Evaluating a Recommendation System for User Stories in Mobile Enterprise Application Development

Matthias Jurisch, Maria Lusky, Bodo Igler, Stephan Böhm

Faculty of Design – Computer Science – Media
RheinMain University of Applied Sciences
Wiesbaden, Germany
Email: {matthias.jurisch,maria.lusky,bodo.igler,stephan.boehm}@hs-rm.de

***Abstract*—Mobile application development is characterized by a higher market volatility and shorter development cycles than traditional desktop application development. Developing mobile applications in large enterprise contexts (mobile enterprise applications) requires additional effort to adapt to new circumstances, since complex processes, user roles and enterprise-specific guidelines need to be taken into account. This effort can be reduced by reusing artifacts from other projects, such as source code, wireframes, documentation, screen designs or requirement specifications. We propose a recommendation system based on user stories in order to make artifacts accessible without requiring users to formulate an explicit search query. We present a prototype implementing this approach using standard methods and tools from information retrieval and evaluate it using different components of user stories as well as taking into account varying user story quality. The results show that using only user story text for calculating recommendations is the most promising approach and that user story quality does not affect the efficiency of recommendations.**

Keywords–Mobile Enterprise Applications; User Stories; Recommendation Systems; Pattern Inventories.

I. Introduction

The market for mobile applications (mobile apps) is characterized by the high volatility of platforms, devices and requirements. Hence, mobile app development projects require a shorter development cycle than traditional desktop applications. Consumer application development has been adapted to these circumstances by using agile methods and prototyping to accelerate app development. In the context of *mobile enterprise applications* (MEA), these adaptions require additional effort. Enterprise-specific guidelines, business processes and complex user roles need to be taken into account, which slows down the development process.

We proposed to approach this problem by building a repository with artifacts from past MEA-projects in the same enterprise and using this repository to accelerate and simplify the development process [1]. Project artifacts are screen designs, source code, requirements and technical documentation as well as all other documents created during the development process. Being able to reuse parts of these artifacts, using them for inspiration or for getting familiar with similar projects could help speeding up development.

In order to access the artifact repository efficiently, a method for user-friendly navigation of artifacts is required. Short, user-centered descriptions of usage scenarios called *User Stories* are common in requirements documentation in mobile app development. Since all other artifacts are in some way related to a user story, we consider user stories to be a reasonable starting point for navigating an artifact repository. Showing artifacts related to similar user stories to a user of the artifact repository should provide her with possible solutions to her problem. The solutions derive from best practices that had been used in previous projects that are similar to the one at hand. This can be realized through a recommendation system for project artifacts.

In this paper, we present a first step towards this recommendation system. When relating user stories to artifacts of similar user stories, the similarity computation between user stories is an important task. We present an approach for a recommendation system for user stories using standard information retrieval techniques and a prototypical implementation. We also evaluate in how far this approach fits the recommendation of user stories, which parts of user stories are relevant to the recommendation computation and how user story quality influences the accuracy of recommendations.

The remainder of this work is structured as follows: Section II describes the context of our research. Related work and its relation to our results are discussed in Section III. Section IV presents the architecture of a user story recommendation prototype. We discuss the evaluation methodology, the corpus used and two experiments to assess our prototype and approach in Section V. Section VI presents the results of the experiments. Implications from the results and potential shortcomings of our experiments are presented in Section VII. A conclusion and an outlook to future work is given in Section VIII.

II. Background

According to Flora et al. [2] mobile apps are "... compact programs developed to work on smartphones, tablets, and feature phones." Thus, an important characteristic of this type of software is that it has to be adapted to the specific requirements of mobile devices, mobile networks and mobile usage contexts. Besides this more technical perspective, the term mobile app has a specific meaning from the user perspective as well. It represents a bit of software that can be obtained from a distribution platform, i.e., an app store, and installed at the device by the user herself. Based on these characteristics, a more comprehensive definition of mobile apps can be given: Mobile apps are application software to run on mobile and network-connected devices, such as smartphones, to solve user-specific problems. They are provided by distribution platforms and consist of programs and data installed by the end user as an important element of handset personalization.

The origins of mobile apps are to be found in the consumer domain and closely related with the introduction of Apple's App Store in 2008. Since then, especially smaller enterprises have entered this emerging market and tried to exploit the business opportunities provided by the new app ecosystem. Even today, the app developer market is dominated by small companies. A global app developer report by Inmobi [3] from 2016 revealed that only eight percent of the participating firms had more than 20 employees. This suggests that most of the app development is carried out in small groups enabling a very direct communication amongst involved employees and characterized by short communication channels. As a result, app development tools and processes are often aligned to the requirements of lean or startup-like companies with agile and flexible structures, but they are less tailored for collaborative work environments within the complex structures and processes of large enterprises with a very high division of work and responsibilities.

However, large enterprises will not be able to evade the growing external demand for mobile apps. They need to adapt and adopt development tools and processes from the consumer segment to their own requirements and needs. For example, large enterprises need to find ways to prevent duplicate work across the entire enterprise and stimulate collaboration and knowledge sharing across teams when shifting to a more decentralized software development approach with agile and independent teams.

Besides the external challenges mentioned above, mobile apps are becoming more and more important for the corporate environment from the internal enterprise perspective. This trend is often discussed in the context of a *consumerization of IT*. As a result, Andriole [4] observes "... a reverse technology-adoption life cycle at work: employees bring experience with consumer technologies to the workplace and pressure their companies to adopt new technologies (with which many corporate technology managers might be only barely familiar)". This also has an impact on the processes and technologies used in enterprises by generating a shift towards the adoption of more consumer-driven approaches [5].

To sum up, large enterprises need to develop frameworks of processes and tools adapted to both, providing the flexibility of lean and consumer-driven approaches (coming from the consumer segment and smaller firms) by taking into account the high complexity of organizational structures within the context of large corporations. With this regard, the contribution of this paper is the evaluation of a user story based recommendation system as an element of a prototyping framework for MEA design in large enterprises. Our approach is based on the analysis of user stories linked to artifacts of MEA projects in a enterprise-wide repository. An identification of similar user stories could not only lead to reusable project artifacts but also foster cross-project communication and cooperation (e.g., by identifying the User Experience designer or developer of existing artifacts derived from completed projects) or help reducing uncertainty by providing reference points or estimates based on the learnings of completed projects (e.g., cost estimates for screen designs or software components). Before we provide an overview on the related work in this field, we will close this section with a brief discussion and conceptualization of the core elements of our approach.

A. Mobile Enterprise Applications

An exact definition of the term mobile enterprise app (MEA) is still missing [6] and it is used here in a wider sense. We are using the term to refer to any mobile app developed or deployed in the context of (large) enterprises and thus not only to mobile front-ends for existing enterprise software applications (EAS).

Mobile enterprise applications are often categorized by target groups into business-to-customer (B2C), business-to-business (B2B), and business-to-employee (B2E) [7], [6]. All the three types of MEA are in the scope of our study, as the challenges described before do not depend too much on the intended target group or user, but more on the organizational characteristics of the enterprise managing the app development process (by internal organizational units or subcontracting external suppliers).

B. Artifact and Repositories

The idea of an artifact repository is inspired by some longer-established concepts of (1) the usage of patterns in software engineering and human-computer interface design [8], (2) the asset reuse as promoted by the product-line approach[9], and (3) the design science research approach to evaluate artifacts for relevance and rigor in a systematical and iterative way [10]. As mentioned before, artifacts can comprise technical as well as non-technical aspects. The artifacts itself can be linked to organizational information or other details relevant for the development process (e.g., responsible developer or development costs).

A repository of all project-relevant information provides the underlying data for further analysis. The repository can be a common knowledge base within a project management software used for MEA development (e.g., Jira). In this respect, one of our main research objectives is to provide an approach to identify reusable project artifacts within this knowledge base to facilitate mobile app prototyping and development in large enterprises.

C. Recommendations

Based on a definition proposed by the organizers of the 2009 ACM International Conference on Recommender Systems (as cited in Robillard et al. [11]) recommendation systems for software engineering (RSSEs) are "... software tools that can assist developers with a wide range of activities, from reusing code to writing effective bug reports." According to this conceptuality, our aim is to evaluate an approach to identify similar project artifacts based on a sample repository. These similarities can then be used to provide a recommendation to UX designers or developers to consider aspects of existing artifacts for reuse or facilitate knowledge transfer across development teams.

However, one problem not mentioned before is the high complexity of MEA implementations, e.g., due to interfaces to back-end systems within the enterprise IT infrastructure. Some artifacts (e.g., login screens) might be characterized by a high level of similarity, but an incompatible implementation. This is why – as a first step – we decided to abstract from more implementation-oriented artifacts and defined an approach based on user stories.

D. User Stories

In the context of software engineering, requirements specify necessary functions and features of software. In traditional

software engineering, requirements are recorded in text documents that are difficult to grasp completely. In agile software development, user stories are user centered requirements expressed in one sentence, as described by Cohn [12].

A user story typically consists of three components: (1) a *role*, (2) a *desire* and (3) a *benefit*. While the role expresses, which kind of person wants the requirement, the desire and benefit describe the feature itself, for example: *As a user I want to mark and select favorites in order to receive information about my daily bus and train connections as fast as possible.* Each user story is associated with *acceptance criteria* that specify required properties for the implementation of the requirement described by the user story, for example: *The favorites should be ordered alphabetically.* Furthermore, Wake [13] defined the INVEST criteria for good user stories. According to his model, a user story should be *independent* from other user stories, *negotiable*, *valuable* with a benefit for the user that is clearly identifiable, *estimable* regarding its cost, *small* and *testable* or verifiable. These guidelines enable developers to easily write user stories that are meaningful on the one hand and comparable on the other hand. Creating user stories complying these criteria is a common practise in agile development.

III. Related Work

Regarding the general question of how enterprises should develop MEAs, according to [14] and [15], many enterprises still lack experience concerning their development. While there are no established process models for MEA development, first research approaches can be found in related literature. Dugerdil [16] presents an approach for transforming enterprise applications to mobile applications. An instrumentation framework that tries to ease the maintenance of MEAs is proposed in [17]. The management perspective of this problem is also represented in literature. Badami [18] examines this aspect from an organizational viewpoint and proposes the concentration of MEA development into "Mobile Centers of Excellence" that focus on competences of mobile experts inside enterprises.

Mobile app prototyping is supported by various tools. Existing prototyping tools (Kony, Verivo, Akula, SAP Mobile Platform) allow rapid prototyping. However, they can not always be used in the context of MEA, since they are focused on predefined use cases or the integration of existing enterprise products.

No processes or tools that specifically support the development of MEAs can be found in literature or in practice. Key questions that need to be answered are how an approach can take into account the specifics of MEAs and how time and effort for development can be decreased. Our approach is to reuse artifacts from existing MEA projects and to utilize recommendation systems in order to relate artifacts in pattern repositories and ease artifact reuse.

Leveraging information from user stories to recommend related artifacts has not been addressed in the literature until 2016 [19]. User stories are often stored in issue management systems that are normally used for bug tracking. Issues as used in issue management share some similarities with user stories. Both usually contain a short description formulated from a user perspective and are related to artifacts that are created to implement the changes required by the issue respectively the user story. A significant difference is that bug descriptions often contain a more technical language and provide less information about why a change is important and what the reason is for a change from an application domain perspective. While user stories have not been in the focus of scientific work regarding artifact recommendation, bug reports from issue management systems have been studied to support issue triage and issue-based project navigation.

In issue triage, systems find duplicates for bug reports. In this way, systems can automatically mark a bug report as a duplicate, minimizing the effort required to manually managing bug reporting systems. An approach by Runeson et al. [20] proposes using information retrieval techniques to detect duplicate bug reports. This approach has been combined with considering other artifacts like execution information [21]. Anvik and Murphy [22] have presented a framework for supporting developers building project-specific recommendation systems that help with assigning bugs to developers and linking them to project components. The framework allows the combination of several techniques for the construction of recommendation systems. Approaches from issue triage are primarily focused on bug reports and removing duplicates. Recommending useful solutions to similar issues, like in our case, is not considered.

Issue-based project navigation uses recommendation systems to support the navigation of projects. Hipikat [23] is a system that gathers information from mailing lists, documents and bug reports to ease the navigation of a source code repository. No quantitative evaluation of this approach exists and the applicability to other domains than issue management is not clear. Nevertheless, an evaluation of our similar approach to user-story-based recommendation systems seems promising.

Work in this direction has been conducted by Pirzadeh et al. [19]. Their approach recommends source code files based on user-story-similarity using standard information retrieval technologies. Connections between issues and source code are discovered based on tagged issue ids in commit messages of a version control system. While an evaluation shows a good performance of recommending source code artifacts in terms of precision, accuracy and specificity, it is not evaluated whether these results are actually caused by user story similarity. Relevant recommendations could also be a consequence of the general importance of some artifacts. Further aspects that are not investigated are which parts of the user stories should actually be used for the recommendation computation and how user story quality affects the recommendations.

In practice, plugins for the issue management system Atlassian JIRA that use information retrieval techniques to find similar issues exist [24]. These Plugins focus on duplicate detection and are not tailored to user story similarity. Also, they do not provide an API to programmatically access recommended similar issues and lack an evaluation. Especially no evaluation of the performance regarding the similarity computation and recommendation for user stories exists.

After reviewing the presented literature, we can conclude that several research questions have not been addressed in the literature: It is unclear to which degree textual user story similarity can be leveraged for getting useful recommendations. It has also not been studied, which parts of user stories are relevant to computing similarities and how user story quality affects the performance of the recommendation systems.

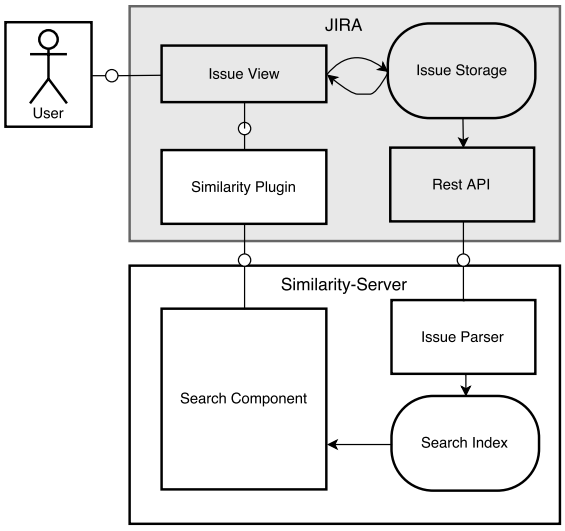

Figure 1. Prototype Architecture.

IV. PROTOTYPE

To support the construction of an artifact repository, a basis for a recommendation system has been implemented. The recommendation system will later interlink existing artifacts from several projects and different domains (e.g., user stories, source code, requirements, technical and organizational documentation) to help those involved in the software project. The foundation for our system that supports the interlinking of artifacts in general is a recommendation system for user stories. Given a user story as an input, the first iteration of the system will output the most similar and therefore relevant user stories. Later, several kinds of artifacts will be regarded.

To compute the similarity between user stories, we use standard procedures from information retrieval [25], in particular the vector space model (VSM), a representation of text documents, and term frequency-inverse document frequency (TF-IDF), a weighting method for terms in search queries and documents. First, all user stories are preprocessed. The documents are tokenized into collections of terms. All stopwords (e.g., "the", "it") are removed. Of the remaining terms, only a stemmed form is stored. User stories are represented in the vector space model (VSM), a common feature representation for natural language documents. In the VSM, each vector component refers to the term frequency of a term in the document.

To find similar user stories to a story, the text of the story is used as a query. The terms are weighted using TF-IDF term-weighting, which is calculated by multiplying the term frequency with the inverse document frequency of a term. The inverse document frequency is $\log \frac{N}{|t|}$, where N is the number of all documents (here user stories) and $|t|$ is the number of occurrences for a term in all documents. The query vector and all user story vectors are compared using the cosine similarity (i.e., the cosine of the angle between two vectors is used as the similarity measure). This process is implemented by Apache Lucene that uses several optimization strategies to improve the search performance. More details regarding these optimizations can be found in the Apache Lucene documentation [26].

User stories are stored as issues in Atlassian Jira [27], an issue management system. To allow easy developer access, our user story recommendation platform is integrated into Jira using the Jira plugin API. An FMC-Diagram of the implementation is given in Figure 1. Gray components are provided by Jira and hence did not need to be implemented. The *User* has access to an *Issue View* that is used to display and edit issues. Issues are persisted using an *Issue Storage* that can be accessed via a Rest API [28]. A *Similarity Server* is used to compute recommendations. The server is separated from Jira to improve the independences between recommendation generation and representation of user stories. In this way, other issue management systems as well as other similarity computation approaches can be used without too much adaption required. As a part of the server, the *Issue Parser* parses the user stories from the *Rest API* to a *Search Index*. A *Search Component* uses the index to answer requests from a *Similarity Plugin* that asks for recommendations for user stories currently displayed by the Jira *Issue View*. This architecture has been implemented as a prototype.

V. EVALUATION

In order to evaluate the performance of the prototype and the feasibility of the overall approach we conducted two experiments that focused on different aspects of user story recommendation.

- Does the usage of different components of user stories affect the usefulness of the recommendations?
- Does the quality of user stories affect the usefulness of the recommendations?

Therefore, the first experiment investigated the quality of recommendations with respect to different components of a user story that were used as a query on the one hand and as part of the corpus on the other hand. The second experiment took into account different levels of user story quality.

A. Corpus

For conducting both experiments, we built a corpus that consisted of 84 user stories for generating recommendations and 60 additional user stories to account for noise effects. We needed different user stories with acceptance criteria, whereas a part of them should describe the same use case. Thus, we created two different use cases *A - favorites* and *B - location* for a popular public transport app and made two short video films of about 20 seconds each that showed typical interactions of each use case. The two video films were then shown to a group of German-speaking students. Each student created a user story and three acceptance criteria per use case, resulting in 42 user stories for each use case A and B. Since our user stories described only two different use cases, we added 60 more user stories from an external data set [29] that were translated into German via a semiautomatic procedure.

If all user stories from the corpus were used as a query and as a part of the set of documents that was searched, the first recommendation would always be the user story used as the query, which would skew the results. To address this issue, the user stories were randomly separated into a query and a search corpus set. Only user stories for use cases A and B were allowed as a part of the query set, since no relevant recommendations could be generated for user stories from the external data set. The query set contained 30 user stories, so it comprised about 25% of the overall document corpus.

Recommendations were generated only for user stories in the query set. Only elements of the search corpus set were used as recommendations.

B. Metric

For each user story, recommendations are calculated based on textual similarity. A recommendation is regarded as "correct", when the recommended user story relates to the same use case A or B as the query. In the context of this work we consider a recommendation system the more *useful*, the higher the precision of the result is. *Precision* and *recall* are standard metrics for evaluating these kinds of scenarios [30]. Since a recommendation system will provide the user with only the first few recommendations, the usage of overall precision and recall is not an appropriate metric in this case. A metric that is more useful regarding recommendation systems in our case is *precision at rank* [25, p.161]:

$$PR = \frac{TP}{SR}$$

TP is the number of true positives (i.e., recommendations of the correct use case) and SR a fixed number of search results. This metric only evaluates the SR best search results. The two experiments were conducted for each rank from one to ten. To get an overview of the overall performance, the average of the precision for each user story was calculated. We therefore measured the average precision at rank for ranks one to ten.

C. Experiment 1: User Story Components

For the first experiment we defined several variants of our data sets: user stories along with their associated acceptance criteria (USAK), user stories only (US) and acceptance criteria only (AK). These are the smallest possible variants that are sensible. Separating the user stories into smaller components would result in parts of very few words, which would not allow meaningful processing using IR techniques. Each variant can be used as a query set on the one hand and as a corpus set on the other hand. We combined each variant as a query with each variant as a corpus and therefore conducted the first experiment nine times with all nine combinations of our data. Since no acceptance criteria were available for the external data set, the same artificial acceptance criteria were added to each of these user stories. Hence, it is less likely to retrieve user stories as recommendations from the external data set while the corpus set consists of user stories with acceptance criteria (USAK) or only acceptance criteria (AK). This would lead to a better average precision at rank for these cases. Effects of this shortcoming will be further discussed in Section VII.

D. Experiment 2: User Story Quality

A second experiment was conducted to evaluate if user story quality affects the effectiveness of a text-similarity based recommendation system. For this purpose we categorized the user stories into three quality groups.

First, the quality of all user stories was rated based on a five point scale. We defined five quality criteria and assigned one point to the user story for each criterion that was satisfied: (1) The user story had to have exactly one *role*. (2) The user story had to express exactly one *desire*. (3) The user story had to have exactly one *benefit*. (4) The user story had to be written in only one sentence. (5) The user story's *benefit* had to be verifiable, as defined by the INVEST-criteria.

Then the user stories (written by students and from the external data source) were categorized into three quality classes: User stories with five points were assigned to quality class 1, user stories with four points were assigned to quality class 2 and user stories with three or less points were assigned to quality class 3, resulting in class 1 comprising 95 user stories, while classes 2 and 3 contained 26 respectively 23 user stories.

To evaluate the effect of user story quality on recommendations, datasets with different user story quality were needed. Using the previously described categories, the data was split into three sets: (1) user stories of all quality classes, (2) user stories of quality classes 1 and 2, (3) user stories of only the high quality class 1. For each of these sets, the precision at all ranks from one to ten was calculated. For this experiment, only the combination of user story parts from experiment 1 with the best results regarding precision was used.

VI. Results

We implemented a small evaluation application that initiates the recommendation-process, gathers user story recommendations, automatically calculates the average precision at rank by evaluating class labels (e.g., corresponding use case) according to the approach presented in Section V and stores results in a CSV file. The results of the experiments performed using this application are discussed in the following subsections.

A. Experiment 1: User Story Components

The results from the first experiment are shown in Figure 2. Each data series represents one combination of corpus and query sets, whereas the first part of the data series label denotes the query set and the second part denotes the corpus type (e.g., *AK/US* represents queries that contain only acceptance criteria and a corpus consisting of user stories without acceptance criteria).

The data series can be split into two categories: (1) Experiments where user story text is used as query *and* corpus and (2) experiments where either the query or the corpus consists *only* of acceptance criteria. With one exception (*USAK/USAK* at rank 1), members of the first category in general have a higher average precision than members of the second. For readability purposes, only representatives of these categories with highest and lowest average precision at rank in the respective group are shown in the data series. Note that *AK/US* and *AK/AK* are representatives with lowest average precision for their group at different ranks. Therefore two data series are displayed as lower bounds for the second category. The data series that are not displayed (*USAK/US* and *US/USAK* in the first group; *AK/USAK* and *US/AK* in the second group) are always between the upper and lower bounds of their respective groups.

The combination using only user stories (*US/US*) shows the highest average precision. The precision for rank 1 is 1.0 and drops to 0.88 when considering the first 10 recommendations. The data series of the combined user stories along with acceptance criteria (*USAK/USAK*) has a lower precision that ranges between 0.71 and 0.79. The data shows that the precision differs widely between these combinations.

In the category of the experiments where either the query or the corpus consists of acceptance criteria only, the combination *USAK/AK* showed the highest precision at rank that ranged from 0.54 to 0.75. The two combinations of *AK/US* and *AK/AK* resulted in the lowest precision varying between 0.58 and 0.63 for *AK/US* and between 0.53 and 0.65 for *AK/AK*. However, the

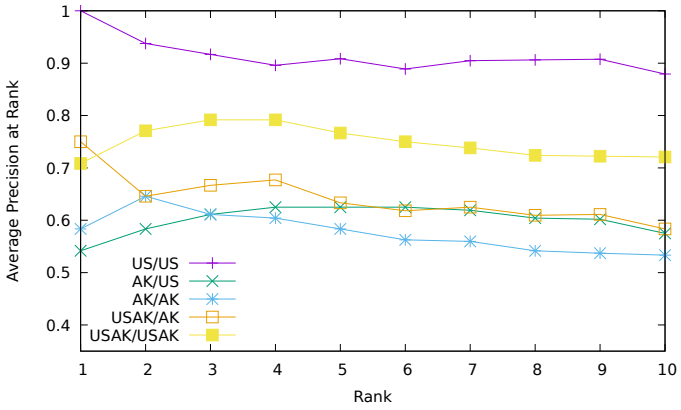

Figure 2. Evaluation results for combinations of query and corpus data.

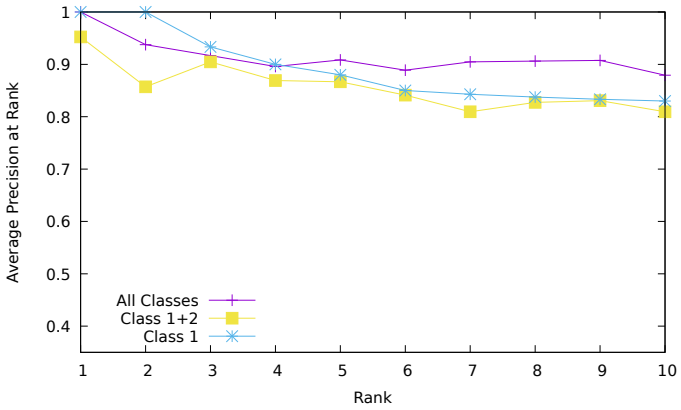

Figure 3. Comparison of results when filtering user story quality.

results show only small differences between all combinations in this group.

In general, data series that use only user stories either in the query or the corpus set have a higher average precision than the data series applying acceptance criteria only in either query or corpus. The three data series with the lowest average precision all use only acceptance criteria as queries. Average precision of these data series is between 0.53 and 0.65 varying in dependence of the rank, while the average precision of the data series of the other category varies between 0.71 and 1.0.

B. Experiment 2: User Story Quality

The results of the second experiment are shown in Figure 3. The data series *Class 1* contains only high quality user stories of class 1, as described in Section V-D. *Class 1 + 2* are medium- and high-quality user stories, while *All Classes* refers to user stories of all quality classes.

Average precision values for all data sets are between 0.8 and 1.0. While class 1 user stories show the best results at ranks 1, 2 and 3, the precision lowers at the following ranks. The precision with data of all quality classes varies between 1.00 and 0.88 and receives the highest values at ranks 5 and following. However, the precision values of quality classes 1 and 2 are the lowest at all ranks.

VII. Discussion

The results we received from our experiments provided us with information about the precision of text-based recommendations for user stories and acceptance criteria. Based on these results we aim to answer our two research questions:

- Does the usage of different components of user stories affect the usefulness of recommendations?
- Does the quality of user stories affect the usefulness of recommendations?

A. Experiment 1

The results from experiment 1 showed that the highest precision is received using the combination US/US. The precision of the other combinations is not only lower, but by far lower. Also, after a decrease from rank 1 to rank 2, the precision of the US/US combination remains relatively stable. Out of all remaining combinations, the ones that use user stories in both the query and the corpus produced the higher precision values. All combinations that use only acceptance criteria in the query and/or the corpus show the lowest precision that is by far lower than the highest received precision.

Based on our results we can therefore answer our first research question positively. Different components of user stories do affect the usefulness of recommendations. We discovered that the usage of acceptance criteria deteriorates the precision of the recommendation system, since these combinations received the lowest precision values. Corresponding to that, the combination that contained only user stories and no acceptance criteria led to the highest precision values. While it is difficult to define limits for the precision of a recommendation system, we believe the observed precision is sufficient for recommendation systems in the context of mobile enterprise application development. We therefore conclude that using only user story text as a basis for recommendation computation in this context seems to be the most promising alternative.

B. Experiment 2

The results from experiment 2 show that the usage of all quality classes leads to a higher precision from rank 4 onward. However, at ranks 1, 2 and 3 the usage of only high quality user stories received the best precision values. Furthermore, it is especially notable that average precision is lower for user stories of quality classes 1 and 2 combined than for all kinds of user stories.

Based on this data, we can not ascertain any impact of user story quality on recommendation precision and therefore, our second research question is to be answered negatively.

C. Thread to validity

One weakness of our experimental setup may be the small number of user stories in our corpus. However, the distribution of user stories to primarily two use cases compensates this shortcoming. Although the quality rating of the user stories did not follow a common model for user story quality, it is based on the main and most popular user story models. Furthermore, to our knowledge there is no established model for quality measurement of user stories on an individual level.

As mentioned in Section V-A, we used data from an external source that did not contain acceptance criteria. One could assume that this would distort the results in favor of the combinations that use acceptance criteria. However, these combinations gained lower precision than the other combinations, despite the lack of acceptance criteria. Therefore, our conclusion that the usage of user stories has a positive effect on precision still holds.

VIII. Conclusion

In this paper we have presented foundation work for an artifact repository to support the development of MEA. The foundation is built on a recommendation system for artifacts based on user-centered requirement specifications called "User Stories", in order to allow the user of the artifact repository an efficient navigation. Standard information retrieval techniques were used to build the recommendation system. We evaluated which parts of the user stories should be used for recommendation computation and how user story quality affects the performance of the system.

Our experiments have shown that the most valuable part for requirement computation from a user story is the user story text. Including acceptance criteria into recommendation computation has a negative impact on recommendation performance. In our experiment data, we could not find a positive correlation between user story quality and quality of recommendations.

As future work, we plan to use our results to extend our recommendation system that includes development artifacts from several domains. Collecting and evaluating methods for connecting different artifact types to user stories will be the next step to reach this goal. In addition, the artifacts will be enriched with further information, such as cost and benefit information or process and workflow data.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research, grant no. 03FH032PX5; the PROFRAME project at RheinMain University of Applied Sciences. All responsibility for the content of this paper lies with the authors.

References

[1] M. Jurisch, B. Igler, and S. Böhm, "PROFRAME: A Prototyping Framework for Mobile Enterprise Applications," in CENTRIC 2016, The Ninth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2016, pp. 7–10, 2016.

[2] H. K. Flora, X. Wang, and S. V. Chande, "An investigation on the characteristics of mobile applications: A survey study," International Journal of Information Technology and Computer Science, vol. 6, no. 11, pp. 21–27, 2014. [Online]. Available: https://doi.org/10.5815%2Fijitcs.2014.11.03

[3] Inmobi, "State of mobile app developers 2016: Based on a survey of 1000+ app developers," http://www.inmobi.com/insights/download/whitepapers/state-of-mobile-app-developers-2016/, accessed: 2017-03-02.

[4] S. J. Andriole, "Managing technology in a 2.0 world," IT Professional, vol. 14, no. 1, pp. 50–57, 2012. [Online]. Available: https://doi.org/10.1109%2Fmitp.2012.13

[5] B. Niehaves, S. Köffer, and K. Ortbach, "The effect of private it use on work performance: Towards an it consumerization theory," in 2013 11th International Conference on Wirtschaftsinformatik. Institute of Electrical and Electronics Engineers (IEEE), 2013, pp. 39–53, 2013. [Online]. Available: http://aisel.aisnet.org/wi2013/3/

[6] A. Giessmann, K. Stanoevska-Slabeva, and B. de Visser, "Mobile enterprise applications–current state and future directions," in 2012 45th Hawaii International Conference on System Sciences. Institute of Electrical and Electronics Engineers (IEEE), 2012, 2012. [Online]. Available: https://doi.org/10.1109%2Fhicss.2012.435

[7] R. C. Basole, "The emergence of the mobile enterprise: A value-driven perspective," in International Conference on the Management of Mobile Business (ICMB 2007). Institute of Electrical and Electronics Engineers (IEEE), 2007, 2007. [Online]. Available: https://doi.org/10.1109%2Ficmb.2007.63

[8] J. O. Borchers, "A pattern approach to interaction design," AI & Society, vol. 15, no. 4, pp. 359–376, 2001. [Online]. Available: https://doi.org/10.1007%2Fbf01206115

[9] K. Kang, J. Lee, and P. Donohoe, "Feature-oriented product line engineering," IEEE Software, vol. 19, no. 4, pp. 58–65, 2002. [Online]. Available: https://doi.org/10.1109%2Fms.2002.1020288

[10] A. Cleven, P. Gubler, and K. M. Hüner, "Design alternatives for the evaluation of design science research artifacts," in Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09. Association for Computing Machinery (ACM), 2009, 2009. [Online]. Available: https://doi.org/10.1145%2F1555619.1555645

[11] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," IEEE Software, vol. 27, no. 4, pp. 80–86, 2010. [Online]. Available: https://doi.org/10.1109%2Fms.2009.161

[12] M. Cohn, User stories applied: For agile software development. Addison-Wesley Professional, 2004.

[13] B. Wake, "INVEST in good stories, and SMART tasks," blog post, [retrieved: 2017.05.10], 2003. [Online]. Available: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks

[14] A. Giessmann, K. Stanoevska-Slabeva, and B. de Visser, "Mobile enterprise applications–current state and future directions," in System Science (HICSS), 2012 45th Hawaii International Conference on, Jan 2012, pp. 1363–1372, Jan 2012.

[15] Gartner, "Gartner says demand for enterprise mobile apps will outstrip available development capacity five to one," website [retrieved: 2017.05.10], 2015. [Online]. Available: https://www.gartner.com/newsroom/id/3076817

[16] P. Dugerdil, "Architecting mobile enterprise app: A modeling approach to adapt enterprise applications to the mobile," in Proceedings of the 2013 ACM Workshop on Mobile Development Lifecycle, ser. MobileDeLi '13. New York, NY, USA: ACM, 2013, pp. 9–14, 2013.

[17] M. Pistoia and O. Tripp, "Integrating security, analytics and application management into the mobile development lifecycle," in Proceedings of the 2Nd International Workshop on Mobile Development Lifecycle, ser. MobileDeLi '14. New York, NY, USA: ACM, 2014, pp. 17–18, 2014.

[18] S. A. Badami and J. Sathyan, "micE Model for Defining Enterprise Mobile Strategy," Int. J. on Recent Trends in Engineering and Technology,, vol. 10, no. 1, p. 9, Jan 2014.

[19] H. Pirzadeh, A. D. S. Oliveira, and S. Shanian, "ReUse : A Recommendation System for Implementing User Stories," in International Conference on Software Engineering Advances, no. c, 2016, pp. 149–153, 2016.

[20] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," Proceedings - International Conference on Software Engineering, pp. 499–508, 2007.

[21] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information 1 2," Proceedings of the 30th international conference on Software engineering, pp. 461–470, 2008.

[22] J. Anvik and G. C. Murphy, "Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 20, no. 3, 2011.

[23] D. Cubranic and G. C. Murphy, "Hipikat: Recommending Pertinent Software Development Artifacts," 25th International Conference on Software Engineering, no. Section 2, pp. 408–418, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=776816.776866

[24] D. Oguz, "Similar issues finder," website, [retrieved: 2017.05.10], 2017. [Online]. Available: https://denizoguz.atlassian.net/projects/SIF/summary

[25] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. New York, NY, USA: Cambridge University Press, 2008.

[26] Apache Foundation, "TFIDF Similarity (Lucene 4.6.0 API)," website, [retrieved: 2017.05.10], 2017. [Online]. Available: https://lucene.apache.org/core/4_6_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

[27] Atlassian, "JIRA Software - Issue & Project Tracking for Software Teams," website, [retrieved: 2017.05.10], 2017. [Online]. Available: https://www.atlassian.com/software/jira

[28] Atlassian Developers, "JIRA REST APIS," website, [retrieved: 2017.05.10], 2017. [Online]. Available: https://developer.atlassian.com/jiradev/jira-apis/jira-rest-apis

[29] Open Knowledge International, "Frictionless data: User stories," website, [retrieved: 2017.05.10], 2016. [Online]. Available: http://frictionlessdata.io/user-stories/

[30] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," Tech. Rep. December, 2007.