Data-driven Elicitation and Optimization of Dependencies between Requirements

Gouri Deshpande, Chahal Arora, Guenther Ruhe
University of Calgary, Calgary, Canada
Email: {gouri.deshpande, chahal.arora1, ruhe}@ucalgary.ca

***Abstract*—Requirement dependencies affect many activities in the software development life cycle such as design, implementation, testing, release planning and change management. They are the basis for various software development decisions. However, requirements dependencies extraction is not only error-prone but also a cognitively and computationally complex problem that consumes substantial efforts, since most of the requirements are documented in natural language. This paper proposes a novel approach to extracts requirements dependencies utilizing natural-language processing (NLP) and weakly supervised learning (WSL) in two stages. In the first stage, binary dependencies (basic dependencies:dependent/independent) are identified, which are further analyzed to detect the type of the dependency in the second stage. An initial evaluation of this approach on the PURE data set - European Rail Traffic Management System - was carried out using three machine learners (Random Forest, Support Vector Machine and Naïve Bayes), which were then compared and tested. Results showed that all the three learners exhibited similar accuracy measures, while SVM needed additional parameter tuning. The machine learners' accuracy was further improved by applying weakly supervised learning to generate pseudo annotations for unlabelled data. Based on these results, agenda is to provide decision support under a dynamic use case scenario that includes (i) continuous updates and analysis of dependencies, (ii) identification of the general types of dependencies, and (iii) dependencies as a key driver of the decision support for the product releases.**

***Keywords*-Dependencies between requirements; Data analytics, Natural language processing, Machine learning, Weakly supervised learning, Advanced dependencies, Optimization, Release planning**

I. Introduction

In an industry case study, in the telco domain, Carlshamre [5] showed that a large percentage of requirements can have one or more dependent relationships. Also, recently, in automotive systems, Vogelsang et al. [28] found that at least 85% of the analyzed vehicle features depend on each other. These findings emphasize how requirements have to be designed or implemented, how they influence the cost and value of other requirements and how they increase and decrease the overall implementation efforts when considered in conjunction.

Many studies in the past have recognized the difficulty associated with requirements dependency extraction in software engineering [9],[5],[7],[24],[23],[31]. Besides their extraction, one fundamental problem related to requirement dependencies is that, similar to the requirements themselves, dependencies evolve and change over time. Maintaining and tracking these changes is equally important. If critical dependencies are missed then this would likely result in reworking in the design, development and testing of the software. Furthermore, ignoring dependencies would reduce the value (for the user) of product releases. In the past, various techniques including Natural Language Processing (NLP) [6],[24], Fuzzy logic [23], Predicate logic [29] was utilized to extract dependencies. Recent study [13], has applied supervised machine learning methods on requirements specific artefacts. The focus of this study has been mainly on *pair-wise* dependencies from traceability perspective with no emphasis on the types of dependency and various degrees of dependencies. Since some dependencies might be important while others optional or good to have in the product, dependencies need not be limited to include just two requirements. Additionally, dependencies can also result from value and effort synergies when considered in conjunction. Hence, it is necessary to focus on these aspects of research and dependencies across multiple requirements.

While dependency extraction is important, release planning refers to the problem of selecting requirements based on the dependencies. A company has to consider and balance the trade-off between all these factors during release planning where the emphasis is to maximize the customer satisfaction and value synergies, minimize the effort synergies while choosing the most dependent requirements for any given release. Although there have been studies to extract value and cost based dependencies [12],[11], there are no studies which consider them as synergies while selecting the requirements. If requirement dependencies are not tracked and maintained in the life cycle of the software development then, it would not be possible to facilitate dependency centred release plans, which, as an end result, would eventually maximize the value of the release and reduce the penalty when the dependencies are missed.

In this paper, as a first step towards broadening the scope of dependency analysis and maintenance, we propose (i) a systematic approach for extraction of mutual requirements dependencies and its initial evaluation utilizing weakly supervised learning, (ii) extend this approach to cover more general types of dependencies, and (iii) model dependency management as a multi-objective optimization problem, which balances value and effort synergies, penalties (if any) from violating dependency constraints for release decisions.

The remainder of the paper is structured as follows: Section II elaborate related work. Background and methodology are explained in Sections III and IV respectively. Section V

explains the results. Section VI provides details on the threats to validity. Finally, Section VII presents a plan for future research.

II. Related work

A. Literature analysis

J.N.och Dag et al. [24] analysed requirements textual content and focused on the dependency identification based on similarity measures, such as Dice, Jaccard, and Cosine coefficients. However, he focused only on the "similar" dependency type and mentioned linguistic methodology and domain-specific vocabulary use as future work. Chichyan et al. [7] discussed how the semantics of unstructured requirements can be deduced to find the dependency types using Natural Language Processing (NLP) techniques. Ngo-The et al. [23] used fuzzy logic to model the uncertainty concerning the identification of structural dependency constraints between requirements. Weston et al. [29] applied predicate logic to detect conflicting requirements.

Goknil et al. [12],[11] explored requirement dependencies in greater depth from a traceability perspective for change impact analysis. This author used formal semantics of relation types to infer new relations and determines contradicting relations in the requirements documents. This research utilized semantic web technologies to identify conflicts-with, refines and contains relations. Although a tool was developed as part of this work, which provided various modules to identify, maintain and visualize the dependency network as a graph, the approach is theoretical in nature and only applied on a toy data set lacking industrial and empirical study.

Semantic cohesiveness between features was considered and studied for the theme-based release planning [16]. For mobile apps, optimized super app functionality was proposed by Nayebi et al. [22]. However, these studies do not address dependency centred release planning over a period of software development life cycle and software evolution.

Researchers have also explored requirement dependencies from traceability perspective and utilized NLP and machine learning to a great extent on requirements artefacts [13]. However, to our knowledge, this study is limited to extract the trace and not the structural type of dependency among the requirements, which is a focus of our research. We base our research on the fact that requirement dependency [4] specific research focuses on relationships and their types between a specific type of trace object – namely, explicitly stated requirements.

B. Survey of Practitioners

In order to identify and focus on the difficulties faced by software industry professionals with respect to requirements dependencies extraction and maintenance, we conducted a survey[1] which attracted seventy responses. Of these participants, 24% were managers, 52% were developers, analysts or testers, while the rest were students and others. As an affirmation, more than 80% of the participants agreed or strongly agreed that identifying the strength of dependencies is important, dependency type extraction is difficult, dependency information has implications on maintenance and ignoring dependencies has a significant impact on project success [8].

[1]https://goo.gl/forms/TBie370fZ2kzoD443

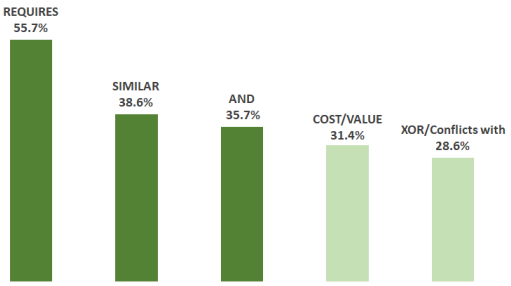

Fig. 1. Ranking of occurrence by types of dependency

Interestingly, the survey also revealed that more than 50% of the participants agreed that dependencies across multiple features are found frequently. More than 90% of the participants also confirmed that they do not use any automation tool to extract and maintain the dependencies. The respondents were asked to rank dependency types based on the frequency of their occurrence. As shown in Figure 1, $REQUIRES$ and $SIMILAR$ types are found to occur most frequently.

III. Background

Maalej et al. [17] projected a paradigm shift in requirements engineering and software evolution towards data-driven processes. In our research, we follow this shift to study data-driven techniques to extract basic (binary), and advanced dependencies (types of dependencies) between requirements. We also outline the use of the dependencies for the optimization of the release decisions. In this section, we provide details on terminologies and concepts used in this paper.

A. Basic Dependencies (Binary: dependent or independent)

For a set of requirements R, and any pair of requirements (r, s) ϵ R, the symmetric relationship is called a *basic dependency*, if there is at least one type of dependency ($REQUIRES$, $SIMILAR$, OR, AND, XOR, value synergy, effort synergy) between r and s (independent of type, direction, and strength).

B. Advanced Dependencies

Assuming, we can extract basic dependencies, the second research question is to extract the type of the dependency. Details of these types are as following.

Definition: For a set of requirements R and any pair of requirements (r, s) ϵ R, if r requires s, or s requires r, then, r and s are in a relationship called $REQUIRES$.

Definition: For a set of requirements R and any pair of requirements (r, s) ϵ R, if r and s are required in conjunction, then, r and s are in a relationship called AND.

Definition: For a set of requirements R and any pair of requirements (r, s) ϵ R, if r and s are semantically similar, then, r and s are in a relationship called $SIMILAR$.

Definition: Violation of structural dependencies (like $REQUIRE$, AND etc.) is supposed to create a *penalty*. The degree of penalty depends on the impact of the violation to the user. We define the *penalty()* function on a nine-points scale (0-9 : low to high).

Definition: A set $RV \subset R$ of requirements creates a value (or effort) synergy dependency *values()* (respectively *efforts()*) if offering the requirements in the same release increases the value (respectively reduces the effort of implementation) when compared to the sum of the value (or effort) of the individual requirements.

C. Weakly supervised learning

Weakly supervised learning (WSL) [32] combines the benefits of supervised learning and unsupervised learning. It is motivated by the high cost of data-labeling. There are various strategies for using unlabeled data to improve the performance of standard supervised learning algorithms, especially in the situation where a small amount of labelled data is available, which is insufficient to train a good learner, while abundant unlabeled data are available. WSL is an umbrella term covering a variety of techniques, which attempt to construct predictive models by learning with weak supervision. This approach is a form of *conservative co-testing* strategy [21] where, for each iteration, an unlabeled example is labelled if the two classifiers agree on the labeling [26].

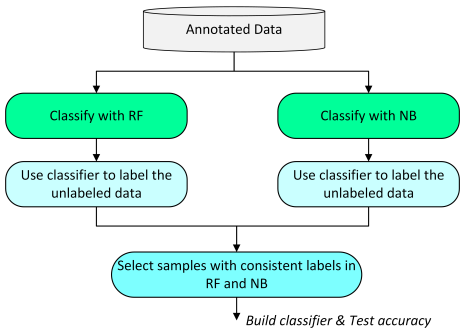

Fig. 2. Main steps of our WSL based approach (NB: Naive Bayes, RF: Random Forest)

As shown in Figure 2, firstly, machine learning models (NB:Naive Bayes and RF:Random Forest) are trained on the *original data set*. Further, these classifiers are used to classify the unlabeled data samples. The strategy is to extract just the data samples on which all the classifiers achieved consistent results. If all classifiers have predicted the same class label for a given data point (sample), then this label is assigned as a pseudo annotation.

IV. Methodology

In this section, we first introduce the research questions. Thereafter, we describe the research method along with the data preparation, classifiers and evaluation phases.

A. Research questions

The approach is to utilize NLP, supervised as well as weakly supervised learning algorithms to automate the extraction of advanced requirements dependencies. The proposed research is organized around the three research questions (RQs).

RQ1 How accurate are supervised machine learning methods RF, SVM, and NB and WSL at extracting pairs of basic dependencies?

RQ2 How accurate are supervised and WSL at extracting advanced requirements dependencies?

RQ3 How effective and how efficient is the usage of interactive swarm intelligence for determining the functionality of the upcoming software release?

Each RQ's output is input to the next RQ. Hence, RQ3 utilizes fine-grained dependencies information from RQ2 to perform release optimization, which has objectives specific to dependencies between requirements. We plan to utilize *swarm intelligence*, a collection of *bio-inspired* optimization algorithms, for this RQ. These algorithms have been proven successful in a large number of application engineering, image processing and data mining [30] providing an option for performing optimization in a *interactive mode* with the release decision-maker. The logical structure of our approach is shown in Figure 3. The following describes the essential steps of the method and how it was applied to our data.

B. Data Preparation

Firstly (Step ❶), raw data from the textual requirements information (or document) is processed to extract the requirement statements. Following this extraction, the manual annotation process (step ❷) must be carried out. To proceed with text classification (generating classifiers), the data set is passed through a NLP pipeline. Thereby, the data is first tokenized, eliminating possible stop words (English dictionary based), and then lemmatized using standard snowball Stemmer and WordNet Lemmatizer [18] (step ❸, ❹, ❺).

C. Classifiers

To solve RQ1 and RQ2, we utilized three classifiers: Random Forest (RF), Naive Bayes (NB) and Support Vector Machine (SVM) (step ❻). Multi-class annotations were also utilized to further develop multi-class classifiers (step ❼). While NB algorithm searches for the best linear separator according to some criterion, SVM and RF have been used successfully and prominently for text classification [18]. We utilized Python's scikit-learn library [3] for implementation.

While RF and NB classifiers performed well with basic settings, SVM needed additional tuning on the hyperparameters. Considering the adaptive capacity of SVM, Radial Basis Function (RBF) was selected as the kernel function, penalty parameter C=2.0 and kernel parameter gamma =2.0 to achieve better classification accuracy [14].

D. Evaluation

A good validation technique should not overestimate or underestimate the model performance on unlabelled data. Since the annotated data set was small (a balanced 300 data samples, where each data sample represents a pair of requirements), we utilized a more robust sampling technique called k times k-fold

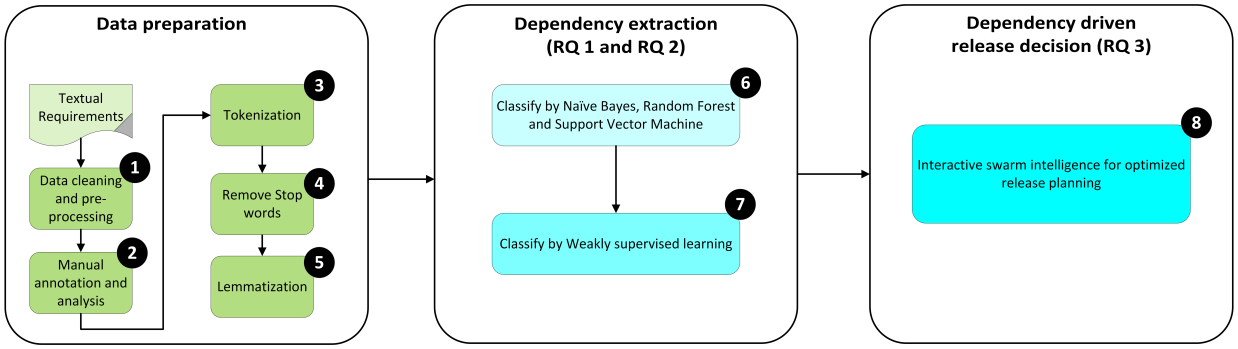

Fig. 3. Structure of our research approach

cross-validation (CV) technique to eliminate possible bias in the classifiers. This validation technique has been proven as the most unbiased validation technique, which can be effectively utilized in the event of smaller training data sets to avoid overfitting [27],[20],[14]. We utilized 10 times 10-fold CV technique for this study.

V. Results

A. Data preparation

For this study, the sample data for evaluation were taken from the public data set: PURE [10]. The Ertms/ETCS [1] functional requirements specification document (SRS) consisted of 199 requirements with average sentence length of 10 words. With these $n = 199$ requirements, we ended up with $n(n-1)/2 = 19,701$ potential pairwise dependencies. In order to generate the ground truth data, we manually annotated the 550 data samples since this data belonged to a public PURE library [10]. Two of the authors that were engaged in this activity independently first studied the project utilizing the description in the SRS document. For more effective manual annotation, the pair-wise cosine similarity was utilized as a starting point for the manual annotation. The final data set consisted of the samples on which both the annotators agreed upon the label. The term *original data set* is used to describe this balanced data set of 300 samples.

In order to ready the data for classification, it was first tokenized then all the English stop words were removed and each token was then lemmatized to perform classification. Further, the pairs with higher cosine similarity were picked first to check about possible dependencies. Finally, a combination of higher and lower cosine similarity pairs was randomly chosen to generate 550 annotations. Out of them, 367 were annotated as independent, 150 as dependent, 33 could not be classified. For multi-class annotations, 38 were of $REQUIRE$ type, 89 $SIMILAR$, 19 AND, 3 OR , and 1 XOR. At this stage, data was now ready for classification.

B. Stage 1: Basic dependency extraction

For the Binary (basic dependency) classifier, the initial data set (after annotation) had a class imbalance. We extracted all 150 data samples (with class = 1, dependent) and randomly extracted 150 independent data samples from the pool of 367 independent samples. Following the methodology explained in Section IV, we generated classifiers with NB, RF and SVM.

Averaged (from 10 times 10-fold CV) results for the classifiers are shown in Table I (a). Of the three, SVM has the highest precision and F1 score. It implies that of all the classifiers SVM could classify the valid dependent pair of requirements most accurately. However, all three classifiers have lower recall, and higher precision rate, which means that they lack the ability to identify the possible dependent pairs correctly. Additionally, all three classifiers generated approximately 0.7 F1 accuracies. We concluded that all of the present classifiers could be utilized as an efficient prediction model. However, to improve accuracy, we analyzed the effect of our proposed WSL mechanism.

TABLE I
RQ1: Average classifier accuracy from 10 times 10-fold CV before (a) and after (b) utilizing pseudo-labelled data using WSL (see Figure 2). Class 0: Independent, 1: Dependent

	Classifier	Precision	Recall	F1 Score
a) Original data set	RF	0.79	0.60	0.67
(#Class 0 = 150)	NB	0.77	0.70	0.73
#(Class 1 = 150)	SVM	**0.85**	0.68	0.74
b) Original data set + WSL sample	RF	0.90	0.81	0.85
(#Class 0 = 300)	NB	0.89	0.85	0.87
(#Class 1 = 300)	SVM	**0.94**	0.85	0.89

C. Pseudo labeling using Weakly supervised learning

In order to (pseudo) label the rest of the 19,000 requirement pairs, RF and NB classifiers created from the original data set were utilized (Figure 2). Since running classifier models have a non-deterministic characteristic, multiple instances for each of them were created. Only the samples (11,141) with consistent classification were finally selected. This new annotation resulted in *pseudo labelled samples*. The updated *original data set*, called as *updated data set*, was then tested using 10 times 10-fold CV through classification.

Results for various data sample sizes are shown in the Table I (b). From the results, we concluded that WSL classifiers

TABLE II
RQ2: AVERAGE CLASSIFIER ACCURACY FROM 10 TIMES 10-FOLD CV BEFORE (A) AND AFTER (B) UTILIZING PSEUDO-LABELLED DATA USING WSL. CLASS 1: $REQUIRES$, 2: $SIMILAR$, 3: OTHERS

	Classifier	**Precision**	**Recall**	**F1 Score**
a) Original data	RF	0.67	0.65	0.61
(#Class 1,2,3 = 38)	NB	0.65	0.60	0.59
	SVM	0.76	0.69	0.69
b) Original & pseudo-labeled data	RF	0.85	0.81	0.81
(#Class 1,2,3 = 76)	NB	0.83	0.81	0.81
	SVM	0.88	0.87	0.87

performed better compared to those original learners. The average CV showed more than 10%+ better performance utilizing just 300 pseudo labelled data samples. Additionally, all three classifiers demonstrated exceptional improvement while SVM performed well overall.

D. Stage 2: Extraction of the dependency type

To address RQ2, we utilized 38 data samples each for the three classes (114 in total). For the purpose of demonstrating our findings, we annotated a multi-class subset of *original data set* is referred to as *baseline multi-class data set* in this paper. Utilizing this data set, baseline accuracies for RF, NB and SVM classifiers were generated. Results are shown in Table II (a).

We chose to report the weighted average results for the multi-class classifiers, because when binary classification metric is extended to multi-class problems, the data is treated as a collection of binary problems, one for each class. There are then a number of ways to average binary metric calculations across the set of classes, each of which may be useful in some scenario. Where available, it is recommended to select the weighted average parameter [2].

Analysis of the statistics revealed that the accuracy of all the classifiers remained in the same range (approx 0.6). However, SVM and RF performed well whereas NB demonstrated poor recall and 10-fold CV score. The poor performance of the classifiers could be attributed to a limited and small data set used for classification.

The results of utilizing WSL on the *baseline Multi-class data set* are documented in Table II (b). The balanced data set for three different classes were extracted randomly for classification from a pool of 4,460 data samples from WSL steps (Figure 3) on 19,000 data samples. This three class balanced data set consisted of 228 data samples (=76*3). Statistical results show that NB and RF performed comparatively well but the performance of SVM was exceptional. Although this improvement in the accuracy is welcoming and supportive of WSL, closer evaluation is required in the future to affirm the improvement in the performance.

VI. THREATS TO VALIDITY

The results are preliminary as we studied the approach just for one data set. We will perform a more comprehensive analysis with data from PURE in the future. In addition, we have attracted industrial data with access to domain experts to further check the validity of the approach.

For RQ1, the size of 500 samples is relatively small when compared to the size of the overall sample set. We proposed WSL to overcome this deficit. For the actual annotations, each sample was annotated twice. In case of inconsistency, samples were not considered. Annotating test data from applying WSL is considered one way to overcome the annotation bottleneck [32]. For that, we only considered samples with consistent classification from all the classification runs.

For RQ2, the annotated data is small and the results might be biased. Also, this might have an impact on the WSL created output. We plan to evaluate our approach with additional annotations, which are specific to a few selected classes.

k-fold CV is one way to validate results. There is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Given these considerations, one performs k-fold cross-validation with k=5 or k=10, as these values have been shown empirically to yield test error rate estimate [15]. We applied k=10. In addition, we applied it multiple times to overcome the impact of the fold selection. However, we need to test our approach with domain experts to see how valid the results are.

The quality of the textual content describing requirements is one of the key performance factors of our approach. Also, the overall use of this approach in different domains needs thorough evaluation, which is part of the future research agenda. To increase the semantic foundations, we envision to address this problem by utilizing evolving ontologies.

VII. FUTURE RESEARCH AGENDA

In this paper, we utilized a WSL based labeling approach, a technique which exploits unlabeled data without querying human experts. In the future, we will extend this approach to utilize active learning, which is based on the approach of utilizing minimal labeling effort by oracles (eg: Human expert), such that the labeling cost of training good model is minimized. This is a robust approach, which is a combination of active learning and co-testing. In particular, we will let oracle label unlabeled examples on which the disagreement within the multiple classifiers is the greatest.

Besides the evaluation of more documents taken from the public PURE library [10], we are working on the data from the industry. Our industry collaborator has an interesting setup of where the product has both software, as well as hardware components and additional dependencies that stem from firmware. This company also participated in our survey (Section II.B) and has already provided first data set.

As part of the extended research agenda, we will evaluate a hybrid approach which is a combination of NLP's information extraction and WSL mechanisms. Also, the agenda is to

extend this approach to identify dependencies among multiple requirements and not just pair-wise requirements.

A. Dependency aware release decisions

Release planning is a wicked problem [25], i.e. the formulation of the problem is cognitively difficult and there is no easy method to decide on a single solution. We approach the wickedness of the release problem by applying an evolutionary modelling and interactive problem-solving mechanism. This includes interaction with the user and decision-maker. Interactive optimization approaches acknowledge existing limitations of modeling and parameter settings. Also, they value the user's expertise in the application domain [19].

Overcoming the simplistic notion of just maximizing a value function defined from isolated requirements, the new problem formulation examines requirements dependencies and makes them the driver of decisions. The novelty of this optimization approach is, to bundle highly dependent requirements and make them the core content of releases. Therefore, the optimization is multi-objective with three optimizing functions:

$$Minimize\ F1\ = \sum_{All\ structural\ dep's\ n} penalty(n)$$

$$Maximize\ F2 = \sum_{All\ value\ synergies\ n} values(n)$$

$$Minimize\ F3 = \sum_{All\ efforts\ synergies\ n} effort(n)$$

F1 is defined on the product set $R \times R$, which results in a quadratic function. The other two objectives F2 and F3 are defined on the power set of R. These functions are very difficult to formulate, and applying swarm intelligence is expected to handle both non-linearity and non-convexity of the set-defined synergy functions: *value* and *effort* respectively. The functions describe the added value and the reduced effort when compared with the case of an isolated selection of requirements.

REFERENCES

[1] European rail traffic management system. https://en.wikipedia.org/wiki/European_Rail_Traffic_Management_System. Accessed: 29 Jan 2019.

[2] Scikit-learn model evalutation. https://scikit-learn.org/stable/modules/modelevaluation.html#precision-recall-f-measure-metrics. Accessed: 15 Apr 2019.

[3] Scikit-learn supervised learning. https://scikit-learn.org/stable/supervisedlearning.html#supervised-learning. Accessed: 29 Jan 2019.

[4] A. Aurum and C. Wohlin, editors. *Engineering and Managing Software Requirements*. Springer-Verlag, 2005.

[5] P. Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 7(3):139–151, Sept. 2002.

[6] R. Chitchyan and A. Rashid. Tracing requirements interdependency semantics. In *Early Aspects*, 2006.

[7] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *Proceedings of the 6th International Conference on Aspect-oriented Software Development*, pages 36–48. ACM, 2007.

[8] G. Deshpande and G. Ruhe. Elicitation and maintenance of requirements dependencies: A state-of-the practice survey. https://community.ispma.org/elicitation-and-maintenance-of-requirements-dependencies-a-state-of-the-practice-survey. Accessed: 12 Dec 2018.

[9] D. M. Fernández and S. Wagner. Naming the pain in requirements engineering: A design for a global family of surveys and first results from germany. *Information and Software Technology*, 57:616–643, 2015.

[10] A. Ferrari, G. O. Spagnolo, and S. Gnesi. Pure: A dataset of public requirements documents. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 502–505. IEEE, 2017.

[11] A. Goknil, I. Kurtev, K. Van Den Berg, and W. Spijkerman. Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, 56(8):950–972, 2014.

[12] A. Goknil, I. Kurtev, K. van den Berg, and J.-W. Veldhuis. Semantics of trace relations in requirements models for consistency checking and inferencing. *Software & Systems Modeling*, 10(1):31–54, 2011.

[13] J. Guo, J. Cheng, and J. Cleland-Huang. Semantically enhanced software traceability using deep learning techniques. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 3–14. IEEE, 2017.

[14] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

[15] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to Statistical Learning*, volume 112. Springer, 2013.

[16] M. R. Karim and G. Ruhe. Bi-objective genetic search for release planning in support of themes. In *Proc. SSBSE*, pages 123–137. Springer, 2014.

[17] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

[18] C. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[19] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud. A review and taxonomy of interactive optimization methods in operations research. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(3):17, 2015.

[20] T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering*, 17(1-2):1–17, Jan. 2012.

[21] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27:203–233, 2006.

[22] M. Nayebi and G. Ruhe. Optimized functionality for super mobile apps. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 388–393. IEEE, 2017.

[23] A. Ngo-The and M. O. Saliu. Fuzzy structural dependency constraints in software release planning. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 442–447. IEEE, 2005.

[24] J. N. och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[25] H. W. Rittel and M. M. Webber. Wicked problems. *Man-made Futures*, 26(1):272–280, 1974.

[26] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, Feb. 2013.

[27] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. An empirical comparison of model validation techniques for defect prediction models. *IEEE TSE*, 43(1):1–18, 2017.

[28] A. Vogelsang and S. Fuhrmann. Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 267–272. IEEE, 2013.

[29] N. Weston, R. Chitchyan, and A. Rashid. Formal semantic conflict detection in aspect-oriented requirements. *Requirements Engineering*, 14(4):247, 2009.

[30] X.-S. Yang, S. Deb, Y.-X. Zhao, S. Fong, and X. He. Swarm intelligence: past, present and future. *Soft Computing*, 22(18):5923–5933, 2018.

[31] W. Zhang, H. Mei, and H. Zhao. A feature-oriented approach to modeling requirements dependencies. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 273–282. IEEE, 2005.

[32] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.