INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.189

Automated Requirement Dependency Analysis for Complex Technical Systems

I. Gräßler, C. Oleff ✉, M. Hieb and D. Preuß

Paderborn University, Germany

✉ christian.oleff@hni.upb.de

Abstract

Requirements changes are a leading cause for project failures. Due to propagation effects, change management requires dependency analysis. Existing approaches have shortcomings regarding ability to process large requirement sets, availability of required data, differentiation of propagation behavior and consideration of higher order dependencies. This paper introduces a new method for advanced requirement dependency analysis based on machine learning. Evaluation proves applicability and high performance by means of a case example, 4 development projects and 3 workshops with industry experts.

Keywords: *requirements management, change management, systems engineering (SE), dependency analysis, artificial intelligence (AI)*

1. Introduction

Throughout the development of complex technical systems, requirements are subject to continuous change. Today, requirement changes are a key driver for project failure (The Standish Group, 2017). Each requirement change may cause additional effort. To reduce the number and negative impact of requirement changes, dependency analysis carries great potential (Gräßler *et al.*, 2020; Koh *et al.*, 2012). Requirement changes are initially caused by external triggers (for example changing customer preferences) or an increased understanding of the system, but also propagate within the requirement set. Such consecutive changes account for about 50% of all requirement changes (Giffin *et al.*, 2009). Dependency analysis not just increases system understanding, but also enables holistic change impact analysis and efficient change management (Gräßler *et al.*, 2021; Morkos, 2012).

Dependency analysis needs to be done continuously, to support change management in all development stages. It is especially challenging in early development stages, before sufficient data about the solution elements is available (Gräßler and Pottebaum, 2021; Gräßler and Pottebaum, 2022). This unlocks efficiency potentials in change management and change impact assessment (Gräßler *et al.*, 2020).

Dependency analysis requires to identify dependencies within large scale interdisciplinary requirement sets, but also to differentiate them by change propagation behaviour. Change propagation between two requirements depends on three elements: the incoming change impulse, the local requirement change behaviour (including the developer's decision on how to implement the change impulse) and the characteristics of the dependency between them (Gräßler *et al.*, 2021). Dependency analysis aims to determine such dependency characteristics and thereby enables an assessment of change propagation behaviour.

To differentiate dependencies based on their characteristics, existing reference models on dependency types can be used (Pohl, 1996; Dahlstedt and Persson, 2005; Gräßler and Hentze, 2017). Exemplary dependency types are: *requires*, *refines*, *similar* or *conflicts*. For an assessment of change propagation behaviour, relevant dependency types need to be modelled. Existing research findings indicate that the dependency types *requires*, *is required by* and *is refined by* propagate inevitably, whereas other dependency type have no or instable correlation to change propagation (Goknil *et al.*, 2014; Zhang *et al.*, 2014; Gräßler *et al.*, 2021).

Today, existing approaches mostly exist in Engineering Change Management (ECM) (Hamraz et al., 2013; Mehr et al., 2021) and Requirement Change Management (RCM) (Jayatilleke and Lai, 2018; Hein, P. H. et al., 2021) and have shortcomings regarding availability of required data in early development stages, differentiation of dependency types, higher order change propagation and ability to process large requirement sets (Gräßler et al., 2021). The contribution at hand aims to address this research gap. Therefore, the following research question is derived: *"How can requirement dependencies in developing complex technical systems be identified and characterized in early development stages and with reasonable application effort?"*

To answer the research question, alternative solution classes are compared on a theoretical level (cf. Section 3) and based on actual performance (cf. Section 4). Using the most promising approach, a method for identifying and characterizing requirement dependencies in the context of developing complex technical systems is introduced (cf. Section 5) and evaluated regarding performance, applicability and usefulness (cf. Section 6).

2. Research Approach

The research methodology is derived from the Design Research Methodology (Blessing and Chakrabarti, 2009). The work presented is part of a prescriptive study of a superordinate type five research project (Gräßler et al., 2022). In previous steps, a research project was conducted to evaluate current solutions and research gaps and to elicit performance goals of dependency analysis (Gräßler *et al.*, 2020). To address the research gaps, a literature review on solution classes beyond the domain of change management was done to identify suitable approaches for dependency analysis (for example ontologies and machine learning) (Graessler *et al.*, 2020).

Within this research, most promising approaches are implemented and compared in laboratory experiments. Based on the findings, a method for advanced requirement dependency analysis is developed and validated by a case study development project (intelligent handling robot), four student development projects (power tools) and three workshops with industry experts. Based on the case examples, experts from a leading engineering service provider of automotive industry assess performance, applicability and usefulness.

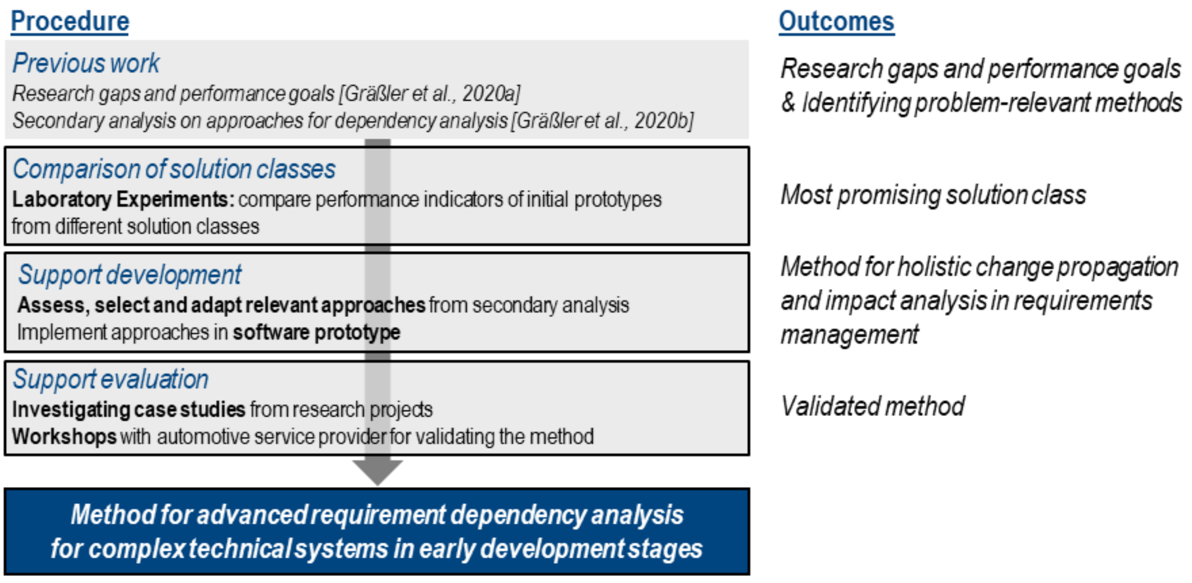
Procedure
Outcomes
Previous work
Research gaps and performance goals [Gräßler et al., 2020a]
Secondary analysis on approaches for dependency analysis [Gräßler et al., 2020b]
Research gaps and performance goals & Identifying problem-relevant methods
Comparison of solution classes
Laboratory Experiments: compare performance indicators of initial prototypes from different solution classes
Most promising solution class
Support development
Assess, select and adapt relevant approaches from secondary analysis
Implement approaches in software prototype
Method for holistic change propagation and impact analysis in requirements management
Support evaluation
Investigating case studies from research projects
Workshops with automotive service provider for validating the method
Validated method
Method for advanced requirement dependency analysis for complex technical systems in early development stages

Figure 1. Research approach

https://doi.org/10.1017/pds.2022.189 Published online by Cambridge University Press

3. Related Work

In literature, there are different types of approaches for the automated detection of requirement dependencies. These approaches are currently not applied in developing complex technical systems. These are subdivided with regard to the differentiated consideration of requirement dependencies (Graessler *et al.*, 2020). Approaches for differentiated consideration determine the type of dependency between requirements. Singular approaches determine whether a certain type of dependency exists or not. Since the number of requirements is high in the development of complex technical systems, approaches with a high degree of automation are only considered here.

Approaches for Differentiated Consideration of Dependency Types

The approaches for differentiated consideration of dependency types are distinguished between ontology-based approaches and machine learning approaches. In so called ontology-based approaches, expert knowledge is formalised in a knowledge base. Here, entities with certain dependency types are networked (Motger et al., 2019; Borrul Baraut, 2019; Zichler and Helke, 2017). Since ontology-based approaches have high creation effort and require knowledge that is not sufficiently available in early development stages, they are not considered further.
In approaches that use machine learning techniques (Deshpande *et al.*, 2019; Atas *et al.*, 2018; Samer *et al.*, 2019), classifiers are trained using supervised learning. Input is the textual description of requirements. The output is the classification of the type of dependency between requirements and is calculated automatically. In supervised learning, the output is evaluated by the expert for training. A disadvantage is that a large amount of data is needed to train the classifiers (Prabhu et al., 2021). Moreover, the early provision of such data can only be guaranteed if the development project is similar to past projects and the classifier can be trained with historical data. Development projects with high novelty might face poor quality of results due to limited transferability of a classifier trained for another development project. The effort to apply a classifier that has already been trained is low.

Approaches for Singular Consideration of Dependency Types

Furthermore, there are approaches that detect individual dependency types and do not aim for an overarching approach. Hamdaqa has developed an approach to detect external cross-references (Hamdaqa and Hamou-Lhadj, 2011). Och Dag et al. use lexical analysis in combination with stemming and stop word removal to determine the similarities between requirements (Natt och Dag *et al.*, 2002). They equate dependency with similarity. Martinez et al. also use semantic similarities to identify dependencies. To do this, they compare requirements descriptions with use cases and scenarios (Martinez *et al.*, 2019). Park et al. calculate the similarity between sets of requirements to identify possible redundancies and inconsistencies and extract the possible ambiguous requirements (Park *et al.*, 2000). The similarity measurement method combines a sliding window and a parser model. Other approaches exist to detect inconsistencies in requirement sets (Zhu and Jin, 2005; Misra, 2016). Di Thomazzo et al. develop an approach to automatically create a Requirements Traceability Matrix (RTM) based on fuzzy logic (Di Thommazo *et al.*, 2013). To determine the dependency between functional requirements, the frequency vector and cosine similarity methods are used. Abadi et al. compare the effectiveness of different information retrieval techniques for finding traceability links from code to documentation (requirements, user manuals ...) (Abadi *et al.*, 2008).

Advantages and Disadvantages of Approaches

Developing an approach for the detection of requirement dependencies is a multi-criteria optimisation problem. Criteria are *application effort*, *completeness of results* and *differentiation of dependencies* by propagation behaviour. Singular approaches are not considered further, because they cannot detect dependencies, which are necessary for analysing propagation behaviour of requirement changes: *requires*, *is required by* and *is refined by* (Gräßler *et al.*, 2021). Since neither singular nor ontology-based approaches are suitable in this context, only machine learning approaches are considered. The advantages of these approaches are the high degree of automation and robustness. Higher order change

https://doi.org/10.1017/pds.2022.189 Published online by Cambridge University Press

propagation and large requirement sets can be analysed efficiently. Disadvantage is the high amount of training data required for achieving high performance in dependency analysis.

4. Comparison of Alternative Approaches

To find the best approach for dependency analysis, a systematic comparison of machine learning approaches is conducted. Especially for BERT (Bidirectional Encoder Representations from Transformers), previous research indicates high potential regarding accuracy and ability to differentiate dependency types (Prabhu *et al.*, 2021; Arslan *et al.*, 2021). Developed by (Vaswani et al. 2017), BERT achieves good results in different application contexts (González and Garrido, 2020). BERT is based on transformers, specifically only the encoder part of the transformers, to learn the contextual relationship of words within the text corpus.

As a benchmark for comparison with alternative approaches, the well-established approaches statistical Multinomial Logistic Regression (MLR) and Support Vector Machine (SVM) as well as Recurrent Neural Network (RNN) are used. Case example data was reengineered based on the open source robotic project "BCN3D Moveo" (https://www.bcn3d.com/bcn3d-moveo-the-future-of-learning-robotic-arm/) to create a suitable dataset for comparison. The dataset includes 145 requirements which were manually enriched by a dependency matrix with 21.025 entries. According to the hold-out method which is considered suitable for validation sets with more than 1000 entries (Reich and Barai, 1999), those are split (80/20) in a training set (16820 entries) and a validation set (4205 entries) for comparison. To enable replication, random state = 42 was used.

Criteria for comparison are the *Precision*, *Recall*, *F1* and the *Receiver Operating Characteristic Area Under the Curve* (*ROC_AUC*). Precision indicates the number of predicted true positives compared to the true and false positive classes. Precision close to 1 indicates reliable models. Recall provides the proportion of correct positive classified results to the total positive result. Recall close to 1 indicates the recognition of relevant elements. The F1-Score is the harmonized average between the Precision and the Recall. The F1-Score measures the extent to which the classes are distributed in a balanced manner and thus evaluates the overall quality of the model. Lastly, the ROC_AUC provides information about the under prediction rate of a classifier and observes the relationship between true positive rate and false positive rate. This metric is used to evaluate unbalanced data. Values close to 1 indicate little under prediction whereas values close to 0.5 merely reflect chance and are less meaningful. (Alpaydın, 2019)

Since the dataset contains a major class ("None") with many entries and several minor classes with few entries (for example "Requires"), the criteria needed to be viewed macro averaged (equal weight on all classes) to be significant. Using random oversampling and class weighted as standard data augmentation approaches, BERT reaches the highest rang and outperforms the other models (cf. Table 1).

Table 1. Comparison of results for detecting requirement dependencies (random oversampling)

Criteria	**MLR**	**SVM**	**RNN**	**BERT**
Precision (macro avg.)	20.47 %	24.00 %	22.07 %	**54.10 %**
Recall (macro avg.)	21.30 %	23.43 %	60.92 %	**56.98 %**
F1 (macro avg.)	6.29 %	23.92 %	12.92 %	**55.12 %**
ROC_AUC (w. avg.)	0.64	0.74	0.81	**0.93**

The comparison shows, that Precision, Recall and F1 values of BERT are promising for a multi-class classification problem [cf. (Prabhu *et al.*, 2021)]. ROC_AUC indicates that the results are close to the optimum of 1 and that the dependency analysis is reliable. As a result, BERT is selected to be investigated in detail.

5. Automated Requirement Dependency Analysis

As shown in the previous section, BERT outperforms conventional approaches. In order to train BERT, two different types of training are needed. First, pre-training and second context specific fine-tuning.

5.1. Pre-Training and Context-Specific Fine-Tuning

Pre-training is designed to capture information by processing large amounts of textual data, such as Wikipedia texts and book corpuses (cf. section 1 in Figure 2). The pre-training task is divided into Masked Language Learning and Next Sentence Prediction. In Masked Language Learning, parts of sentences are randomly replaced by masked tokens and fed into the language model. The model's task is to classify the masked words based on probability to train word relations. Sentence prediction feeds a pair of sentences as input to the model, separated by segment embeddings. The model's task is to decide whether a sentence pair belongs together or not. The aim is to train the model to identify contextual related text data (Devlin *et al.*, 2018). Within the research at hand a generic pre-trained BERT model was used provided open source by huggingface (huggingface, 2018).
After pre-training, the model is fine-tuned on a labelled text dataset (cf. section 2 in Figure 2) in a supervised manner to learn the context specific task which is "identify requirement dependencies". Fine-tuning leaves trained hyper-parameters unchanged, but adjusts task-specific inputs and outputs in BERT end-to-end. By adding a classification layer on top of the transformer output, BERT is able to determine certain (dependency) classes (Devlin *et al.*, 2018). Within this research, the requirement data was artificially enhanced using random selection and weighted by classes for data augmentation. This is required to compensate a highly unbalanced dataset with "none" as the predominantly class and few data on the other dependency types. Within the last step (cf. section 3 in Figure 2), the BERT model is exported and can be used in a working environment.

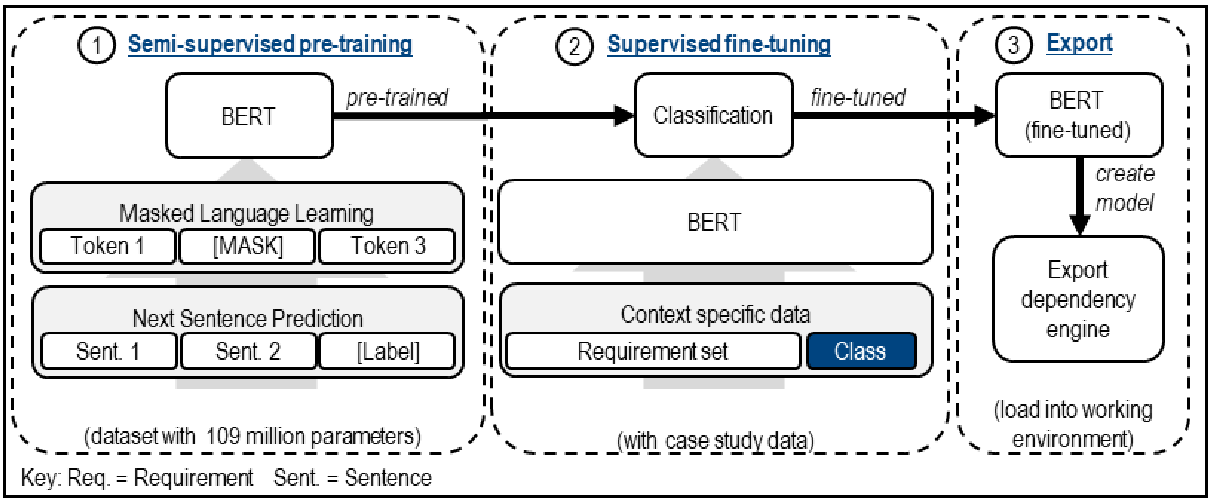

Figure 2. BERT Training

5.2. Applying BERT for Requirement Dependency Analysis

A BERT model is used for automated requirement dependency analysis. The method consists of the following steps: data pre-processing, classification of dependencies and transformation of the results into a requirement dependency list.
For **data pre-processing**, the input data (textual requirement descriptions; e.g. "max. load is 5 kg." and "max. width of object is 10 cm.") needs to be cleaned and transformed into a mathematical representation (cf. section 1 in Figure 3). This includes concatenation of input requirements, forming an ongoing text of two input requirements (text field; e.g. "max. load is 5 kg . max. width of object is 10 cm .") added by a dummy label (label field). The dummy labels will be changed later by the

https://doi.org/10.1017/pds.2022.189 Published online by Cambridge University Press

BERT model into a dependency type. After the dataset is formed, the requirement text needs to be tokenized – splitting the sentence into single words. For this research, the BERT tokenizer was used (huggingface, 2020). Tokens are differentiated by content words and stop words (e.g. "load" as content word and "of" as stop word). Stop words have grammatical purposes and support the statement of content words (Rao and McMahan, 2020). This is supported by different software packages. Within this case study the software package NLTK (Natural Language Toolkit) was used (NLTK Project, 2021). NLTK has a list of 543 German stop words that can be automatically filtered. Finally, the text data can be transformed into a mathematical representation and used as input for the BERT model.
The pre-processed input data is processed by the BERT model (cf. section 2 in Figure 3) for **classification of dependencies**. The task is to determine the estimated dependency type between two requirements. To indicate propagation behaviour, the dependency types refines, refined by, requires and required are significant (cf. Section 1) and used for classification. Classification is done fully automated, resulting in a dataset of the requirement text and their estimated dependency type.
For further usage of results, the output of the BERT model needs to be **transformed into a suitable data format** – e.g. xls or ReqIF (cf. section 3 in Figure 3). The aim is to create an interpretable representation of the requirements and their dependencies. The result is a matrix of the requirements and their classified dependency type, which can be used for impact assessment and change management.

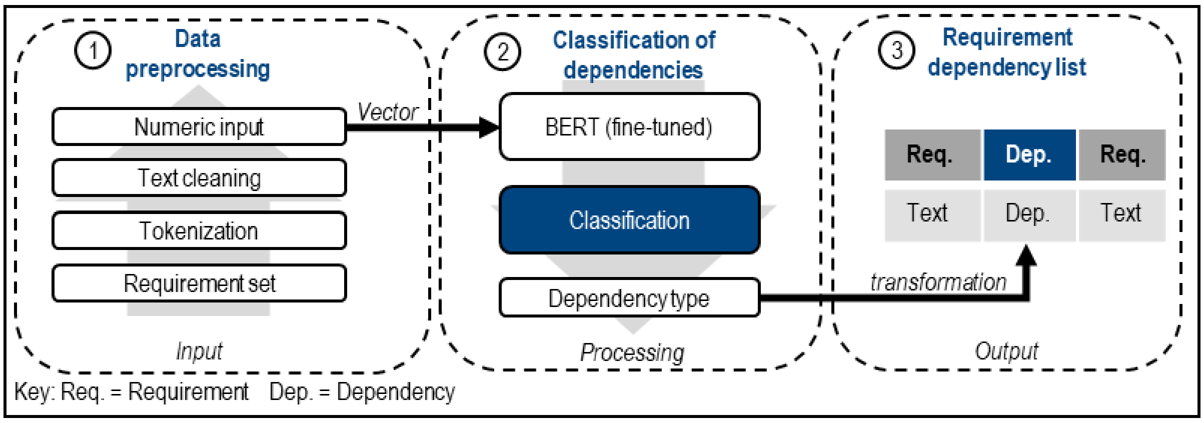

Figure 3. Applying BERT for Dependency Analysis

6. Evaluation

The method for automated requirement dependency analysis is evaluated in a four-step approach. First, requirements from a case example of developing an intelligent handling robot are analysed to determine performance of dependency analysis. Furthermore, requirements from an interdisciplinary development project of an engineering service provider are investigated. Second, the reusability is evaluated by using the initial model (trained with data on the intelligent handling robot) to classify requirements of four student development projects on power tools and determine performance. Third, the model is trained with data from the intelligent handling robot as well as the student development projects, and is also tested with data from these projects to examine the performance when using heterogeneous data for training. Fourth, applicability and usefulness of the method and software-prototype is assessed by pilot users from industry.

6.1. Case Example of Intelligent Handling Robot: Evaluating Performance

For evaluating the performance of the requirement dependency analysis, requirements from the case example of the intelligent handling robot "BCN3D Moveo" is used (cf. chapter 4). The performance indicators Precision, Recall and F1 are determined for respective dependency types for evaluation (cf. Table 2).

Table 2. Results for intelligent handling robot

Dependencies	Number	Precision	Recall	F1
None	3081	98,70 %	98,44 %	98,57 %
Refines	15	38,89 %	46,67 %	42,42 %
Refined by	15	38,46 %	33,33 %	35,71 %
Requires	31	44,44 %	38,71 %	41,38 %
Required by	31	50,00 %	67,74 %	57,53 %
Macro avg.	-	54,10 %	56,98 %	55,12 %
Weighted avg.	-	97,12 %	97,01 %	97,05 %

The results show that "none" dependencies are classified correctly with a high performance (F1: 98,57 %). The model is only conditionally suitable for distinguishing different dependency types (macro average F1: 55,12 %). Therefore, performance is investigated without differentiation of dependency types. All dependency types are assigned to "dependent" (cf. Table 3), which is legitimate since they all indicate inevitable propagation behaviour (Gräßler *et al.*, 2021).

Table 3. Results when dependency types are not distinguished

Dependencies	Number	Precision	Recall	F1
None	3081	99,12 %	99,06 %	99,09 %
Dependent	92	69,15 %	70,65 %	69,89 %
Macro avg.	-	84,14 %	84,86 %	84,49 %
Weighted avg.	-	98,25 %	98,24 %	98,24 %

The macro average F1 is 84,49 %, which shows that the performance for detecting dependencies is higher, when dependency types are not distinguished. The reason for this effect is that the data for fine-tuning BERT is more heterogenous, which reduces issues of overfitting. Due to increased performance while maintaining sufficient information on propagation behaviour, dependency types are no longer differentiated for the following investigations. This is also recommended in general, in case the training data basis is too small to reach adequate performance for multiple classes to be classified.

6.2. Student Development Projects on Power Tools: Reusability

For evaluating the reusability of fine-tuned models, a case example of four student projects on developing power tools (e.g. cordless screwdriver) is investigated. Requirements of these student projects are analysed using the BERT model which is fine-tuned with data from the intelligent handling robot (cf. section 6.1). For calculation of the Performance Indicators, results of the four projects are cumulated (cf. Table 4).

Table 4. Results for student projects

Dependencies	Number	Precision	Recall	F1
None	4291	86,79%	99,39%	92,67%
Dependent	655	18,75%	0,92%	1,75%
Macro avg.	-	52,77%	50,16%	47,21%
Weighted avg.	-	77,78%	86,35%	80,63%

"None" dependencies are classified with a high performance (F1-Score: 92,67 %) while "dependent" requirements are classified with a low performance (F1-Score: 1,75%). The results show that the performance of the model decreases, when the model is not fine-tuned using data from the investigated project (macro average-F1 of 47,21 % vs. 84,49 % cf. Table 3).

https://doi.org/10.1017/pds.2022.189 Published online by Cambridge University Press

6.3. Case Examples of Intelligent Handling Robot and Student Development Project: Performance Using Heterogeneous Data

Data from the intelligent handling robot and student development projects are used to train and test the model. This investigation shows how high the performance is when the model is trained with heterogeneous data from different contexts (cf. Table 5).

Table 5. Results for determining performance using heterogeneous data

Dependencies	Number	Precision	Recall	F1
None	2054	98,44 %	98,25 %	98,34 %
Dependent	556	93,57 %	94,24 %	93,91 %
Macro avg.	-	96,01 %	96,25 %	96,12 %
Weighted avg.	-	97,40 %	97,39 %	97,40 %

Existing dependencies are classified more accurately when trained with heterogeneous data in comparison to homogeneous data (dependent F1: 93,12 % to 69,89 %, cf. Table 3). Performance of classifying "none" is marginal lower (98,34 % to 99,09 %, cf. Table 3). To achieve high performance, data from different development projects are necessary for training.

6.4. Expert Evaluation: Applicability

Applicability is determined by the following three aspects: acceptable application effort, availability of required information, comprehensibility of results and processing time. The evaluation was done in the course of three workshops (each 120 minutes; three participants: head of department, requirements engineer and process owner). The first workshop was about the **availability of required information** and **reasonability of application effort** to create training sets for machine learning solutions. Both are seen as fulfilled. The second workshop was on the lack of comprehensibility of results and limitations of calculation time. Lack of **comprehensibility** is not seen as an issue, if an explanation of the general approach is given. **Processing time** is seen as an issue when it exceeds 5 hours. Internal testing with various amounts of requirements and with different computing power (16 GB RAM/i7 CPU: regular laptop versus 13 GB RAM/P100-GPU: high performance cloud sever *Google Colab*) was done to determine fulfilment. Testing shows that within the time limit of 5 hours, up to 100 requirements can be processed on a regular laptop (2 seconds per requirements pair) and more than 1000 requirement with a high-performance cloud sever (0,001 seconds per requirements pair). Therefore, the requirement on processing time is partly fulfilled and needs further improvement to process extensive requirement sets on regular equipment. Third workshop was on the **learning mechanism** to continuously improve the performance. Again, availability of information and suitability of application effort is evaluated as fulfilled. Therefore, the applicability of the method is fulfilled, with improvement potential regarding the processing time of extensive requirement sets.

7. Conclusion and Outlook

The results of the evaluation show that the method for automated requirement dependency analysis based on machine learning (BERT) has a high overall performance. It is possible to apply the method in different application contexts. Performance as well as applicability in different contexts depended on sufficiency of scope (performance) and heterogeneity (different application contexts) of the data base . Future work should be focused on how to limit the required training data to further increase performance and reusability. Industry workshops proof that the applicability is sufficient considering application effort, availability of information, comprehensibility of results and processing time. As a result, the method is an answer to the research question on how requirements dependency analysis can be done in early development stages of the development of complex technical systems. It enables change impact assessment and proactive change management from early development stages on. Thus, a potential to reduce the risk of project failure by insufficient handling of requirement changes can be assumed. Future work can focus on using formalized requirements for improving performance of dependency analysis, allowing further relationships between input data to be learned by the classifiers.

References

Abadi, A., Nisenson, M. and Simionovici, Y. (2008), “A Traceability Technique for Specifications”, in Krikhaar, R. (Ed.), 16th IEEE International Conference on Program Comprehension, 2008: ICPC 2008 ; 10 - 13 June 2008, Amsterdam, The Netherlands, 6/10/2008 - 6/13/2008, Amsterdam, IEEE, Piscataway, NJ, pp. 103–112.

Alpaydın, E. (2019), *Maschinelles Lernen, De Gruyter Studium*, 2. Edition, De Gruyter Oldenbourg, Berlin.

Arslan, Y., Allix, K., Veiber, L., Lothritzm Cedric, Bussyande, T., Klein, J. and Goujon, A. (2021), “A Comparison of Pre-Trained Language Models for Multi-Class Text Classification in the Financial Domain”, in Leskovec, J. (Ed.), *Companion Proceedings of the Web Conference 2021, 19-23.04.2021, Ljubljana Slovenia*, Association for Computing Machinery, New York, NJ, USA, pp. 260–268.

Atas, M., Samer, R. and Felfernig, A. (2018), “Automated Identification of Type-Specific Dependencies between Requirements”, in 2018 *IEEE/WIC/ACM International Conference on Web Intelligence (WI), 12/3/2018 - 12/6/2018, Santiago*, IEEE, Piscataway, NJ, pp. 688–695.

Blessing, L.T.M. and Chakrabarti, A. (2009), *DRM, a Design Research Methodology*, 1. Edition, Springer London, Guildford, Surrey.

Borrull Baraut, R. (2019), “Incorporation of models in automatic requirement dependencies detection”, Master Thesis, Universitat Politècnica de Catalunya, 28 January.

Dahlstedt, Å.G. and Persson, A. (2005), “Requirements Interdependencies: State of the Art and Future Challenges”, in Aurum, A. and Wohlin, C. (Eds.), *Engineering and Managing Software Requirements*, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, pp. 95–116.

Deshpande, G., Arora, C. and Ruhe, G. (2019), “Data-Driven Elicitation and Optimization of Dependencies between Requirements”, in Damian, D., Perini, A. and Lee, S.-W. (Eds.), *27th International Requirements Engineering Conference, 23-27 Nov*. 2019, *Jeju Island, Korea (South)*, IEEE, Piscataway, NJ, pp. 416–421.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018), *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, available at: arXiv:1810.04805.

Di Thommazo, A., Ribeiro, T., Olivatto, G., Werneck, V. and Fabbri, S. (2013), “An Automatic Approach to Detect Traceability Links Using Fuzzy Logic”, in *27th Brazilian Symposium*, pp. 21–30.

Giffin, M., Weck, O. de, Bounova, G., Keller, R., Eckert, C. and Clarkson, P.J. (2009), “Change Propagation Analysis in Complex Technical Systems”, *Journal of Mechanical Design*, Vol. 131 No. 8.

Goknil, A., Kurtev, I., van den Berg, K. and Spijkerman, W. (2014), “Change impact analysis for requirements: A metamodeling approach”, *Information and Software Technology*, Vol. 56 No. 8, pp. 950–972.

González-Carvajal, S. and Garrido-Merchán, E. (2020), *Comparing BERT against traditional machine learning text classification*, available at: http://arxiv.org/pdf/2005.13012v2.

Graessler, I., Preuß, D. and Oleff, C. (2020), “Automatisierte Identifikation und Charakterisierung von Anforderungsabhängigkeiten – Literaturstudie zum Vergleich von Lösungsansätzen”, in Krause, D., Paetzold, K. and Wartzack, S. (Eds.), *Proceedings of the 31st Symposium Design for X (DFX2020)*, pp. 199–208.

Gräßler, I. and Hentze, J. (2017), “Structuring and Describing Requirements in a Flexible Mesh for Development of Smart Interdisciplinary Systems”, in Araujo, A. and Mota Soares, C.A. (Eds.), *Smart Structures and Materials*, Springer International Publishing, Basel, pp. 1622–1631.

Gräßler, I., Oleff, C. and Preuß, D. (2021), “Holistic change propagation and impact analysis in requirements management”, in Wagner, B. and Wilson, J. (Eds.), *Proceeding of R&D Management Conference* 2021.

Gräßler, I., Oleff, C. and Scholle, P. (2020), “Method for Systematic Assessment of Requirement Change Risk in Industrial Practice”, *Applied Sciences*, Vol. 10 No. 23, p. 8697.

Gräßler, I., Pottebaum, J. (2021), “Generic Product Lifecycle Model: A Holistic and Adaptable Approach for Multi-Disciplinary Product–Service Systems“, *Applied Sciences*, Vol. 11 No. 10, p. 4516.

Gräßler, I., Oleff, C. and Preuß, D. (2022), “Proactive Management of Requirement Changes in the Development of Complex Technical Systems“, *Applied Sciences*, Vol. 12 No. 4; p. 1874.

Gräßler, I., Pottebaum, J. (2022), “From Agile Strategic Foresight to Sustainable Mechatronic and Cyber-Physical Systems in Circular Economies“, in Krause, D. and Heyden, E. (Eds.), *Design Methodology for Future Products*, Springer International Publishing, Cham, pp. 3–26.

Hamdaqa, M. and Hamou-Lhadj, A. (2011), “An approach based on citation analysis to support effective handling of regulatory compliance”, *Future Generation Computer Systems*, Vol. 27 No. 4, pp. 395–410.

Hamraz, B., Caldwell, N.H.M. and Clarkson, P.J. (2013), “A Holistic Categorization Framework for Literature on Engineering Change Management”, *Systems Engineering*, Vol. 16 No. 4, pp. 473–505.

huggingface (2018), “bert-base-cased”, available at: huggingface.co/bert-base-cased (accessed 4 Nov. 2021).

huggingface (2020), “BERT Tokenizer”, available at: https://huggingface.co/transformers/main_classes/tokenizer.html (accessed 4 November 2021).

Hein, P. H. Kames, E., Chen, C. and Morkos, B. (2021), “Employing machine learning techniques to assess requirement change volatility“, in *Research in Engineering Design*, 32(2021) 2, pp. 245–269.

Jayatilleke, S. and Lai, R. (2018), “A systematic review of requirements change management”, *Information and Software Technology*, Vol. 93, pp. 163–185.

Koh, Y., Caldwell, M. and Clarkson, J. (2012), “A method to assess the effects of engineering change propagation”, *Research in Engineering Design*, Vol. 23 No. 4, pp. 329–351.

Martinez, G.G., Carpio, A.F.D. and Gomez, L.N. (2019), “A Model for Detecting Conflicts and Dependencies in Non-Functional Requirements Using Scenarios and Use Cases”, in *XLV Latin American Computing Conference (CLEI), Piscataway, NJ*, IEEE, pp. 1–8.

Mehr, M. R., Rashed, S. A. M., Lueder, A. and Mißler-Behr, M. (2021), “An Approach to Capture, Evaluate and Handle Complexity of Engineering Change Occurrences in New Product Development“, in *International Journal of Industrial and Manufacturing Engineering*, 15(2021) 9; pp. 400–408.

Misra, J. (2016), “Terminological inconsistency analysis of natural language requirements”, *Information and Software Technology*, Vol. 74, pp. 183–193.

Morkos, B. (2012), “Computational representation and reasoning support for requirements change management in complex system design”, Dissertation, Clemson University, 2012.

Motger, Q., Borrull, R., Palomares, C. and Marco, J. (2019), *OpenReq-DD: A requirements dependency detection tool, REFSQ Workshops*, available at: ceur-ws.org/Vol-2376/NLP4RE19_paper01 (accessed 4 Nov. 2021).

Natt och Dag, J., Regnell, B., Carlshamre, P., Andersson, M. and Karlsson, J. (2002), “A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development”, *Requirements Engineering*, Vol. 7 No. 1, pp. 20–33.

NLTK Project (2021), “Natural Language Toolkit”, available at: https://www.nltk.org/ (accessed 4 Nov. 2021).

Park, S., Kim, H., Ko, Y. and Seo, J. (2000), “Implementation of an efficient requirements-analysis supporting system using similarity measure techniques”, *Information and Software Technology*, Vol. 42, pp. 429–438.

Pohl, K. (1996), *Process-centered requirements engineering, Advanced software development series*, Vol. 5, Wiley; Research Studies Press, New York, NY, Taunton, Somerset, England.

Prabhu, S., Mohamed, M. and Misra, H. (2021), *Multi-class Text Classification using BERT-based Active Learning*, available at: http://arxiv.org/pdf/2104.14289v2.

Rao, D. and McMahan, B. (2020), Natural Language Processing mit PyTorch: *In*telligente Sprachanwendungen mit Deep Learning erstellen, 1. Edition, O'Reilly Verlag, Heidelberg.

Reich, Y. and Barai, S.V. (1999), "Evaluating machine learning models for engineering problems", *Artificial Intelligence in Engineering*, 13 (1999) 3, pp. 257-272.

Samer, R., Stettinger, M., Atas, M., Felfernig, A., Ruhe, G. and Deshpande, G. (2019), “New Approaches to the Identification of Dependencies between Requirements”, in *31st International Conference on Tools with Artificial Intelligence*, *4-6 Nov.* 2019, *Portland, OR, USA*, IEEE, Piscataway, NJ, USA, pp. 1265–1270.

The Standish Group (2017), *Chaos Manifesto* 2018, West Yarmouth, USA.

Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q. and Li, M. (2014), “Investigating dependencies in software requirements for change propagation analysis”, *Information and Software Technology*, Vol. 56, pp. 40–53.

Zhu, X. and Jin, Z. (2005), “Inconsistency measurement of software requirements specifications: an ontology-based approach”, paper presented at 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), 16-20 June 2005, Shanghai, China.

Zichler, K. and Helke, S. (2017), “Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft”, in Dencker, P., Klenk, H., Keller, H.B. and Plödereder, E. (Eds.), *Automotive - Safety & Security 2017: Sicherheit und Zuverlässigkeit für automobile Informationstechnik 30.-31. Mai 2017 Stuttgart, Germany, GI-Edition - lecture notes in informatics (LNI) Proceedings*, Gesellschaft für Informatik e.V. (GI), Bonn, pp. 121–134.