Towards Boosting Requirements Engineering of a Health Monitoring App by Analysing Similar Apps: A Vision Paper

Jialiang Wei, Anne-Lise Courbis, Thomas Lambolais, Pierre Louis Bernard, Gérard Dray

Towards Boosting Requirements Engineering of a Health Monitoring App by Analysing Similar Apps: A Vision Paper

Jialiang Wei*, Anne-Lise Courbis*, Thomas Lambolais*,
Pierre Louis Bernard** and Gérard Dray*

*: EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France
**: EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Montpellier, France
*: firstname.lastname@mines-ales.fr **: firstname.lastname@umontpellier.fr

***Abstract*—Due to the variety of sensors and portability of mobile phones, an increasing amount of mobile apps are released for the purpose of health monitoring. To design a new health monitoring app, a conventional approach is to define a goal model with the intervention of stakeholders. As there are a large number of apps in this domain, many of them have similar features, which can be exploited to reduce the time consumption of requirements elicitation, improve their quality and help the requirements prioritization. In this paper, we propose a novel requirements engineering approach by analysing similar apps. In this approach, we identify similar apps and analyze their descriptions, user reviews, Android APK and source code for the construction and enrichment of a Domain Feature Model. This model will be used as a support for the requirements engineers in different phases of requirements engineering.**

I. Introduction

The proportion of senior citizens in the population is growing as people worldwide are living longer. The effects of aging are significant in the area of functional independence. From a motor point of view, aging is active on all of the functions on which stabilization, locomotion and grip capacities depend. Digital devices are a privileged means of evaluating behavior and proposing adapted and individualized solutions [1]. For the purpose of health monitoring of seniors, we are going to develop a system including a mobile app. It is not easy to develop such an app from scratch, while similar apps can be a good reference for the requirements engineers. Requirements of similar apps can be extracted and reused for the production of a new app. However, the System Requirements Specifications (SRS) of these apps are usually not publicly available due to the organizational privacy policies. The external researchers can hardly access and explore these documents. As alternative, we identify similar apps from app stores and GitHub, and analyse the descriptions, user reviews, Android APK and the source code to support the requirements engineering of an app. Analysing similar apps can help the requirements engineers gain a domain knowledge, identify common features, bugs and usage scenarios, know the users' reaction to existing features [2] and conduct a competitor analysis [3].

In this paper, we define similar apps as apps with similar features. The similar apps can be found at app stores and GitHub. Online app stores, like Google Play, Apple Store, provide useful data resource for analysing product features: for one thing, app descriptions introduce the main features of the users; for another, the app user reviews show the reaction of user on existing features. We will evaluate our approach using Google Play, as there were 3.48 million apps in 2021, which makes Google Play the app store with biggest number of available apps[1]. GitHub is another source for identifying similar products, as of May 2022, GitHub reported over 401K open source repositories related to iOS[2] and 1.29 million repositories that relate to Android[3]. The code and documents on GitHub is another resource for requirements reuse.

Faced with so many apps, it is difficult for the requirements engineers to find useful information. To illustrate the problem, consider the following scenario. Jay is a requirements engineer of GoldenAge, a mobile app for health monitoring of seniors. He wants to use a prototype to conduct the discussion with stakeholders, but the construction of such a prototype is time consuming. An alternative is to use similar apps as prototypes, the features of these apps can inspire more ideas while brainstorming. The user reviews of these similar apps can also help the requirements prioritization. In this process, he encounters at least four difficulties:

1) How to **find the similar apps** from millions of apps? The app stores and GitHub provide search function, and most app stores have categorized the apps, but it is hard to get similar apps through search engine and categorization.
2) How to **extract features** from these apps? The manual extraction of features from the app descriptions is tedious. Is it possible to extract also the User Interface (UI) and source code of corresponding features?
3) How to get **user reactions** for existing features? Due to the large number of user reviews, manual analysis can be hardly finished in an acceptable time.

[1]https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
[2]https://github.com/search?q=ios
[3]https://github.com/search?q=android

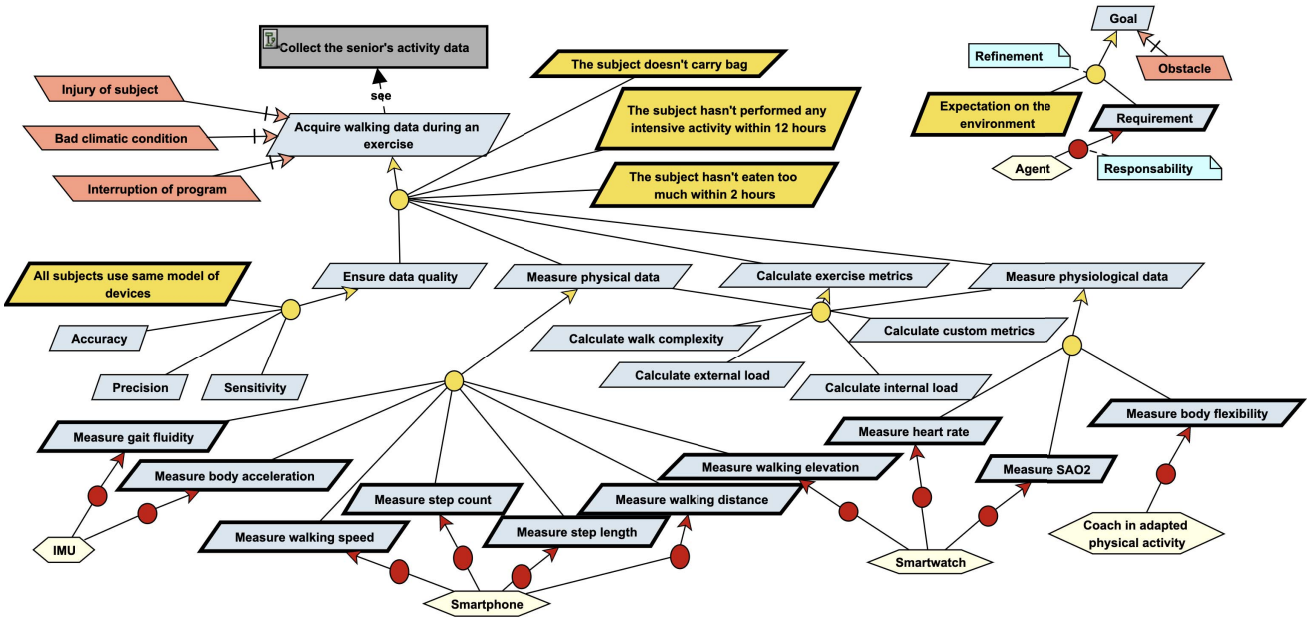

Fig. 1. Excerpt of the Goal Model of GoldenAge

4) How to **present the extracted information**? The extracted information can be chaotic, and difficult to comprehend by human mind. Jay wants them to be readable to facilitate his communication with team members.

In this paper, we propose an approach to analyze the data (app descriptions, UI, user reviews and source code) in app stores and GitHub for supporting the design of Jay's product. Data extracted from similar apps are formalized into a Domain Feature Model (DFM). The constructed DFM can be served for the analysis of strengths and drawbacks of existing features in similar apps, inspiration of new ideas, requirements validation and requirements prioritization. Matching the DFM with a goal model will help us identify our original features.

This paper is structured as follows: Section II presents the target app; Section III describes our proposed approach in detail; Section IV presents the related work and finally, a conclusion is provided at the end of paper.

II. Our target app

As in a conventional approach of requirements engineering, the first step is collecting, examining and synthesising the information related to the system for background study. Then the requirements engineers conduct interviews with selected stakeholders, which include end users, clients, managers, etc. [4]. In this phase, brainstorming is used for exploiting new ideas and prototypes are used to deal with uncertainty [5]. Then different models are constructed for the abstract descriptions of the elicited requirements. The object model describes *what* constitutes the environment and the system-to-be, the operation model presents *how* the system operates and the goal model focuses on *why* the system is wanted. Finally, the requirements engineers write the SRS based on these models and improve the SRS according to the stakeholders' feedback.

We consider the development of an application for the health monitoring of seniors by gerontologists in a controlled environment. For the first requirements elicitation and requirements analysis phase, we still apply the conventional method to develop a goal model. A goal is an objective that the system-to-be should achieve. A goal model is a graph, whose nodes are goals. It is built by asking '*for what purpose*?' and '*in what way*?' from a root node [6]. For our app, we use the KAOS method [7], supported by the objectiver tool, for the construction of the goal model.

The Fig. 1 shows an extract of our goal model. Note that in the diagram, goals only appear with their title, but a more precise description is associated (definition, category, pattern). The root goal of this model is to collect seniors' activity data. One of the activities is walking. The obstacles of acquiring *walking* exercise data are the injury of the subject, bad climatic conditions and interruption of the program. To acquire walking exercise data, we suppose that the subject does not carry any bag while walking, has not performed intensive activity and has not eaten too much (these two assumptions are more precisely defined in the goal model, indicating for instance that last meal has been taken more than two hours ago, and that no activity during more than 1h30 at 80% of intensity has been performed in the last 24h), because these actions would impact the performance during walking exercise. Then, the goal of acquiring walking data is refined into four sub-goals, ensuring data quality, measuring physical data, calculating exercise data and measuring physiological data. The data quality is declinated into accuracy, precision and sensitivity goals, and it assumes that all subjects use the same model of devices.

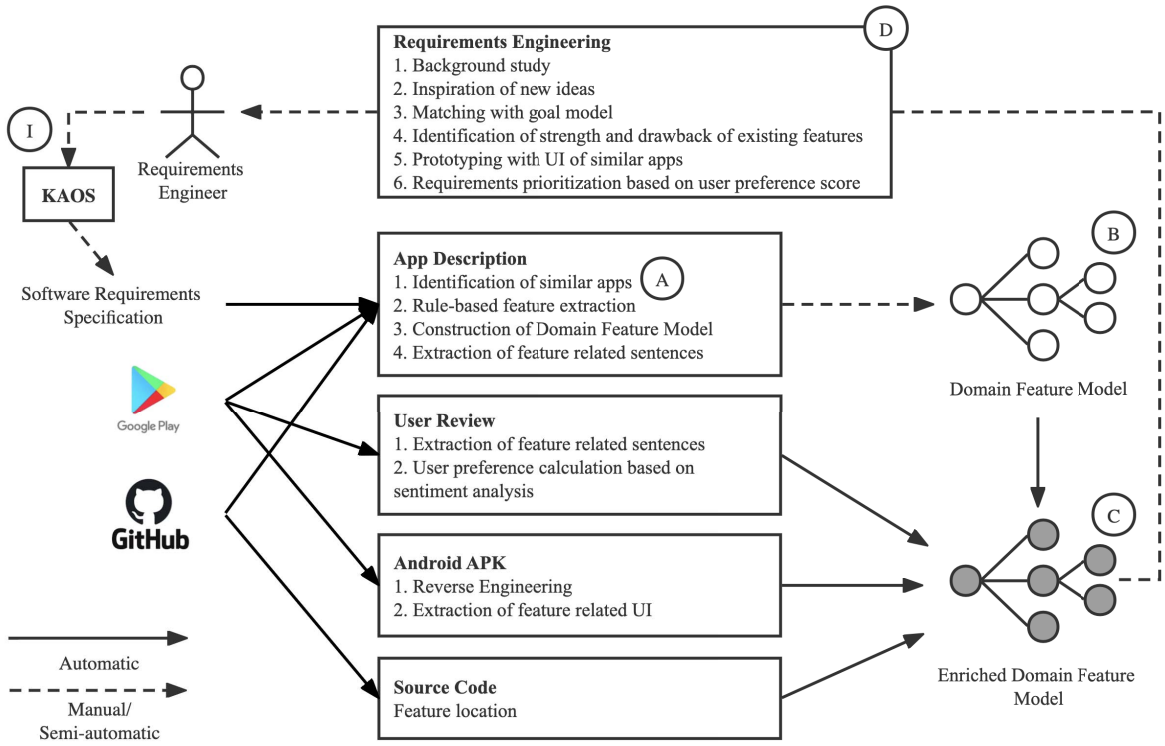

Fig. 2. Overview of our proposed approach

Meanwhile, for the three other sub-goals, different data types and the devices/specialists that collect these data are specified.

In the health and aging domain, it is appropriate to enrich this approach by analysing existing applications for the following reasons: there are more than 120 thousand health & fitness apps in Google Play[4], which is a rich source for mining health related app features; the seniors are more fragile, an analysis of user reviews can reduce the potential risk (e.g. the risk of physical exercises) of health related apps; seniors may have difficulties with new technologies, hence analysing the reviews is a way for enhancing non functional requirements. The SRS obtained in the conventional approach will be used as the input of our proposed approach for the identification of similar apps.

III. Overview of the Proposed Approach

The goal of our research is to develop a new paradigm of data driven requirements engineering for health aging. In our approach, requirements engineers can easily construct a DFM with our provided tool, and apply the model for requirements elicitation, requirements analysis and requirements validation. As shown in Fig. 2, it contains five steps: I. initialization, which corresponds to the conventional approach using KAOS presented above, A. identifying similar apps, B. constructing DFM, C. enriching DFM and D. integrating into requirements engineering.

A. Identifying similar apps

We want to find the apps with similar features of our target app. As the goal is to develop an app for the health monitoring of seniors, one source of similar apps is the apps that are categorized as "*Health & Fitness*" by Google Play or the projects that contain the keyword "*Health monitoring*" on GitHub. However, the apps that belong to the same category and contain same keywords may be totally different. For example, the search results for the keyword "*walk*" on Google Play include *Walk with Map My Walk*[5] and *Walking*[6]. These two apps are both categorized as "*Health & Fitness*" by Google Play, while the former app is for fitness training and the latter app is used to control walking machines.

As the categorization and the search engine in app stores and GitHub do not exhibit a good quality in finding similar apps, we use topic modeling [8]. In Natural Language Processing (NLP), a topic model is a statistical model used to discover the abstract "topics" in a series of documents. Latent Dirichlet Allocation (LDA) is a topic model, which can give the topics of each document in the document set in the form of a probability distribution. A topic is identified as a set of terms that describe a theme. The result of running LDA on a set of documents is the probability of relevance of each document to each generated topic and the distribution of each topic over a set of terms in the corpus. Many researchers used this

[4]https://www.statista.com/statistics/279286/google-play-android-app-categories/

[5]https://play.google.com/store/apps/details?id=com.mapmywalk.android2

[6]https://play.google.com/store/apps/details?id=com.walking.ble

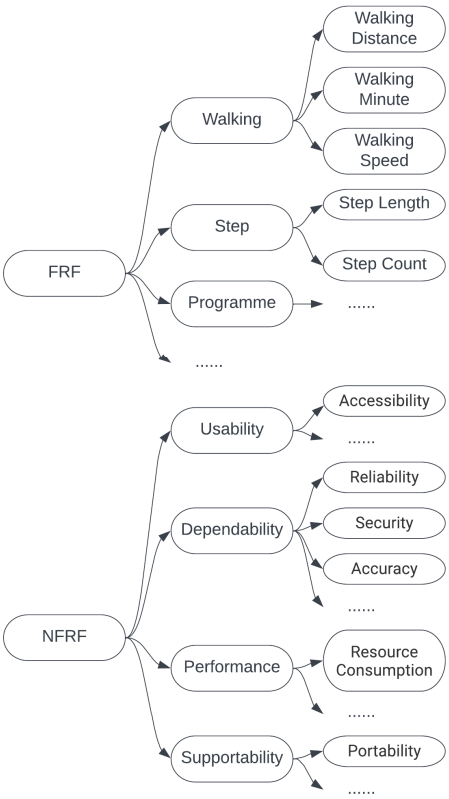

Fig. 3. Example of a Domain Feature Model

algorithm to generate the topic distribution of app descriptions and measure the similarity between apps [9]–[11].

After background study and interviews with stakeholders, a first draft of SRS for the app-to-be is written. We thus apply LDA on the SRS and get the topic distribution. Then we use LDA on the descriptions of collected apps and get the topic distribution of each app. Finally, we calculate the cosine similarity of the topic distribution between our SRS and the collected app descriptions, the top n apps are considered as similar apps.

B. Constructing a domain feature model

Our Domain Feature Model (DFM) is a graph that represents the existing features of similar apps, unlike the feature framework of Liu et al. [11] which contains only Function Related Features (FRF), DFM contains two tree structures, one for the FRF, the other for Non-Function Related Features (NFRF). The model presents the domain knowledge mined from the similar apps. An example of DFM is shown in Fig. 3.

As the Non-Functional Requirements (NFR) are common in different categories of mobile apps, we use the classification of Jha et al. [12] for the construction of the NFRF tree of the model. The first level of NFRF tree is a root node, while the second level contains usability, dependability, performance and supportability.

The construction of FRF tree comprises two steps: 1) feature extraction and 2) feature integration.

1) Feature Extraction: This step aims at extracting feature related text from the app description. Johann et al. [13] proposed the SAFE approach, which contains 18 Part-of-Speech (POS) patterns and 5 sentence patterns that denote app features. Liu et al. [11] used keyword-based linguistic rules to extract features from release text. Wu et al. [14] introduced KEFE to identify key features from user reviews. KEFE extracts features via a textual pattern-based filter and a deep learning classifier. In our approach, we use POS patterns to extract FRF from the app descriptions. Three examples are shown in Table I.

POS pattern	Extracted Features
NOUN NOUN	distance tracker
VERB NOUN	lose fat
ADJ NOUN	indoor workout

TABLE I
EXAMPLES OF POS PATTERNS

2) Feature Integration: The next step is to integrate the extracted features into the DFM. As shown in Fig. 3, the FRF tree contains three levels. The first level is a root node. The second level is the feature level highlighting high level features built from the top n most frequent noun parts of extracted features. The third level is a specific level. It is the extension of the second level obtained by searching features that contain the second level nouns.

C. Enriching domain feature model

The DFM constructed in the previous step is not very helpful for the requirements engineers as it includes only the feature name without any detail. For example, a node of DFM is "step count". From this name, we cannot know the details, the implementation, the UI nor the user preferences of this feature in different apps. However, it is the backbone leading the subsequent analysis.

1) App description: The app descriptions often contain some details about the features. For example, in the description of app *Step Counter — Pedometer, MStep*[7], the feature "*step count*" is described as follows: "*This step counter uses the built-in sensor to count your steps*", "*You can pause and start step counter at any time to save power*". These sentences can be extracted automatically via keyword based search. Each node of the third level of DFM tree is used as keyword of search.

2) User reviews: The user preference towards certain features are expressed via the rating and the content of user reviews. It is an important indicator for requirements prioritization. The user reviews can be classified automatically into several classes [15], including *user experience* and *bug report*. These two classes contain the praise or criticism towards the app features. As in the analysis of app description, we use the third level nodes of the DFM tree as keyword to search feature related sentences from the user experience and bug

[7]https://play.google.com/store/apps/details?id=pedometer.steptracker.calorieburner.stepcounter

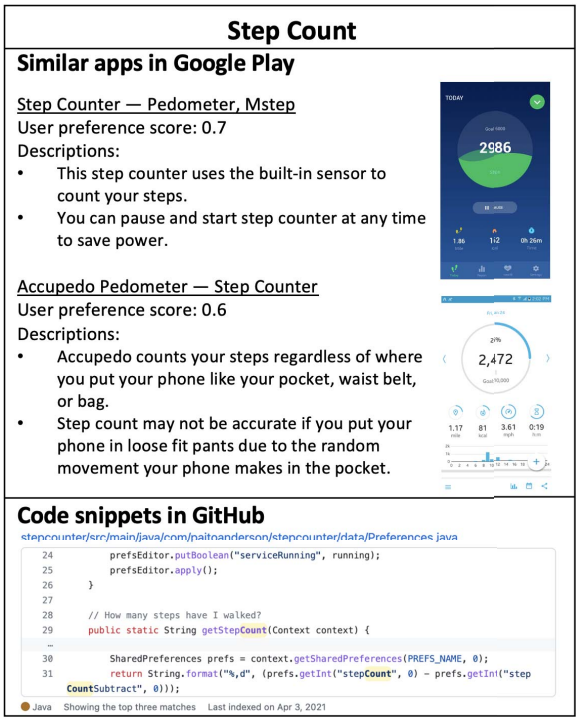

Fig. 4. Example of a node in Enriched Domain Feature Model

report. Then, the feature and the feature related sentences for each app is mapped. Sentiment analysis is an NLP method, it can be used to quantify the opinion of the user reviews [16]. The result of running sentiment analysis method to a text is the polarity and subjectivity. Polarity shows if the text is a positive of negative statement, while the subjectivity indicates whether the text is a personal opinion or a factual information. With the combination of user rating, polarity and subjectivity, we can calculate the user preference score towards a feature.

3) User interface: Most app features are associated with UI. The UI of similar apps can inspire the elicitation of new requirements while brainstorming with stakeholders. Chen et al. [17] proposed an approach for recommending app features base on UI comparison. Android app reverse engineering tools, such as Apktool[8], can convert the executable source code to a human-readable form [18]. For each similar app, we apply Apktool to extract the UI elements from the Android APK. Then for each feature, the related UI can be identified by comparing the UI textual elements and the feature name.

4) Source code: Code reuse is beneficial for code quality, coding efficiency, and maintenance [19]. GitHub is the biggest open source platform where we can find many similar apps. However, we can hardly find apps on GitHub that exactly meet our needs. To tailor the similar apps on GitHub to our need, we first need to identify the location in the source code that corresponds to a specific functionality, which is known as *feature location* [20]. The input of feature location is the source code and the textual description of a feature. Among feature location techniques, the output of running feature location are portions of source code at different levels of granularity (files, classes, methods or functions, and statements).

[8]https://ibotpeaches.github.io/Apktool/

D. Integrating into requirements engineering

The Enriched Domain Feature Model (EDFM) constructed in the previous steps can be served in different phases of requirements engineering. Here are some scenarios for usage:

1) Requirements elicitation: The EDFM contains a rich domain knowledge, it is a good reference for the background study. By showing the features of similar apps during brainstorming, the stakeholders will get inspired.

2) Requirements analysis: With EDFM, we can identify the strengths and drawbacks of existing features in similar apps. The EDFM can also be matched with the goal model of system-to-be, in order to help the requirements engineers to identify which features are original and which features are widespread in similar apps but have not been considered before.

3) Requirements validation: Validating whether natural language requirements meet the needs of the stakeholders is often difficult and error-prone. Prototyping is an effective way for requirements validation [21]. However, the construction of prototypes is time consuming and costly. With EDFM, the requirements engineers can find the feature related UI or code easily and use them for the discussion with stakeholders.

4) Requirements prioritization: Satisfying all requirements may be infeasible given budget and schedule constraints, and as a result, requirements prioritization may become necessary [4]. Although user feedback does not reflect preferences for features that do not exist in similar apps, the user preference score is still an important indicator for the requirements prioritization.

The SRS updated during these phases will be used again as the input of similar apps identification.

IV. Related Work

To support the development of mobile apps, many researchers mine various data from similar apps. Liu et al. [11], [22], [23] mined descriptions and user reviews of similar apps for background study, support of feature updating and functions comparison. Jiang et al. [24] proposed SAFER to recommend new features by inspecting the descriptions of similar apps. Malik et al. [25] presented a methodology to aid users and developers to compare the features across multiple apps by analyzing user reviews. Assi et al. [26] introduced a review analysis method called FeatCompare, which identifies high-level features mentioned in user reviews and creates a comparison table summarizing users' perceptions of each identified feature in competing apps. Chen et al. [17] proposed an approach to recommended app features through UI comparison of similar apps. The above work proves that data from app stores can provide valuable information for requirements elicitation, while other phases of requirements engineering are paid less attention. Our work analyzes various data from app stores and GitHub, and apply it throughout the whole requirements engineering process.

V. Conclusion

Mobile devices are widely used for the health monitoring of seniors, a great number of apps in this domain can be found in app stores. In this paper, we propose an initial design of an approach for boosting requirements engineering by analysing similar apps. In our approach, we identify similar apps automatically by calculating similarity between app descriptions and the software requirement specifications that we wrote for our health monitoring app. Then, a Domain Feature Model (DFM) containing the features of similar apps is constructed based on the descriptions. Finally, the DFM is enriched on four aspects: feature descriptions from app descriptions; user preference towards features; UI associated to features; and code snippets corresponding to features. The enriched DFM can be used in different phases of requirements elicitation, requirements analysis, requirements validation and requirements prioritization.

Our future work will be focused on the development of a tool to support our approach and integrating it into the requirements engineering process of health monitoring mobile apps. At the first stage, we work on the construction of DFM and the integration of user reviews. In our previous work [27], we proposed a model which classified automatically user reviews from health monitoring apps into *feature request, bug report, user experience* and *rating*. The main challenge lies in the automatic determination of relationships between the app features in the DFM.

Acknowledgements

We thank Robert Darimont (Respect-it—Requirements Engineering and Specification Techniques for Information Technology, Belgium), who is one of the creators of the KAOS method, for his collaboration on the Goal Model design with KAOS and the Objectiver tool.

References

[1] G. Guarino de Moura Sá, L. Fernanda Silva, A. M. Ribeiro Dos Santos, J. Dos Santos Nolêto, M. Teles de Oliveira Gouveia, and L. Tolstenko Nogueira, "Technologies that promote health education for the community elderly: Integrative review," *Revista Latino-Americana de Enfermagem*, vol. 27, 2019.

[2] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, and M. Harman, "App Store Effects on Software Engineering Practices," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 300–319, 2021.

[3] F. Dalpiaz and M. Parente, "RE-SWOT: From User Feedback to Requirements via Competitor Analysis," in *Requirements Engineering: Foundation for Software Quality* (E. Knauss and M. Goedicke, eds.), (Cham), pp. 55–70, Springer International Publishing, 2019.

[4] A. Bennaceur, T. Than Tun, Y. Yu, and B. Nuseibeh, "Requirements Engineering," in *Handbook of Software Engineering*, pp. 1–44, springer, ed., 2019.

[5] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, (New York, NY, USA), pp. 35–46, Association for Computing Machinery, 2000.

[6] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 249–261, 2001.

[7] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, Jan. 2009.

[8] D. Blei, L. Carin, and D. Dunson, "Probabilistic topic models," *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 55–65, 2010.

[9] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking App Behavior against App Descriptions," in *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, (New York, NY, USA), pp. 1025–1035, Association for Computing Machinery, 2014.

[10] A. Al-Subaihin, F. Sarro, S. Black, and L. Capra, "Empirical comparison of text-based mobile apps similarity measurement techniques," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3290–3315, 2019.

[11] H. Liu, Y. Wang, Y. Liu, and S. Gao, "Supporting features updating of apps by analyzing similar products in App stores," *Information Sciences*, vol. 580, pp. 129–151, 2021.

[12] N. Jha and A. Mahmoud, "Mining non-functional requirements from App store reviews," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3659–3695, 2019.

[13] T. Johann, C. Stanik, A. M. Alizadeh B., and W. Maalej, "SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 21–30, 2017.

[14] H. Wu, W. Deng, X. Niu, and C. Nie, "Identifying key features from app user reviews," *Proceedings - International Conference on Software Engineering*, pp. 922–932, 2021.

[15] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[16] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[17] X. Chen, Q. Zou, B. Fan, Z. Zheng, and X. Luo, "Recommending software features for mobile applications based on user interface comparison," *Requirements Engineering*, vol. 24, no. 4, pp. 545–559, 2019.

[18] Y. L. Arnatovich, L. Wang, N. M. Ngo, and C. Soh, "A comparison of android reverse engineering tools via program behaviors validation based on intermediate languages transformation," *IEEE Access*, vol. 6, pp. 12382–12394, 2018.

[19] M. Gharehyazie, B. Ray, and V. Filkov, "Some from Here, Some from There: Cross-Project Code Reuse in GitHub," *IEEE International Working Conference on Mining Software Repositories*, pp. 291–301, 2017.

[20] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: a taxonomy and survey," *Journal of Software: Evolution and Process*, vol. 25, pp. 53–95, jan 2013.

[21] D. Aceituna, H. Do, and S.-W. Lee, "Interactive requirements validation for reactive systems through virtual requirements prototype," in *2011 Model-Driven Requirements Engineering Workshop*, pp. 1–10, 2011.

[22] Y. Liu, L. Liu, H. Liu, X. Wang, and H. Yang, "Mining domain knowledge from app descriptions," *Journal of Systems and Software*, vol. 133, pp. 126–144, 2017.

[23] H. Liu, X. Yin, S. Song, S. Gao, and M. Zhang, "Mining detailed information from the description for App functions comparison," *IET Software*, vol. 16, no. 1, pp. 94–110, 2021.

[24] H. Jiang, J. Zhang, X. Li, Z. Ren, D. Lo, X. Wu, and Z. Luo, "Recommending New Features from Mobile App Descriptions," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, oct 2019.

[25] H. Malik, E. M. Shakshuki, and W. S. Yoo, "Comparing mobile apps by identifying 'Hot' features," *Future Generation Computer Systems*, vol. 107, pp. 659–669, 2020.

[26] M. Assi, S. Hassan, Y. Tian, and Y. Zou, "FeatCompare: Feature comparison for competing mobile apps leveraging user reviews," *Empirical Software Engineering*, vol. 26, no. 5, 2021.

[27] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, and G. Dray, "Towards a data-driven requirements engineering approach: Automatic analysis of user reviews," in *7th National Conference on Practical Applications of Artificial Intelligence, APIA 2022*, pp. 102–107, 2022.