Applying Machine Learning for Automatic User Story Categorization in Mobile Enterprises Application Development

https://doi.org/10.3991/ijim.v14i15.15263

Matthias Jurisch (✉), Stephan Böhm, Toby James-Schulz
RheinMain University of Applied Sciences, Wiesbaden, Germany
matthias.jurisch@hs-rm.de

Abstract—Mobile enterprise applications (apps) are developed in dynamic and complex environments. Hardware characteristics, operating systems and development tools are constantly changing. In larger institutions, comprehensive corporate guidelines and requirements have to be followed. In addition, larger enterprises often develop numerous apps and lack an overview of development projects. Because of the size of such companies, a comprehensive direct information exchange between developers is often not practicable. In this situation, IT support is necessary, for example to prevent unnecessary duplication of work in the development of software artifacts such as user stories, app screen designs or code features within the company. One approach to overcome these challenges is to support reusing results from previous projects by building systems to organize and analyse the knowledge base of enterprise app development projects. For such systems an automatic categorization of artifacts is required. In this work we propose using a machine learning approach to categorize user stories. The approach is evaluated on a set of user stories from real-world mobile enterprise application development projects. The results are promising and suggest that machine learning approaches can be beneficially applied to user story classification in large companies.

Keywords—Mobile enterprise applications, text classification, user stories.

1 Introduction

Mobile application development as a relatively new field of software development is prone to rapid changes. This poses a challenge to developers: new applications need to be built over the course of months, not years, even for relatively large projects. This is especially challenging in the field of mobile enterprise applications [10] or in the context of mobile work [22]. When developing applications for these areas, developers are often confronted with several new technologies at the same time, e.g., cloud and mobile technologies [5]. In the enterprise context, the requirement for fast development occurs concurrently with corporate demands such as high expectations regarding usability and productivity enhancements, compliance with legal requirements, corporate design specifications, enterprise IT architecture, security standards as well as

other enterprise-specific guidelines and requirements. Adhering to guidelines and saving time can be addressed by improving the reuse of results that already complied with enterprise requirements in previous development projects. Such reuse can be supported by building platforms that automatically archive and organize software artifacts from previous development projects. Development of mobile enterprise applications is usually implemented with some kind of agile approach. A common method used in agile approaches for documenting requirements is writing user stories [7]. A user story contains information on the actors, functions and objects relevant to a certain aspect of the system that is described. This information contained in the user story is also very important for building a system that can automatically make other artifacts from development projects, e.g., screen designs or code features, accessible and easy to find. How to leverage this information for building such a system is the question at the core of this work. Our contribution is an approach that uses state-of-the-art text classification methods based on machine learning, i.e., neural networks, for the classification of user stories from mobile enterprise application development. The goal of the approach is to automatically categorize user stories to help developers to find reusable artifacts they can use in their current work. To achieve this, we manually categorize a set of user stories from real-world mobile enterprise application development projects. We then evaluate how well a neural network is able to predict these categories. The best F1 score achieved for this classification is 0.9, which indicates that such a machine learning classification approach can be useful in practice.

The remainder of this work is structured as follows: Section 2 gives an overview of the background of mobile enterprise applications and user stories. Related work and the research gap this work fills are discussed in 3. Section 4 describes the classification approach. The dataset from real-world development projects we use as well as the classes that are predicted by the classification approach are presented in Section 5. An evaluation of our approach based on the dataset is presented in Section 6. The results of the evaluation are discussed in Section 7. Section 8 gives a conclusion and an outlook on future work.

2 Mobile Enterprise Application Development and User Stories

The term “mobile enterprise application” is not generally defined [10]. In this work, we use this term to describe software solutions for mobile devices that are created or used in the context of the daily work of (large) enterprises, ranging from applications that are used to coordinate processes internally as well as customer-facing applications (such as banking apps [2]). While the term ”mobile enterprise application” is often used to refer to mobile front-ends only, i.e., the software package installed on mobile devices, the definition applied in this work is more broad and refers to every component of a mobile application developed for the use in large enterprises, including its back-end infrastructures and interfaces to other corporate systems.

Today, most of the mobile applications, and this also applies to the enterprise context, are developed by using agile project management approaches [11]. The documentation of requirements in such agile development projects is based on user

stories. User stories are a common format to manage and document requirements from a user perspective. A user story should describe how a user can interact with a system to achieve a specific goal. A very common template for user stories is the Connextra-format [7]. This template is shown in the following Listing:

- As a < user >
- I want < feature >
- So that < reason >

The user -part describes the user. This description can contain what kind of user interacts with the system, e.g., the platform the user is using or the user's organizational role. The feature-part describes the interaction with the system. The reason is used to describe, why the user wants to interact with the system in the way described in the user story. This part is especially useful when it comes to deciding which stories to implement – if the reason is weak or hard to determine, then the story probably should not be implemented. To better illustrate the application of user stories in practice the following example from [13] shows the type of user story we consider in this work:

As a tablet user I want to mark and select favorites in order to receive information about my daily bus and train connections as fast as possible.

As was found in previous work [13] user stories often vary regarding their quality and granularity. In the dataset considered in our earlier works, the reason aspect of user stories is often left out or can be derived from the context of the story. The rationale for leaving out the reason seems to be based on two factors: either the reason is woven into the feature part of the user story, or the author of the story sees the reason based in common sense. An example for this are stories such as" as a user I want a cancel button" and" as a user I want to receive error notifications". Besides, user stories are not always formulated in a way that they are easily understandable for a person not working on that specific project. When trying to apply user stories to improve software reuse, this is an important challenge to keep in mind.

3 Related Work

Our work is related to the area of recommendation systems in software engineering, which is an active field of research [20]. However, most approaches focus on the source code but" unexploited opportunities lie in the development of recommender systems outside the source code domain" as stated by [9]. Hence, we focus on user stories. Several approaches for relating short textual data to support software engineering have been published. Hipikat [8] uses textual similarity measures from the area of information retrieval to compute similarities between bug reports. These similarities are then used to support the developer in fixing bugs. Similar ideas have been proposed in the area of issue triage. The goal of issue triage is to automatically assign bug reports to developers.

Information retrieval approaches have also been used in this domain [3, 21]. More recent ideas use deep learning methods to approach this problem [16]. Information retrieval techniques have also been applied to user stories: [19] proposes an approach

to recommend artifacts by relating user stories from the same project to each other using textual similarity measures. Previous work [14], evaluated how well information-retrieval-based approaches can distinguish between two types of user stories and which aspects of the user story are important to it. Moreover, we presented first evidence on how these approaches perform on real-world data [15]. It was found that when working with user stories from practice, quality and heterogeneity of the story descriptions are a challenge to this recommendation approach.

It was also found, that methods from information retrieval are particularly good at finding stories from the same project, which made finding related stories from different projects relatively hard. To overcome this issue, we proposed manually categorizing relevant parts of stories [13]. Manually determined categories have several advantages: they can capture synonyms, they can be tailored to specific needs and building the categories can lead to new insights in the domain. However, this also requires a significant amount of manual labour.

The drawback of requiring manual labour leads to our approach: Given a labeled subset of user stories, we want to predict the categories of the remaining stories in the dataset. These automatically assigned categories can then be used to better organize and structure stories and related artifacts in a software repository which in turn can improve retrievability and reuse of artifacts like screen designs or code. Thus, finding an approach for automatically categorizing user stories from mobile enterprise application development is the research gap we address in this work.

4 Approach

Our Approach to helping developers organizing user stories is based on two ideas:

1) Manually assigning categories to important components for a subset of user stories in the dataset. As a result, a set of categories exists for the user type, the function the story describes and the object this function interacts with. The annotation process is described in detail in [13]. This yields a set of user stories that have been assigned to three categories (user, function and object) each.
2) The set of annotated user stories is then used as training data for a text classification model. The trained model is deployed to assign categories to the remaining user stories, so they can be organized easily by mobile enterprise application developers, e.g., by integrating such a mechanism within a repository used for managing agile software development projects.

An overview of the text classification method is given in Figure 1. First, all user stories are tokenized and transformed into a set of n-grams. This yields the set of tuples of size n, where each tuples contains n words that appear in succession in the user story. On these sets of n-grams we apply the TF-IDF algorithm, a method for document representation based on term occurrence in documents that is very common in information retrieval [12]. In this paper we apply the algorithm to n-grams instead of terms. Each user story d is represented by a vector W_d, that contains an entry for each

n-gram used in the dataset. Each vector component$W_{d,t}$ represents the importance of an n-gram *t* for the user story *d*.

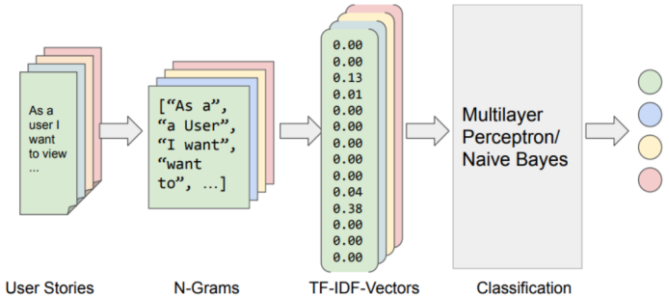


Fig. 1. Overview of Classification Approach

The importance of an n-gram is computed by multiplying $tf_{d,t}$, the frequency of n-gram t in document d, by a representation of how common the n-gram t is in all documents. For measuring the commonality of the n-gram t, the inverse document frequency $log\left(\frac{N}{df_t}\right)$ is used. Nrepresents the number of all documents and df_t is the number of occurrences for the n-gram t in all documents. This yields the following formula for a document's vector representation:

$$W_{d,t} = tf_{d,t} * log\left(\frac{N}{df_t}\right) \quad (1)$$

The resulting vectors are then used as feature vectors to train a text classification model. As text classification approach we used a feed-forward neural network, more specifically a multilayer perceptron. We have chosen this relatively simple model, as there is little training data available. To investigate, whether a neural network is actually required to perform a classification task in this domain, we also train a naive Bayes classifier on the same data.

5 Dataset

The dataset we use is the result of the work described in [13]. Stories are taken from a Jira [4] system that is used by more than 100 employees of a large enterprise in Germany that works on creating mobile enterprise applications. The Jira system is used to manage about 30 Projects. The user story data is extracted from several projects. The subset of stories for the manual classification approach was chosen randomly.

Categories are assigned as described in [13]. In total, 403 stories where manually labeled following this approach. Since categories were chosen from a standpoint of usefulness to developers, the assignments are not evenly distributed, but very heterogeneous. As mentioned before, categories are assigned to three parts of a story: User, Function, and Object. For categorization, the user story descriptions were manually examined and then clusters were formed based on similarity.

The first set of categories describe the user part of the story. An overview is given in Table 1. The categories describe different aspects of users, e.g., what platform they use or what role they embody in a company. The role aspect is especially important for enterprise applications, since in a working environment with high degrees of specialisation and division of labour there can be very diverse and well differentiated roles in contrast to consumer apps. Except for category 5, class categories are relatively equally distributed.

Table 1. User Categories of the Subset's User Stories

ID	Category Description	Frequency
0	Generic User	91
1	Platform-specific User	85
2	Organizational Role	119
3	Supervisor Role	43
4	External Users	56
5	Department	9

The second set of categories describes one specific aspect of the feature from the Connextra template: the type of function that is applied. These are referred to as *function* categories. The function categories categorized in the data subset are shown in Table 2. The distribution of classes are more uneven than for the user classes. An interesting thing we found is these kinds of categories contain elements of a classic design pattern from database design, namely the CRUD pattern (create, read, update, and delete) [17]. In other words, many of the user stories are requesting important elementary functions as well as other standard functions commonly found in software solutions. Especially the user stories containing frequently implemented features can be used to save time but also serve as a sort of checklist for future development projects.

The last category describes the *object*, that the feature refers to. An overview of categories for objects is given in Table 3. The most common categories are 1-4, for other categories only a few samples could be found in the subset of data analyzed. Of these categories "Documents and Reporting" is the most frequent. This is also related to the fact that these are mobile enterprise applications and that documentation and reporting are not usually features of consumer apps.

Table 2. Function Categories of the Subset's User Stories

ID	Category Description	Frequency
0	No Function	5
1	Creating Data	12
2	Updating Data	23
3	Deleting Date	21
4	Data Collection	118
5	Import/Export	32
6	Technical/System Functions	92
7	Conversational	28
8	Other	72

Table 3. Object Categories of the Subset's User Stories

ID	Category Description	Frequency
0	No Object	7
1	Widgets	74
2	Documents and Reporting	161
3	Technical Terms	51
4	Feedback	62
5	Error Handling	13
6	Views and Presentation	18
7	Notifications	17

The manual categorisation was carried out with the aim of finding clusters of user stories in the dataset that are as uniform as possible. This is because the resulting clusters are more likely to contain interchangeable software artifacts suitable for a reuse. For clustering, it is necessary to abstract from project specific terminology and find generic terms. This requires a certain amount of expertise, but it can be assumed that within a company, domain-specific terms also occur across projects and can thus be" "learned". Our approach therefore assumes that categorisation patterns can be identified or learned from the manually labelled user story components and used for automatic classification by machine learning methods.

6 Evaluation

To evaluate our approach, we applied the method described in Section 4 on the dataset described in Section 5. Our research questions are the following:

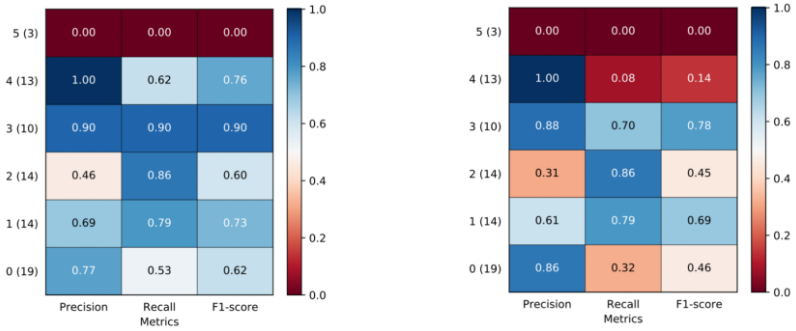
a) What is the benefit of using a Neural Network instead of a more simple classifier such as Naive Bayes?
b) Is it possible to predict categories of unseen user stories? What performance can be achieved regarding standard classification metrics such as precision, recall, F1 score and accuracy?
c) Which categories are easier/harder to predict? What are possible reasons for differences?

To answer these research questions, we separate the 403 stories in the dataset into a training part of 330 stories, the remainder is used as a test set. We train the classifiers on the training set and evaluate them on the test set. We do not tune hyperparameters but use a standard configuration. More specifically, the multilayer perceptron uses three hidden layers with 64 units. The training loss used is sparse categorical cross-entropy. The models are implemented using Keras [6], tensorflow [1], and sklearn [18].

6.1 Prediction of user category

In Figure 2, we show the detailed results for precision, recall, and F1 score for the classification of user categories regarding each category. The category id is shown on the left. The number in parenthesis indicates the number of samples for the

corresponding category in the test set. The color encodes the values according to the metric displayed at the bottom, where blue encodes the higher and red the lower values. Additionally, the values for each metric are included in the diagram. The overall classification accuracy is 0.70 for the multilayer perceptron-based model and 0.51 for Naive Bayes.



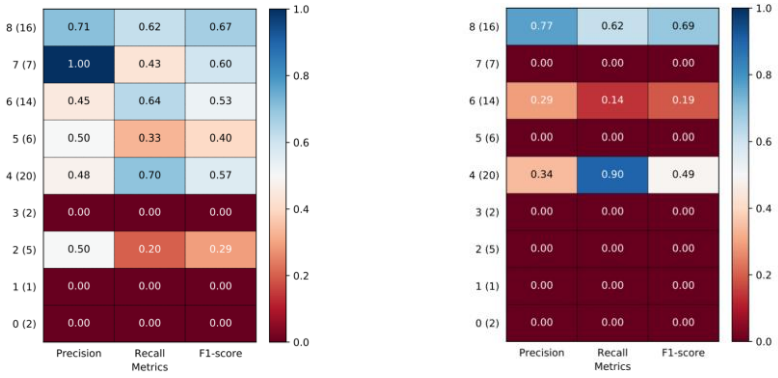
a) Multilayer Perceptron b) Naive Bayes

Fig. 2. Classification Results for User Class Prediction

Except for class 5 (Part of Company/Role), which contains only very few samples, the multilayer perceptron is able to predict most classes with F1 scores above 65 percent. This is not the case for the Naive Bayes classifier, that only achieves a similar performance for category 1 (Device/Platform) and 3 (External Role). Categories 1 and 3 should be relatively easy to predict, as samples in this class contain n-grams such as "mobile user", "android user" or "customer". Compared to the Naive Bayes classifier, the multilayer perceptron model seems to be more flexible at capturing more complex classes such as supervisor roles or functional roles.

6.2 Prediction of the function category

Figure 3 shows the classification results for the prediction of function categories. The figure uses the same format as described in Section 6.1. The overall classification accuracy is 0.47 for the multilayer perceptron model and 0.41 for the Naive Bayes model. For function class prediction, the performance of the multilayer perceptron model seems to be more similar to the performance using Naive Bayes. In the test set, only very few examples for the function categories 0 (No Function, 1 (Creating Data), 2 (Updating Data), and 3 (Deleting Data) exist. Since this is the same for the representation of these categories in the training set, these categories are hard to predict.

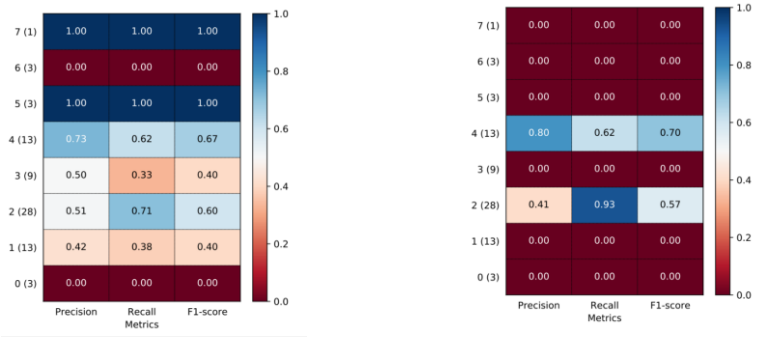


a) Multilayer Perceptron b) Naive Bayes

Fig. 3. Classification Results for Function Class Prediction

6.3 Prediction of the object category

Classification results for the prediction of the object category are shown in Figure 4. Overall classification accuracy is 0.56 for multilayer perceptron classification and 0.47 for Naive Bayes. While these numbers seem relatively close, from looking at the figure, the multilayer perceptron seems to achieve much better results than Naive bayes. Naive Bayes classification is only able to correctly predict examples from classes 2 (Documents and Reporting) and 4 (Feedback). In contrast, the multilayer perceptron approach not only achieves acceptable performance when classifying examples for classes 2 (Documents and Reporting) and 4 (Feedback) but outperforms Naive Bayes classification for categories 5 (Error Handling) and 7 (Notifications).



a)Multilayer Perceptron b) Naive Bayes

Fig. 4. Classification Results for Object Class Prediction

7 Discussion

From the results shown in Section 6, we can draw conclusions regarding the first research question on the viability of classification using a neural network: The highest precision for classes with at least ten samples is 1.0, the best recall 0.9, and the best F1 score is 0.9. The highest overall accuracy is 0.7. The lowest values for categories with at least ten samples are a precision of 0.45, a recall of 0.53 and a F1 score of 0.53. These values indicate that the presented classification approach seems to be viable for the problem regarded in this work.

With respect to the second research question, a comparison between Naive Bayes and multilayer perceptron classification shows that multilayer perceptron clearly outperforms the Naive Bayes classifier. While there are few classes, where Naive Bayes achieves better metrics, the differences in these cases are very small. The computational effort for training the classifiers was negligible for both models. Hence, using a neural network for this problem is clearly preferable.

Regarding classification performance of different user story categories, as addressed with the third research question, we find that the categories for the user component of a user story are the easiest to predict. Only one user category could not be predicted at all. This category has only a few examples in the training set as well as in the test set. It was to be expected, that the user categories are the easiest to predict, as the user part contains a few and very specific words only. Annotating user categories was easy as the user is typically described with a few words at a single position in the text of the use case. Thus, a few n-grams only are relevant for the categorization. In summary, the best data in this dataset of user stories was available for the user categories. In this context it is noticeable, that for the user categories, the difference between multilayer perceptron and Naive Bayes classification in accuracy was the highest. This shows that with better data, the multilayer perceptron classifier outperforms the Naive Bayes classifier even more significantly.

The categories for the function component of the user stories were harder to predict. Already the labelling of these components was more difficult. This is because – different from the user component – the information on the function was more distributed within the text of the user story descriptions. Hence, there are more n-grams that can be important for the categorization of the function component within a user story, which makes classification harder. We also found that some important categories could not be predicted. The function categories 0-3 that closely resemble CRUD patterns could not be predicted at all. This could be caused by either the low number of training samples from these categories or the fact that the information gained from the training examples could not be generalized. Still, the multilayer perceptron outperformed the naive Bayes approach.

Compared to the function part of the stories, the object categories are slightly easier to predict, but still harder than categories for the user component of the stories. For all categories except class 0 and 6, at least a decent classification was possible. Problems for predicting categorizes for this part of the user stories can be caused by the same issues as for the function component. Again, the multilayer perceptron classification of the object categories achieved better results than Naive Bayes.

Overall, we can conclude, that – as to expected for machine learning approaches – only categories with a sufficient sample size were easy to predict. But the number of samples is not the only requirement for successful categorization. Some categories with a decent number of training samples were hard to predict. To improve the usefulness and performance of the automatic categorization, the categories should probably be adjusted. However, it can be assumed that the possibilities for optimizing automatic categorization in this respect are limited. If the approach presented here is to be implemented more comprehensively in practice, it is likely that user story descriptions will have to be more consistently aligned to the Connextra template or an equivalent format. This needs to be done for the user stories archived in the repository to obtain larger amounts of suitable training data. A consistent use of such a format is also a requirement for new use cases to be categorized by the machine learning approach.

Regarding the usefulness of this approach for reuse of code, we can conclude, that categories important for implementation could be predicted successfully. Such predictions could be applied to categorize new user stories automatically without manual work. If used within a software solution for managing app development projects within an enterprise, our approach could be used to identify similar user stories. Such a similarity could indicate that a solution for a corresponding user requirement is already existing within the corporate context and might be available for reuse or could be adapted to the current problem. Since it is good practice to link user stories to the code that implements them, a developer can then use this system to discover source code and other artifacts (e.g., screen designs) in appropriate repositories. In addition to a reuse of technical artifacts, organizational information, such as contact details of responsible developers, can be identified if archived together with the user stories. Access to such organizational information can improve efficiency in the development of mobile enterprise applications by supporting the exchange of knowledge within large organisation where a direct exchange of knowledge is not an option.

8 Conclusion and Outlook

Strengthening reuse in mobile enterprise application development is a promising option to reduce costs and reduce time-to-market. In our work, we present an approach to this issue based on automatically categorizing user stories, so that related artifacts such as source code or design documents can be reused. We show that automatic categorization is possible and works especially well for user types of stories. For some classes, the categorization is not possible with our approach given the dataset we use. One approach to strengthen the categorization is to label more data. Future research needs to evaluate, whether this can improve categorization performance.

Some categories that can be automatically assigned are especially helpful for reuse. For example, looking at the category of platform-specific users (e.g., Android vs. iOS) can help developers dealing with platform-specific issues by reusing related code and knowledge. However, building a system that supports organizing access to user stories and supporting reuse requires more work: such a system should be integrated into platforms used for working with user stories such as Jira. Also, the categories we use

in this work should be reworked to improve categorization and allow generalization beyond the dataset used for our paper.

In future research, user story categories could be further examined: it might be interesting to categorize other parts of stories such as acceptance criteria, since they describe how a story should be implemented. It is also possible to use automatically assigned categories to compute story similarity in real-time, i.e. when entering a new user story, to automatically recommend similar stories to a story a developer is currently working on. Combining similarity measures from information retrieval and the neural network model for recommending similar stories is also a possible direction for future research.

9 References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Man´e, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Vi´egas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), http://tensorflow.org/, software available from tensorflow.org https://doi.org/10.1145/3190508.3190551

[2] Aldiabat, K., Al-Gasaymeh, A., & Rashid, A. K. (2019). The Effect of Mobile Banking Application on Customer Interaction in the Jordanian Banking Industry. https://doi.org/10.3991/ijim.v13i02.9262

[3] Anvik, J., Murphy, G.C.: Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions. ACM Transactions on Software Engineering and Methodology (TOSEM) 20(3), 10:1–10:35 (2011). https://doi.org/10.1145/2000791.2000794

[4] Atlassian: JIRA Software - Issue & Project Tracking for Software Teams. website, [retrieved: 2020.03.09] (2020), https://www.atlassian.com/software/jira

[5] Bernsteiner, R., Kilian, D., & Ebersberger, B. (2016). Mobile Cloud Computing for Enterprise Systems: A Conceptual Framework for Research. *International Journal of Interactive Mobile Technologies (iJIM)*, *10*(2), 72-76. https://doi.org/10.3991/ijim.v10i2.5511

[6] Chollet, F., et al.: Keras. https://keras.io (2015)

[7] Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA (2004)

[8] Cubranic, D., Murphy, G.C., Singer, J., Booth, K.S.: Hipikat: A project memory for software development. IEEE Trans. Softw. Eng. 31(6), 446–465 (Jun 2005). https://doi.org/10.1109/tse.2005.71

[9] Gasparic, M., Janes, A.: What recommendation systems for software engineering recommend: A systematic literature review. Journal of Systems and Software 113, 101–113 (2016) https://doi.org/10.1016/j.jss.2015.11.036

[10] Giessmann, A., Stanoevska-Slabeva, K., de Visser, B.: 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). https://doi.org/10.1109/hicss.2012.435

[11] Jabangwe, R., Edison, H., Nguyen, A.N.: Software engineering process models for mobile app development: A systematic literature review. Journal of Systems and Software 145, 98 – 111 (2018) https://doi.org/10.1016/j.jss.2018.08.028
[12] Jurafsky, D., Martin, J.H.: Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2009).
[13] Jurisch, M., Böhm, S., James-Schulz, T.: Mobile enterprise application development in practice: an analysis of real-world user stories. In: CENTRIC 2019, The Twelfth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services. pp. 31–36 (2019).
[14] Jurisch, M., Lusky, M., Igler, B., Böhm, S.: Evaluating a recommendation system for user stories in mobile enterprise application development. International Journal On Advances in Intelligent Systems 10(1 and 2), 40–47 (2017).
[15] Lusky, M., Jurisch, M., Böhm, S., Kahlcke, K.: Evaluating a User Story Based Recommendation System for Supporting Development Processes in Large Enterprises. In: CENTRIC 2018, The Eleventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services. pp. 14–18 (2018).
[16] Mani, S., Sankaran, A., Aralikatte, R.: Deeptriage: Exploring the effectiveness of deep learning for bug triaging. arXiv preprint arXiv:1801.01275 (2018) https://doi.org/10.1145/3297001.3297023
[17] Martin, J.: Managing the Data Base Environment. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn. (1983).
[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011) https://doi.org/10.3389/fninf.2014.00014
[19] Pirzadeh, H., Oliveira, A.D.S., Shanian, S.: ReUse: A Recommendation System for Implementing User Stories. In: International Conference on Software Engineering Advances. pp. 149–153 (2016).
[20] Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T.: Recommendation Systems in Software Engineering. Springer Publishing Company, Incorporated (2014).
[21] Runeson, P., Alexandersson, M., Nyholm, O.: Detection of duplicate defect reports using natural language processing. Proceedings - International Conference on Software Engineering pp. 499–508 (2007). https://doi.org/10.1109/icse.2007.32
[22] Vainio, T., Oksman, V., Wigelius, H., Markova, M., & Kulju, M. (2008). Exploring the Transformations of Interaction in Mobile Work Contexts. *International Journal of Interactive Mobile Technologies*, *2*(2).

10 Authors

Matthias Jurisch is a Research Assistant at RheinMain University of Applied Sciences and a doctoral student at Goethe University Frankfurt. His research interests are Software Engineering, Information Retrieval and Semantic Technologies. Email: matthias.jurisch@hs-rm.de

Prof. Dr. Stephan Böhm is a Professor of Telecommunications and Mobile Media at the Rhein Main University of Applied Sciences and co-founder of the Center for Advanced E-Business Studies (CAEBUS)in Wiesbaden. He teaches on media

technology and media management topics. His research focus is on media innovations and acceptance research.

Toby James-Schulz Toby is a bachelor student at Frankfurt University of Applied Sciences and a Research Assistant at Rhein Main University. His research interests are Chatbots, Blockchain and User Experience.

Article submitted 2020-04-28. Resubmitted 2020-05-22. Final acceptance 2020-05-24. Final version published as submitted by the authors.