10 recommendations.

Figure 2 shows hit ratio values for four different approaches including standard *k*NN, binary representation, binary with known data, and the baseline case in which items are recommended in random order. Hit ratio represents the probability that a recommended item among the top n items is selected by the user. The *railway* data set represents more than 1600 feature requests created by extracting requirements from the public specifications of two large-scale railway systems, the Canadian Rail Operating Rules and the Standard Code of Operating rules published by the Association of American Railroads. In this data set, known data was derived from the users' individual roles in the project.

Recommender Systems in Requirements Elicitation

Recommender systems have also been used to recommend new ideas, features, and requirements during the initial domain analysis and elicitation process. Most projects include both a domain analysis phase, in which project stakeholders analyze related software systems to identify, organize, and represent features common to systems within

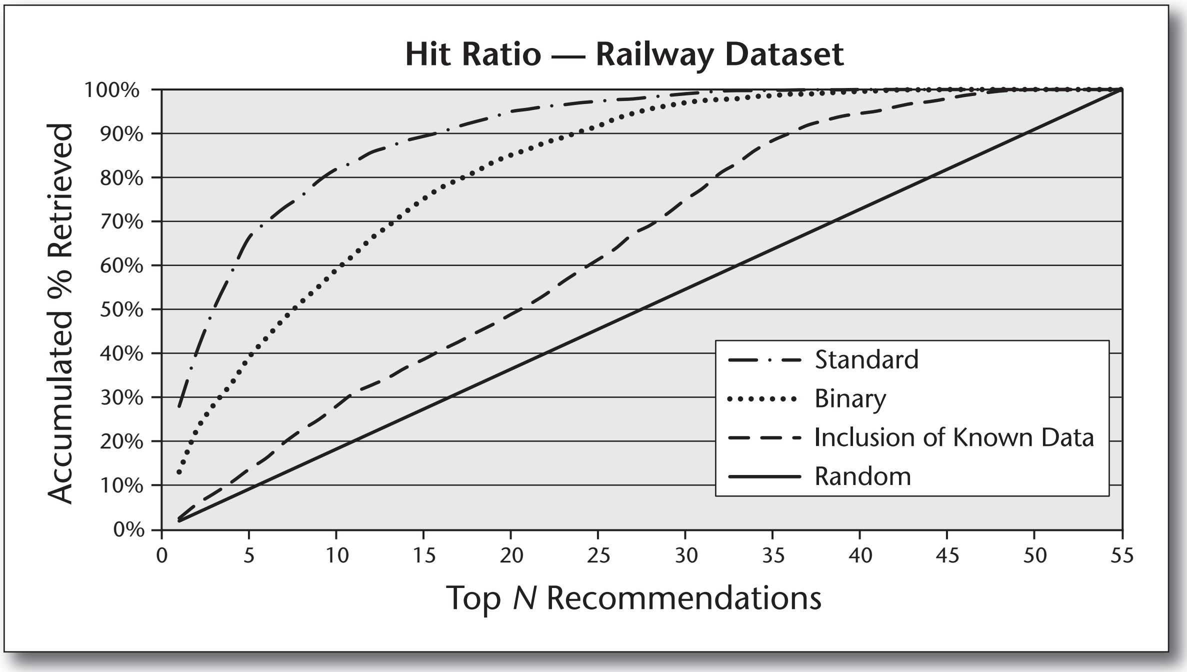


Figure 2. Hit Ratio Graph.

a domain (Arango and Prieto-Diaz 1989), and a focused requirements elicitation phase, in which the actual project requirements are elicited. Recommender systems have been used to support each of these individual tasks.

Recommending Requirements

Kumar, Ajmeri, and Ghaisas (2010), proposed a conversation-style recommender for use in agile development in which the system recommends a set of basic user stories (that is, brief descriptions of functionality) and tasks in response to an initial project description. The approach utilizes different ontologies that represent the environmental context, agile requirements, general requirements, and the problem domain. The authors provide an initial validation of their approach for an agile project and illustratively show that it generates useful recommendations. In related work, Romero-Mariona, Ziv, and Richardson (2008) developed an ad hoc knowledge-based system that takes initial goals of the system and recommends relevant models and requirements in the form of misuse cases and abuse scenarios, antimodels, and security-related goal models.

Recommending Features

The identification and timely reuse of domain assets in early stages of the requirements process can potentially reduce development costs, shorten time to market, improve quality of the delivered system, and increase product competitiveness. Existing domain analysis techniques primarily focus on either manual or semiautomated analyses of a small number of requirements specifications or product brochures in order to identify and extract reusable features. As a result, domain analysis tends to be quite labor intensive, dependent upon the expertise of available analysts or subject matter experts, and constrained by the availability of requirements specifications from previous related projects.

Dumitru et al. (2011) developed a novel recommender system that addresses these limitations by mining raw feature descriptions from online project repositories such as SoftPedia, clustering the descriptions into features, generating a feature model for a given product category, and then using this feature model to recommend features and combinations of features that might be relevant for a new product under development. The approach uses association rule mining (Agrawal, Imielinski, and Swami 1993) to identify affinities among product features and the *k*NN machine-learning strategy to identify similar products and make predictions about the existence of features not previously identified (figure 3).

More formally, once feature descriptions have been mined and clustered, a product-by-feature matrix is generated, $M := (m_{i,j})_{PxF}$, where P represents the number of products and F the number of identified features. To generate a recommendation for a new item, a new profile is created by parsing an initial product description and then using cosine similarity to match the terms in that description with features in the matrix. In many

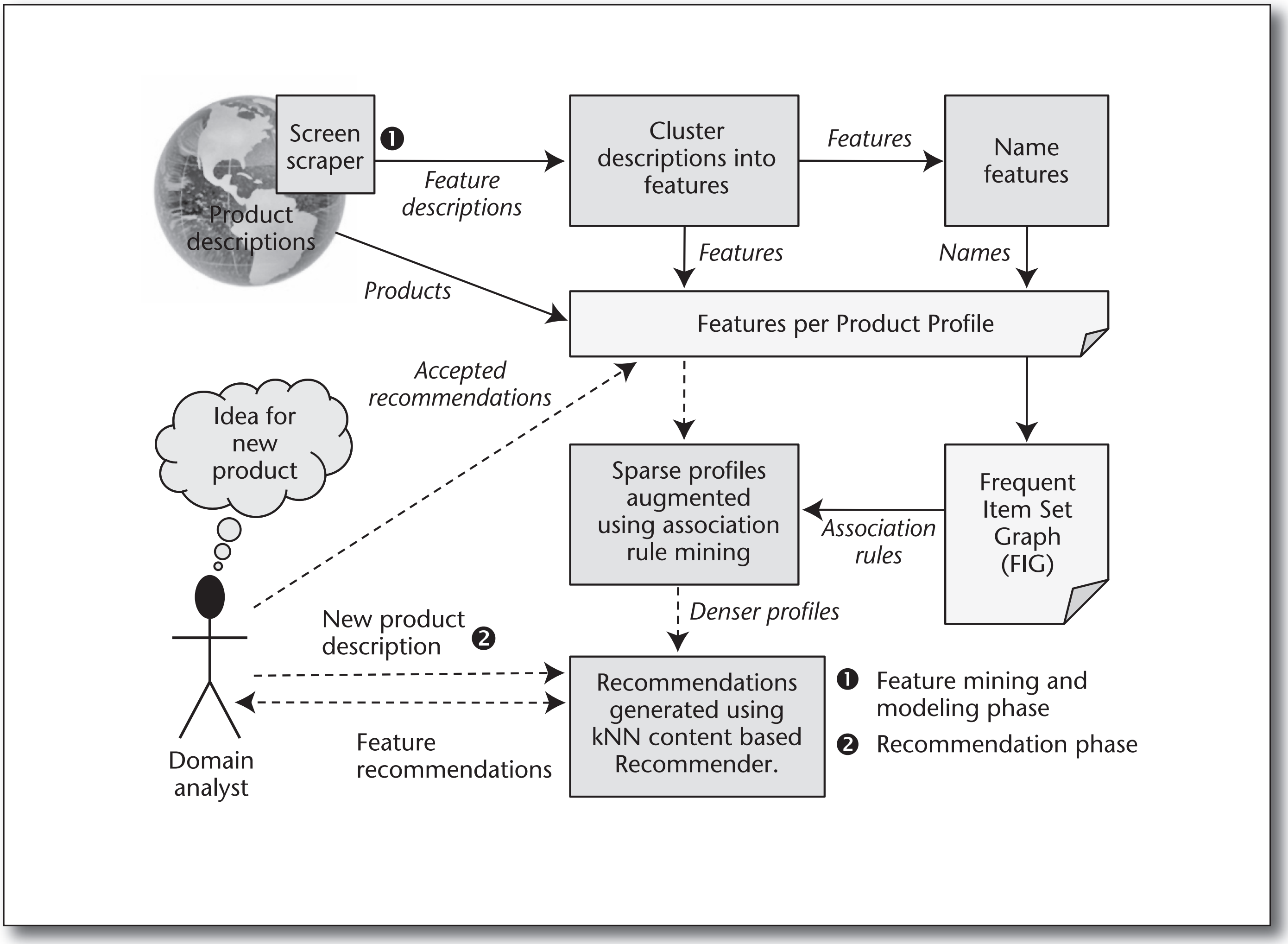

Screen scraper
❶
Feature descriptions
Product descriptions
Products
Cluster descriptions into features
Features
Name features
Features
Names
Features per Product Profile
Accepted recommendations
Idea for new product
Sparse profiles augmented using association rule mining
Association rules
Frequent Item Set Graph (FIG)
Denser profiles
New product description ❷
Feature recommendations
Recommendations generated using kNN content based Recommender.
Domain analyst
❶ Feature mining and modeling phase
❷ Recommendation phase

Figure 3. Feature Extraction and Recommendations.

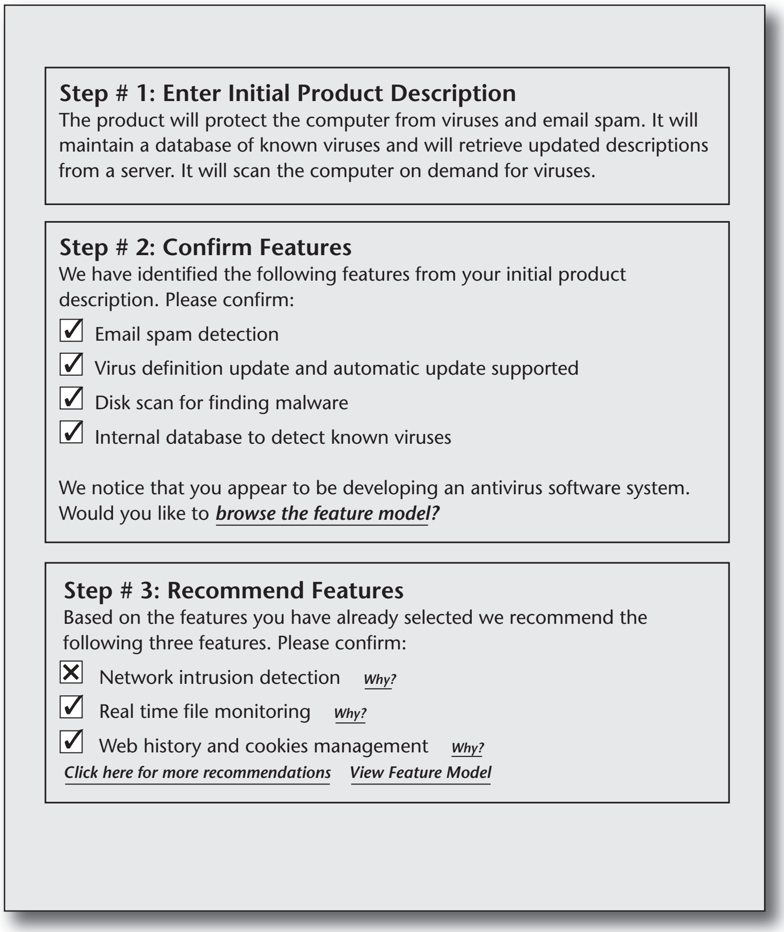
cases, the initial product profiles may be rather sparse and unable to provide sufficient information to generate high-quality recommendations using more traditional techniques such as content-based or collaborative filtering.

Given a set of product profiles P, a set of features $F = \{F_1, F_2, \cdots, F_k\}$, and a feature set $fs \subseteq F$, let $P_{fs} \subseteq P$ be the set of products that have all the features in fs. The *support* of the feature set fs is defined as $\sigma(fs) = |P_{fs}| / |P|$. Feature sets that satisfy a predefined support threshold are referred to as *frequent feature sets*. An association rule r is expressed in the form $X \Longrightarrow Y(\sigma_r, \alpha_r)$, where X and Y are feature sets, σ_r is the support of the feature set $X \cup Y$, and α_r is the *confidence* for the rule r given by $\sigma(X \cup Y) / \sigma(Y)$.

After generating the association rules, new features can be recommended based on an initial product profile by finding all the matching rules. In order to reduce the search time, the frequent feature sets are stored in a directed acyclic graph, called a frequent itemset graph, or FIG (Mobasher et al. 2001; Sandvig, Mobasher, and Burke 2007). The graph is organized into levels from 0 to l, where l is the maximum size among all discovered frequent feature sets. Each node at depth d in the graph corresponds to a feature set I of size d and is linked to feature sets of size $d + 1$ that contain I at the next level. Given a product profile with feature set fs, a depth-first search is performed on the graph to level $|fs|$ in the graph. If a match is found, then the children of the matching node are used to generate candidate recommendations; and each child of this node corresponds to a frequent itemset $fs \cup \{r\}$. The feature r will be added to the recommendation list if the confidence of the rule exceeds a prespecified minimum threshold. The same procedure will be repeated for all the subsets of the feature set f.

The association-based recommender tends to generate a small, yet relatively accurate set of features. These recommended features are automatically added to the initial product profile. Then a standard *k*NN learning strategy is used to compute a feature-based similarity measure between the new product and each existing product. The top *k* most similar products are considered *neighbors* of the new product, and this information is used to recommend features that might be relevant to the current analysis. In general, prediction scores will be computed for each candidate feature, and the features with highest predictions will be recommended.

Figure 4 illustrates a feature recommendation scenario for an antivirus product. An initial product description is mapped to four features in the recommender's knowledge base related to *spam detection, disk scans, virus definitions*, and *virus databases*, which serve as seeds for generating feature recommendations. User feedback is fed back into the system and used to drive future recommendations.



Step # 1: Enter Initial Product Description
The product will protect the computer from viruses and email spam. It will maintain a database of known viruses and will retrieve updated descriptions from a server. It will scan the computer on demand for viruses.

Step # 2: Confirm Features
We have identified the following features from your initial product description. Please confirm:

- ☑ Email spam detection
- ☑ Virus definition update and automatic update supported
- ☑ Disk scan for finding malware
- ☑ Internal database to detect known viruses

We notice that you appear to be developing an antivirus software system. Would you like to ***browse the feature model*?**

Step # 3: Recommend Features
Based on the features you have already selected we recommend the following three features. Please confirm:

- ☒ Network intrusion detection *Why?*
- ☑ Real time file monitoring *Why?*
- ☑ Web history and cookies management *Why?*

Click here for more recommendations *View Feature Model*

Figure 4. Example of Feature Recommendations.

Decision Support

The final emerging area of recommender systems provides support for decision making in the requirements engineering process. More specifically, it is designed to address the four areas of decision making identified by Regnel et al. (2001), namely evaluating the quality of requirements, selecting requirements for the next release, classifying requirements by type, and making estimation decisions. Felfernig et al. (2010) have developed a recommender system named IntelliReq, which provides support for individual and group decisions. IntelliReq is designed to capture individual opinions related to requirements qualities, estimations, and priorities and then to use utility functions and a recommender system to help resolve conflicts, recommend requirements relevant to the current task, highlight possible requirement dependencies, and finally to recommend ways in which to maximize agreement between stakeholders. Parts of these scenarios have already been integrated in the IntelliReq decision support environment utilizing collaborative filtering, content-based filtering, knowledge-based recommendation, and group-based recommendation algorithms. In its current form IntelliReq utilizes group recommendation algorithms to support requirements negotiation.

Conclusions

The ongoing shift in the requirements engineering domain toward large-scale, distributed, collaborative, and tool-supported environments offers numerous opportunities for supporting human-intensive tasks through the use of recommendation technologies. In this article we have described three active areas of recommender system research related to identifying stakeholders, generating requirements and features, and providing support for several decision-making tasks. The work reported in these areas shows significant promise for the effective use of recommender systems in the requirements engineering domain.

However, making recommendations as a natural part of a complex requirements engineering task is nontrivial and introduces numerous human-computer interaction factors that need to be taken into consideration. For example, we need to ensure that recommendations are timely and contextualized and, furthermore, that they actually help the requirements engineer perform the task more effectively.

In many respects it is harder to build a successful recommender system in the requirements engineering domain than it is in a more traditional e-commerce context. Whereas a user purchasing books online may be pleased with a recommendation for a single interesting book, a domain analyst may not find a recommender system useful or trustworthy unless the system can successfully recommend all the basic and commonly occurring features in a targeted product. In certain cases, the role of the recommender system therefore transitions from the goal of generating serendipitous ideas and suggestions of potentially interesting items to the goal of making the complete and accurate set of recommendations needed to support a complex requirements engineering task.

Despite these difficulties, the various applications of recommender systems described in this article have highlighted the potential usefulness in the requirements engineering domain. Future work is expected to develop advanced techniques, algorithms, and processes for utilizing recommender systems in the requirements engineering domain and to explore the usefulness of the various approaches within the context of real software projects.

Acknowledgments

The work described in this article was partially funded by grant III:0916852 from the National Science Foundation.

References

Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216. New York: Association for Computing Machinery.

Arango, G., and Prieto-Diaz, R. 1989. *Domain Analysis: Acquisition of Reusable Information for Software Construction*. Los Alamitos, CA: IEEE Computer Society.

Boehm, B. W., and Turner, R. 2004. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. In *Proceedings of the 26th International Conference on Software Engineering*, 718–719. New York: Association for Computing Machinery.

Brandes, U. 2001. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology* 25(2): 163–177.

Castro-Herrera, C.; Cleland-Huang, J.; and Mobasher, B. 2009. Enhancing Stakeholder Profiles to Improve Recommendations in Online Requirements Elicitation. In *Proceedings of the 17th IEEE International Requirements Engineering Conference*, 37–46. Los Alamitos, CA: IEEE Computer Society.

Castro-Herrera, C.; Duan, C.; Cleland-Huang, J.; and Mobasher, B. 2009. A Recommender System for Requirements Elicitation in Large-Scale Software Projects. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, 1419–1426. New York: Association for Computing Machinery.

Cleland-Huang, J.; Dumitru, H.; Duan, C.; and Castro-Herrera, C. 2009. Automated Support for Managing Feature Requests in Open Forums. *Communications of the ACM* 52(10): 68–74.

Duan, C.; Cleland-Huang, J.; and Mobasher, B. 2008. A Consensus Based Approach to Constrained Clustering of Software Requirements. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management,* 1073–1082. New York: Association for Computing Machinery.

Dumitru, H.; Gibiec, M.; Hariri, N.; Cleland-Huang, J.; Mobasher, B.; Castro-Herrera, C.; and Mirakhorli, M. 2011. On-Demand Feature Recommendations Derived from Mining Public Product Descriptions. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (1)*. New York: Association for Computing Machinery.

Felfernig, A.; Friedrich, G.; Jannach, D.; and Zanker, M. 2006. An Integrated Environment for the Development of Knowledge-Based Recommender Applications. *International Journal of Electronic Commerce* 11(2): 11–34.

Felfernig, A.; Schubert, M.; Mandl, M.; Ricci, F.; and Maalej, W. 2010. Recommendation and Decision Technologies for Requirements Engineering. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 11–15. New York: Association for Computing Machinery.

Kumar, M.; Ajmeri, N.; and Ghaisas, S. 2010. Towards Knowledge Assisted Agile Requirements Evolution. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 16–20. New York: Association for Computing Machinery.

Leffingwell, D. 1997. Calculating the Return on Investment from More Effective Requirements Management. *American Programmer* 10(4): 13–16.

Lim, S. L.; Quercia, D.; and Finkelstein, A. 2010a. Stakenet: Using Social Networks to Analyse the Stakeholders of Large-Scale Software Projects. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering* (1), 295–304. New York: Association for Computing Machinery.

Lim, S. L.; Quercia, D.; and Finkelstein, A. 2010b. Stakesource: Harnessing the Power of Crowdsourcing and Social Networks in Stakeholder Analysis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering* (2), 239–242. New York: Association for Computing Machinery.

Maalej, W., and Thurimella, A. K. 2009. Towards a Research Agenda for Recommendation Systems in Requirements Engineering. In *Proceedings of the Second International Workshop on Managing Requirements Knowledge,* 32–39. Los Alamitos, CA: IEEE Computer Society.

Mobasher, B.; Dai, H.; Luo, T.; and Nakagawa, M. 2001. Effective Personalization Based on Association Rule Discovery from Web Usage Data. In *Proceedings of the Third ACM Workshop on Web Information and Data Management*. New York: Association for Computing Machinery.

Pazzani, M. J., and Billsus, D. 2007. Content-Based Recommendation Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science 4321. Berlin: Springer-Verlag.

Regnell, B.; Paech, B.; Aurum, A.; Wohlin, C.; Dutoit, A.; and Dag, J. N. O. 2001. Requirements Mean Decisions! Research Issues for Understanding and Supporting Decision-Making in Requirements Engineering. Paper presented at the 1st Swedish Conference on Software Engineering Research and Practice, Ronneby, Sweden, 25–26 October.

Robertson, S., and Robertson, J. 1999. *Mastering the Requirements Process*. Boston, MA.: Addison-Wesley.

The 25th International FLAIRS Conference

Marco Island Hilton Beach Resort and Spa
Marco Island, Florida
May 23-25, 2012

The 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25) will be held May 23-25, 2012 at The Marco Island Hilton. In this silver anniversary for FLAIRS, we will proudly continue our tradition of presenting and discussing artificial intelligence research in a friendly atmosphere within a beautiful setting. Events will include invited speakers, special tracks, discussion panels, presentations of papers and posters, and will introduce a brand new session for impromptu presentations (stay tuned for that). As always, there will be a Best Paper award and a Best Poster award. In addition, because FLAIRS has a rich tradition of encouraging student authors, there will be a Best Student Paper award for the best paper written primarily by a student. Submissions are now invited for **full papers (6 pages)**, **short papers to be presented as a poster (4 pages)**, and **poster abstracts (250 words)**. The proceedings will be published by the AAAI. The conference is hosted by the Florida Artificial Intelligence Research Society in cooperation with AAAI.

Topics of interest are in all areas of AI, including but not limited to

- ***Foundations:*** Knowledge Representation, Cognitive Modeling, Perception, Reasoning & Programming, Search, Learning
- ***Architectures:*** Agents and Distributed AI, Intelligent User Interfaces, Natural Language Systems, Information Retrieval, Robotics
- ***Applications:*** Aviation and Aerospace, Education, Entertainment, Medicine, Management and Manufacturing, World Wide Web
- ***Implications:*** Philosophical Foundations, Social Impact and Ethics, Evaluation of AI Systems, Teaching AI
- ***Special Tracks:*** Numerous special tracks offer opportunities for focused interaction. All special track papers are published in the proceedings.

http://www.flairs-25.info/
In cooperation with the Association for the Advancement of Artificial Intelligence

General Chair
H. Chad Lane
lane@ict.usc.edu
USC Institute for Creative Technologies

Program Chairs
G. Michael Youngblood
youngbld@uncc.edu
University of North Carolina at Charlotte

Philip M. McCarthy
philmccarthy1@gmail.com
University of Memphis

Special Tracks Coordinator
Chutima Boonthum-Denecke
chutima.boonthum@gmail.com
Hampton University

Important Dates

Paper Submission:	Nov. 21, 2011
Author Notification:	Jan. 20, 2012
Camera-Ready Copy:	Feb. 20, 2012

Romero-Mariona, J.; Ziv, H.; and Richardson, D. J. 2008. SRRS: A Recommendation System for Security Requirements. Paper presented at the International Workshop on Recommendation Systems for Software Engineering, Atlanta, GA, 10 November.

Sandvig, J.; Mobasher, B.; and Burke, R. 2007. Robustness of Collaborative Recommendation Based on Association Rule Mining. In *Proceedings of the 2007 ACM Conference on Recommender Systems*. New York: Association for Computing Machinery.

Schafer, J. B.; Frankowski, D.; Herlocker, J. L.; and Sen, S. 2007. Collaborative Filtering Recommender Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science 4321. Berlin: Springer-Verlag.

Bamshad Mobasher is a professor of computer science and the director of the Center for Web Intelligence at the School of Computing of DePaul University in Chicago. He has published extensively in the areas of web data mining, web personalization, recommender systems, and predictive user modeling. He has also served in leadership positions for numerous related conferences and workshops, including as general chair, program chair and steering committee member of the ACM International Conference on Recommender Systems. Mobasher has served as an associate editor for *ACM Transactions on the Web* and on the editorial boards of several other prominent computing journals, including *User Modeling and User-Adapted Interaction* and the *Journal of Web Semantics*.

Jane Cleland-Huang is an associate professor of software engineering and director of the Systems and Requirements Engineering Center at the School of Computing of DePaul University in Chicago. She also currently serves as North American director of the Center of Excellence for Software Traceability (coest.org). She publishes extensively in the areas of requirements engineering and software traceability. She is currently associate editor for *IEEE Transactions on Software Engineering, IEEE Software* magazine, and the *Requirements Engineering Journal,* and has also served in various organizational and program committee roles for conferences such as the International Conference on Software Engineering (ICSE) and the International Requirements Engineering Conference (RE).