Towards Automatically Identifying Potential Sustainability Effects of Requirements

Iris Groher
Johannes Kepler University
Linz, Austria
iris.groher@jku.at

Norbert Seyff
FHNW
Windisch, Switzerland
University of Zurich
Zurich, Switzerland
norbert.seyff@fhnw.ch

Tahira Iqbal
fortiss GmbH
Munich, Germany
iqbal@fortiss.org

***Abstract*—Software developers are gradually becoming aware that their systems have effects on sustainability. The identification of potential effects software-intensive systems can have on different sustainability dimensions over time is yet in its infancy. Researchers are currently exploring approaches which strongly make use of expert knowledge to identify potential effects. In this work in progress paper, we are looking at the problem from a different angle: we report on the exploration of a machine learning-based approach to identify potential effects. Such an approach allows to save time and costs but increases the risk that potential effects are overseen. First results of applying the machine learning-based approach in the domain of home automation systems are promising, but also indicate that further research is needed before our approach can be applied in practice. Furthermore, we have learned that even providing the ground truth for training the algorithms is a challenging task.**

***Index Terms*—Sustainability, Analysis, Requirements, Machine Learning.**

I. Introduction

Software-intensive systems do not operate in isolation but in complex socio-technical contexts. Therefore, they have an impact on this context, manifesting itself in different dimensions such as the environmental, economic, social, individual, and technical dimension [1]. As effects can occur over time, we can also identify three different orders of effects [2] [3]. The cumulative positive and negative effects a software-intensive system has on its context define its sustainability.

In previous research [4], we have learned that practitioners are not aware of the fact that software-intensive systems have an impact on sustainability and that raising awareness is an essential step towards the development of sustainable software-intensive systems. Furthermore, the complexity of this matter and the lack of adequate methods and tools supporting the identification of potential effects are hurdles for practitioners who are already aware of their responsibility to build sustainable systems.

Requirements are the key to sustainability [1], which indicates that the identification of potential sustainability effects needs to start before systems are actually built or when certain aspects of a system are modified in the context of system evolution. This, in part, also makes the identification of potential effects a hypothetical endeavour, which often needs to be based on expert opinions rather than facts. Only after development, when the system is used in its application context, one can eventually validate its effects on the different sustainability dimensions over time.

Researchers have started to build methods and tools to support the identification of sustainability effects [5] [6] [7]. Although such approaches can be successfully applied to identify potential sustainability effects, it appears that they require a significant time investment of companies, which might prevent their adoption.

In this work in progress paper, we follow a current trend in software and requirements engineering and propose the use of Machine Learning (ML) for the early identification of potential sustainability effects. In this paper, we present this idea in more detail and also report on a first application experiment in the home automation domain. Based on requirements for home automation systems, we have identified potential sustainability effects and have used these results as ground truth for training our algorithms. Early results indicate that using ML for the identification of sustainability effects is promising, which motivates us to continue with this research.

In Section 2 we discuss existing approaches for identifying sustainability effects in requirements. In Section 3 we present our ML-based approach and report on a first experiment we conducted in Section 4. In Section 5 we conclude the paper and present an outlook on next steps we plan.

II. Existing Sustainability Effect Identification Approaches and Tools

The work presented in this paper is motivated by research in the field of requirements engineering, where researchers aim at identifying potential sustainability effects.

In previous work on tailoring requirements negotiation to sustainability [5], an extension of the WinWin negotiation model was proposed. This approach incorporates sustainability so that the negotiation is used to identify potential effects of requirements on sustainability. For requirements which might have negative effects, alternative requirements options are

discussed to minimize these negative effects. This method was applied in an exploratory industrial case study, where it allowed practitioners to reflect on requirements and their effects on sustainability.

Recent work presents a question-based framework for raising awareness of the potential effects of software systems on sustainability [6]. The Sustainability Awareness Framework (SusAF) was used by students to carry out interviews for a software system of their choice to identify potential effects of these systems on sustainability and in particular even identify potential chain of effects. Results from this feasibility study indicate that SusAF stimulates the discussion about potential effects of software systems on sustainably.

Alharthi et al. [7] present the SuSoftPro tool, which supports the analysis of the impact of requirements on different sustainability dimensions via a Fuzzy Rating Scale method. This tool-supported approach allows for different visualization option of the results (e.g., a bar graph that illustrates the sustainability level).

We conclude that researchers have identified the need for identifying sustainability effects and that first promising methods and even tool-supported approaches are appearing. However, all presented methods strongly depend on human involvement and might require time-intensive discussions, the involvement of experts or a large number of people to derive results. The quality of the produced results might further vary a lot depending on different factors such as the complexity of the domain and the level of expertise of the people involved. Nevertheless, bespoke methods can raise awareness and can help, at least in part, to improve the sustainability of software-intensive systems.

III. Machine Learning-Based Effect Identification

The goal of our ongoing research is to automate the identification of potential sustainability effects by analysing the requirements of a software-intensive system. In contrast to existing methods in place, we expect that such an approach will result in the significant reduction of the effort needed for the analysis. However, we also see the risks that such an approach might result in overlooking potential effects.

A. ML in Software and Requirements Engineering

To achieve this goal, we follow a recent trend in software and requirements engineering and explore the application of ML [8] [9]. ML has already been successfully used to classify software requirements into functional and non-functional requirements [10] [11]. The analysis of a large number of user feedback from multiple sources such as the app store and Twitter has been automated by applying ML [12] [13]. This analysis helps to identify useful information such as bug reports and feature requests to support software evolution. For validating requirements, automated analysis of requirement traceability with the help of natural language processing and ML has been studied [14] [15]. ML has also been applied in requirements management such as visualizing requirements on different levels of granularity and prioritizing requirements [16] [17].

B. ML Application Overview

To automatically identify the impact of requirements on sustainability we follow the ML workflow [18], as shown in Fig. 1.

The first key step is data preprocessing that helps to avoid data incompleteness and inconsistency issues. This data is used as an input for the ML algorithm, which means that the ML algorithm learns from existing data. On the basis of this learning, a learning model is produced as output, which can be used to make predictions on a different dataset than the one used for training. The learning model can be evaluated based on its performance. For measuring the performance, we use well-known ML parameters such as precision, recall, accuracy, and F-score.

In the next section, we will describe how we have performed the above discussed steps in our experiment.

IV. A First Experiment

We performed a first experiment in the domain of home automation systems to evaluate our ML-based approach. In the next subsection, we describe the setup of our experiment and in the subsequent subsection we present our preliminary results.

A. Setup

Data: The dataset used for training and evaluating our ML-based approach is comprised of publicly available smart home requirements[1]. The requirements were collected as part of research on crowd-RE [19].

In a first step, three annotators manually classified 200 randomly selected requirements from the total set of available requirements (around 2900). All three annotators had proficient knowledge and expertise for sustainable systems in software engineering. For each requirement, each annotator independently marked which sustainability dimension(s) it had an effect on. To support this classification, a literature review has been performed on sustainability dimensions and created a classification guideline based on the results of this analysis. The guideline contained for each dimension a set of influence factors and for each factor a description, rational, example requirements, and literature references for further reading. Table I shows an example influence factor in the environmental dimension.

The annotators used the guideline as a reference during the manual classification of the 200 smart home requirements. Each requirement was independently classified according to its influence on the five dimensions of software sustainability as positive, negative, or neutral. The plus sign (+) was assigned for positive influence, a minus sign (-) for negative influence, and no sign for indicating no influence as shown in Table II. The ratings were merged and cases in which the researchers did not agree were discussed until consensus was reached.

[1]Smarthome Crowd Requirements Dataset, https://crowdre.github.io/murukannaiah-smarthome-requirements-dataset/

Fig. 1. Basic ML workflow steps

TABLE I
EXAMPLE INFLUENCE FACTOR

Dimension	Environment
Factor	Recycling
Description	Process of converting waste materials into new materials and objects
Influence/Rational	Strong positive influence on the environment as not only the amount of waste is reduced but also the farming of natural resources to create new products is decreased.
Example	If a device needs to be removed from the system, the system should provide information on how to dispose it properly.

B. Application

Pre-processing: We applied natural language processing techniques on our data before applying the different ML algorithms. First, we applied text tokenization on each requirement. Then we eliminated all stop words and converted the text into small characters. We applied stemming as our next step. As the last step, we converted pre-processed text as a vector space model using Term Frequency-Inverse Document Frequency (TF-ID or TF-IDF) as a weighting scheme:

$$tfidf(t,d) = tf(t,d) * idf \tag{1}$$

Here t is a term in a vector and d is a requirement in a collection of requirements [20].

Classification: For the automated classification of sustainability requirements and their dimensions, we trained our model using the annotated requirements dataset. We implemented Nave Bayes (NB), k-Nearest Neighbor (KNN), Decision trees (DT), Support Vector Machine (SVM), Logistic regression (LR), and Neural Network (NN) algorithms and also trained our classifier with and without stemming the data, as discussed in [2]. The results were quite similar with a minor difference and we used stemmed data for final analysis. We used tenfold cross-validation for evaluating our results.

C. Results

The results for six different classifiers from our experiments are shown in Table III. For choosing the best classifier, we evaluated the performance of all these classifiers on the basis of commonly used ML metrics i.e., accuracy, precision, recall, and F1-score. These metrics can be calculated using the following formulae:

$$Accuracy = TP + TN/TP + FP + FN + TN \tag{2}$$

$$Precision = TP/(TP + FP) \tag{3}$$

$$Recall = TP/(TP + FN) \tag{4}$$

$$F1 - score = Precision/Recall \tag{5}$$

Here, TP (True Positive) is the number of requirements correctly classified as belonging to a category. TN (True Negative) is the number of requirements that are correctly classified as not belonging to a category. FP (False Positive) is the number of requirements incorrectly classified as belonging to a category, and FN (False Negative) is the number of requirements that are incorrectly classified as not belonging to a category.

In simple words, accuracy is the ratio of correctly classified data over total data. This helps to predict the model performance whereas high accuracy results in a better performance of the model. However, data can be asymmetric and thus parameters other than accuracy need to be evaluated. The precision metric refers to the ratio of correctly predicted positive values to the total number of predicted positive values. On the other hand, recall is the ratio of total predicted positive values to the actual number of positive values. It is not possible to maximize both recall and precision metrics at the same time, as one comes at the cost of another. To consider both, F1-score is used which is the harmonic mean of precision and recall.

The highest accuracy and F1-score decide which algorithm is the best among others. We achieved the highest accuracy with DT classifier (70% precision) followed by SVM (69% precision). Our dataset was not balanced, meaning that the five different sustainability dimensions were not equally represented in the dataset (see Fig. 2). The economic, environmental, and social dimensions were almost equally represented. The individual dimension had high occurrences and the technical dimension was almost inexistent. To overcome this problem, we used the weighting technique by assigning more weight to fewer data. After applying this setting, we achieved the highest accuracy for SVM (75%), as shown in column SVM (b) of Table III. Recall and precision are 63% and 57% respectively, which is acceptable according to our accuracy. We also calculated the F1-scores, and the highest value was achieved with SVM (60%).

The results from this initial experiment can be improved further as we observed some issues that are impacting our results. The structure of our dataset varied with respect to the length of the textual requirements. For example, one requirement consisted of 20 words, and another one consisted of 200 words. Due to the significant difference regarding their length, our ML classifier features were sparse, which might have lead to an underfitted model.

TABLE II
MANUALLY CLASSIFIED SMART HOME REQUIREMENTS

ID	Req	T	I	S	Ec	En
1	Music should be available throughout the house	-	+		-	-
2	Temperature in the house should be adjusted based on the weather outside		+		+	+
3	The lights shut be shut off in the rooms with nobody in them		+		+	+
4	The garage door should be opened when it senses my vehicle arriving outside of it		+			

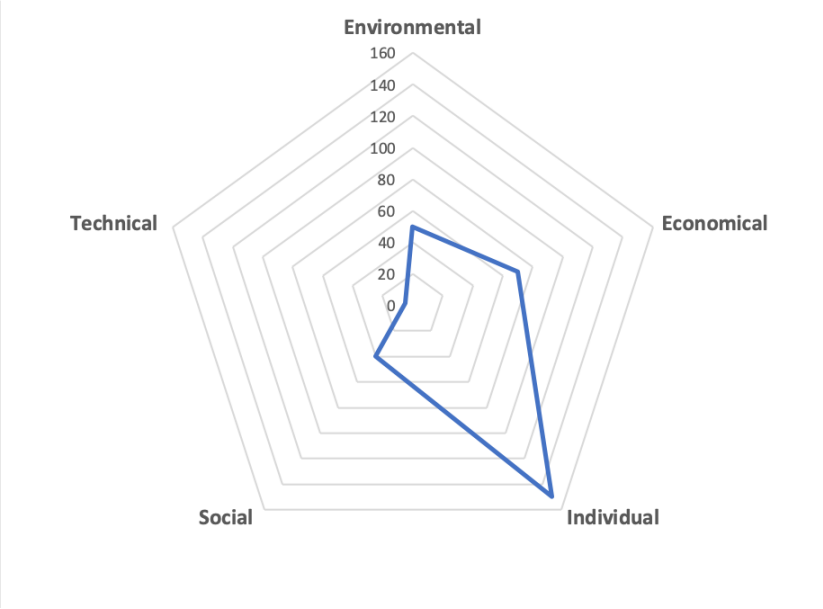

Fig. 2. Distribution of sustainability dimensions

TABLE III
SCORES FOR THE DIFFERENT CLASSIFIERS

	NB	KNN	DT	LR	SVM	SVM (b)	NN
Accuracy	0.58	0.65	0.7 (0.03)	0.62	0.67	0.75	0.69
Precision	0.46	0.51	0.46	0.57	0.5	0.57	0.44
Recall	0.43	0.56	0.56	0.54	0.48	0.63	0.47
F-score	0.44	0.53	0.51	0.56	0.48	0.60	0.45

Moreover, our negative and positive influence values on the sustainability dimensions were also not equally distributed. Our data only contained 12 requirements with negative influence, the rest were positive influences.

V. CONCLUSION AND NEXT STEPS

The goal of our ongoing research is the automation of the identification of requirements, which potentially have effects on the sustainability of a software-intensive system.

In this paper, we present the application of a state-of-the-art ML approach to support effect identification. Our first results indicate that ML can be successfully used for the identification of potential sustainability effects. However, we have learned that the results from our first experiment can be improved further. This starts with the dataset. The current dataset can be improved to generate a more suitable learning model for the classifiers.

As a next step, we plan to increase the size of our dataset. We have already designed a web-based solution where experts can update the categorized requirements and add additional requirements. This web-based tool will help us to improve our labeling and support us in getting more data. It will also allow us to provide a more balanced dataset.

Our current results indicate that there is the risk of overlooking requirements which have potential effects. As a next step, we plan to focus on optimizing recall to minimize this risk. We envision that our approach could be used to complement existing methods. Instead of discussing each requirement manually, our approach could provide a list of relevant requirements, which should be discussed further by human stakeholders. High recall might result in lower precision, which means that human stakeholders are confronted with a larger number of false positives. However, we expect that providing a reduced set of requirements for discussion will enable practitioners to save time compared to a full manual analysis.

Similar to other work in this field, we have also experienced that manually identifying potential effects can lead to different opinions amongst the annotators. Although, the three annotators were able to agree on a ground truth used for training our classifier, we would like to highlight that our results might reflect a rather subjective viewpoint of the annotators.

Overall, we would like to explore how we can integrate our work into existing studies for requirements classification. In particular, we envision to use our approach within planned work on crowd-focused semi-automated requirements engineering for evolution towards sustainability [21]. This would also allow us to use our approach for other domains than smart homes, which might also result in datasets with different characteristics allowing us to further improve the classification results.

ACKNOWLEDGEMENT

The authors would like to thank Robert Ördög for implementing the tool support for our ML-based approach and for conducting the experiment presented in this paper. This research was partially funded by the European Unions Horizon 2020 research and innovation program under the Marie Skodowska-Curie grant agreement No. 674875.

REFERENCES

[1] Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. Requirements: The key to sustainability. *IEEE Software*, 33(1):56–65, Jan 2016.

[2] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. Sustainability design and software: The karlskrona manifesto. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, pages 467–476, Piscataway, NJ, USA, 2015. IEEE Press.

[3] Jeremy L Caradonna. *Sustainability: A history*. Oxford University Press, 2014.

[4] Ruzanna Chitchyan, Christoph Becker, Stefanie Betz, Leticia Duboc, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. Sustainability design in requirements engineering: State of practice. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 533–542, New York, NY, USA, 2016. ACM.

[5] Norbert Seyff, Stefanie Betz, Leticia Duboc, Colin Venters, Christoph Becker, Ruzanna Chitchyan, Birgit Penzenstadler, and Markus Nöbauer. Tailoring requirements negotiation to sustainability. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 304–314. IEEE, 2018.

[6] Leticia Duboc, Stefanie Betz, Birgit Penzenstadler, Sedef Akinli Kocak, Ruzanna Chitchyan, Ola Leifler, Jari Porras, Norbert Seyff, and Colin C Venters. Do we really know what we are building? raising awareness of potential sustainability effects of software systems in requirements engineering. In *27th IEEE International Requirements Engineering Conference*, 2019.

[7] Ahmed D Alharthi, Maria Spichkova, and Margaret Hamilton. Susoftpro: Sustainability profiling for software. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 500–501. IEEE, 2018.

[8] T. Iqbal, P. Elahidoost, and L. Lcio. A bird's eye view on requirements engineering and machine learning. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 11–20, Dec 2018.

[9] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. A survey of app store analysis for software engineering. *IEEE transactions on software engineering*, 43(9):817–847, 2016.

[10] Zijad Kurtanović and Walid Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495. IEEE, 2017.

[11] Douglas S Lange. Text classification and machine learning support for requirements analysis using blogs. In *Monterey Workshop*, pages 182–195. Springer, 2007.

[12] Grant Williams and Anas Mahmoud. Mining twitter feeds for software user requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 1–10. IEEE, 2017.

[13] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 116–125. IEEE, 2015.

[14] Sandeep Reddivari, Zhangji Chen, and Nan Niu. Recvisu: A tool for clustering-based visual exploration of requirements. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 327–328. IEEE, 2012.

[15] Hakim Sultanov and Jane Huffman Hayes. Application of reinforcement learning to requirements engineering: requirements tracing. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 52–61. IEEE, 2013.

[16] Vincenzo Gervasi and Didar Zowghi. Mining requirements links. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 196–201. Springer, 2011.

[17] Paolo Avesani, Anna Perini, Alberto Siena, and Angelo Susi. Goals at risk? machine learning at support of early assessment. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 252–255. IEEE, 2015.

[18] Moussa Amrani, Levi Lúcio, and Adrien Bibal. Ml+ fv=? a survey on the application of machine learning to formal verification. *arxiv: 1806.03600*, 2018.

[19] Pradeep K Murukannaiah, Nirav Ajmeri, and Munindar P Singh. Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd re. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 176–185. IEEE, 2016.

[20] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[21] Norbert Seyff, Stefanie Betz, Iris Groher, Melanie Stade, Ruzanna Chitchyan, Letícia Duboc, Birgit Penzenstadler, Colin Venters, and Christoph Becker. Crowd-focused semi-automated requirements engineering for evolution towards sustainability. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 370–375. IEEE, 2018.