Machine Learning in Requirements Engineering

An Abstract Process for Validating Requirements using Machine Learning

Orcun Talha Vardarcik
Johannes Kepler University
Linz, Austria
orcunvardarcik@gmail.com

Sebastian Kloibhofer
Johannes Kepler University
Linz, Austria
sebastian.kloibhofer@gmail.com

***Abstract*—Requirements engineering is a key part of the software engineering process. Any shortcomings or problems during this phase may cause project failures or cost spikes later on. Thus, it is essential to include all stakeholders for the elicitation process as well as detect and specify all key requirements of the project. Research has shown that machine learning techniques are applicable in this area to optimize some of the workflows and reduce the amount of data that has to be processed. In our work we give an overview of the current state of machine learning applications throughout the requirements engineering phase and describe a pipeline-based semantic model for incorporating machine learning into different steps of the process. In the end, we summarize different methods on how to evaluate requirements using classification techniques and suggest a simple scorer based on the found literature.**

***Index Terms*—Requirements engineering, machine learning, supervised learning, unsupervised learning**

I. Introduction

Requirements Engineering (RE) forms the initial step in the software engineering process. It includes the selection (or determination) of the various stakeholders as well as gathering and prioritizing their individual requirements. As those may overlap or even contradict each other, the process also involves negotiation with and between the stakeholders to agree upon mutually satisfying requirements which form the basis of the actual project. Ideally, the process culminates in a software requirements specification (SRS) which describes all major project requirements in addition to use cases, constraints and metrics about importance and severity of the individual requirements [1], [2].

While the requirements engineering process is built on a solid theoretical foundation, the practical application still requires experience to provide good coverage and correctly classify the different requirements. Studies furthermore confirm, that problems in requirements engineering are often related to lack of expertise in the area and underestimation of the actually required effort [3], [4]. Additionally, the amount of data to process or the variety of stakeholders may represent another challenge depending on the project.

An idea that has come up frequently in recent years is to apply machine learning (ML) techniques to simplify or generalize those steps in the machine learning process. ML algorithms and methodologies should simplify frequently occurring tasks within requirements engineering such as classification of data (requirements), extracting or generating features from them (i.e. quality metrics, priorities) or process semi- / unstructured data such as user / stakeholder requests and requirements documents. As there exists a wide selection of research on this topic, we aim to summarize the different findings and proposal to showcase different usage scenarios of ML throughout the requirements engineering process. Furthermore, we categorize the different methodologies in an abstract framework which describes a generic way of integrating various ML algorithms and techniques into arbitrary steps of the process.

The following sections first describe the existing literature in this field (section II) and give a brief overview of our research methodology. Then, in section III we define our custom taxonomy that classifies the found literature. In the core part in section IV, we finally go into detail about the proposed framework and characterize the different components.

II. Related Work

The research in this area has already been very comprehensive and shown impressive results for different projects [5], [6]. We gathered our materials via an extensive literature search that covered both aspects of software (requirements) engineering as well as individual machine learning or - more general - artificial intelligence techniques.

One major contribution in this selection stems from Iqbal et al. [7] which already provides a comprehensive summary of key techniques in integrating machine learning into requirements engineering. In contrast to this work, we do not aim to completely cover all aspects of the requirements engineering process with current machine learning applications but strive to generate an abstract model that showcases the different use cases in the process while also describing the links in between.

Regarding such abstract models, Castro-Herrera et al. suggest similar ideas for one particular step of the process, namely *requirements elicitation* and *stakeholder management* [8]. Similarly, Avesani et al. describe a formal framework for the *requirements prioritization* step that relies on machine learning techniques [9]. Lastly, we also want to mention Li et al. and their work about “Automatically classifying user requests in crowdsourcing requirements engineering” [10] where they give a detailed overview of a comprehensive model for

handling unstructured (requirements) data and classifying them accordingly.

III. Designing a Taxonomy for Requirements Engineering and Machine Learning

The first goal of our research was to create a common taxonomy of the existing research work. Structuring the material was based on identified document tags and keywords. This led to an initial classification of items into the different requirements engineering phases. Additionally, we tried to create orthogonal categories based on the machine learning concepts that were used in the research material. In the end, this resulted in three classification axes:

- Requirements engineering perspective
- Machine learning perspective
- Functional perspective

Figure 1 shows an overview of our global taxonomy. The following sections explain the individual categories and justify the different elements within.

A. Requirements Engineering Perspective

The first categorization aspect we chose comes from the general phases that requirements engineering is divided into. Since we managed to find suitable research on most of those, it was a straight forward approach to map them into those categories. Specifically, we chose the phases *Elicitation*, *Analysis*, *Validation* and *Management* which cover the requirements engineering process for the most part.

While usually there is also a dedicated phase for documenting the requirements and formalizing the specification, we chose to skip over this one. Instead, we incorporated the little research which we found for this area into a more general *Management* phase that also covers some interesting applications of machine learning to handle requirements throughout the rest of the software engineering process.

1) Requirements Elicitation and Discovery: At the start of a project, it is important to determine all related stakeholders, note their specific concerns and subsequently their requirements.

During the requirements elicitation phase the project developers and stakeholders usually map out their individual needs and compile a list of concerns that the project has to fulfill from their point of view. Naturally, this depends very much on the type of project, as open source projects on the one hand with large user groups as stakeholders need to manage large data sets of potentially dissimilar entries (feature requests, issues, user stories etc.). On the other hand, individual projects targeted for specific customers potentially need to prioritize selecting the right requirements for their purpose.

Therefore, research has proven a number of ways to automate parts of the process: One popular method is to optimize the stakeholder selection by grouping them according to interests or their relation to the project [8], [11]. Additionally, models exist that simplify extracting functional requirements from a given set of goals or win conditions [12]. Ramadan et al. introduce another way of guidance in this process by describing a method of using artificial neural networks to select appropriate elicitation techniques [13].

2) Requirements Analysis and Negotiation: During the analysis phase, the requirements engineers try to correct any issues in the gathered set of requirements and come up with a prioritization scheme. As individual stakeholders have different interests in the project, it is essential to weight those influences and decide on a suitable set of project requirements. Byproducts of this process include concrete use case specifications and process descriptions.

In literature, the prioritization and ranking process actually represents a popular target for machine learning techniques. Perini et al. as an example showcase a semi-automated approach using a pair-wise ranking technique [14]. In the context of large requirements data sets, Avesani et al. developed a similar framework using pair-wise prioritization to learn a ranking model [9]. Furthermore, Sycara proposes a neural network and decision-tree-based model for conflict resolution during the negotiation process [15].

3) Requirements Validation and Verification: The selected and prioritized requirements on the one hand must fulfill a number of quality criteria (wording, ambiguity etc.) and on the other hand must finally cover the whole business case. In this phase the requirements data set is verified so that coverage of all necessary kinds of quality requirements can be ensured. As an example, software based around sensitive data must fulfill more and more detailed kinds of security criteria whereas real-time applications have to be evaluated especially for performance-critical issues.

Applications of machine learning in this category are similarly diverse, with solutions ranging from evaluating requirements based on quality metrics that are defined by a classifier [6] to checking security requirements specifications for completeness via *Natural Language Processing (NLP)* [16].

4) Requirements Management: In this last category we try to combine all approaches that aid the requirements engineering process in a different way or help to manage requirements throughout the development process.

Notable tools include measures to optimize the requirements tracing. This concerns the mapping of the requirements to artefacts that are produced during development and tracking of changes or adaptations of the same. Machine learning in this case is actively used to trace regulatory requirements as well as link the high-level specification to low level requirements [17], [5], [18]. Other studies developed models to verify the requirements coverage or map them to the corresponding tests [19], [20].

B. Machine Learning Perspective

Machine learning can generally be classified into two major categories: *Supervised* and *Unsupervised Learning*. A main difference between those two methods is, that while supervised learning generates a classifier or model based on a *labeled* training set (i.e. a prepared set where each data sample is

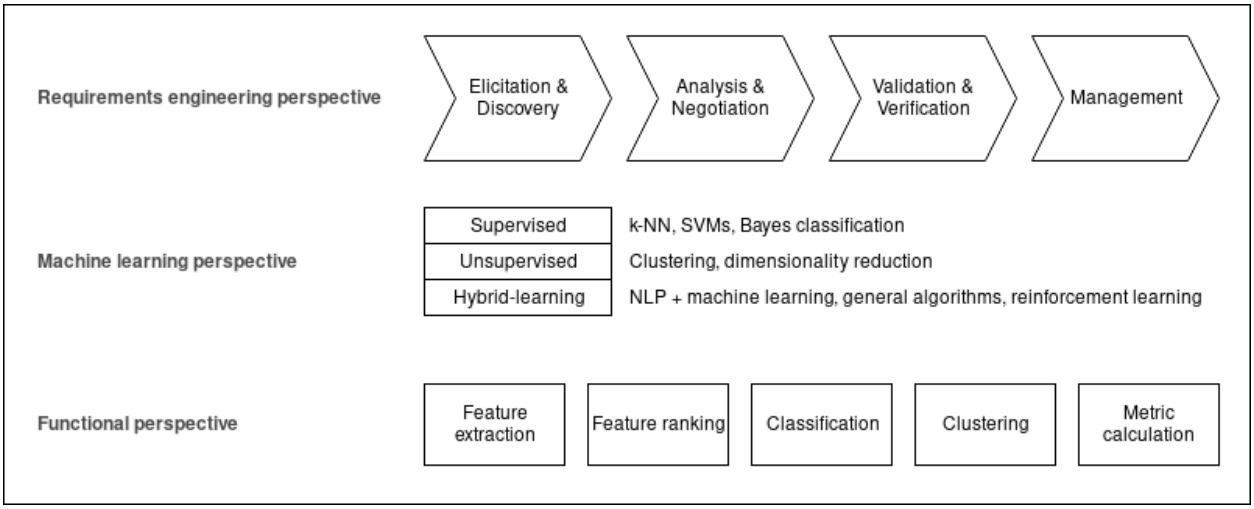

Figure 1. Overall taxonomy and categorization axes

manually equipped with its target category or value), unsupervised learning techniques tries to find commonalities between previously "unknown" data samples. The former approach is especially useful to determine regression or classification functions for future data, while the latter one can be used to determine the critical features of a data set or reduce it to its core dimensions (i.e. main entropy contributors).

Additionally, we introduce another category, namely *hybrid-learning*, that captures approaches that combine different techniques or use *artificial intelligence / natural language processing* in combination with machine learning.

1) Supervised learning: Regarding requirements engineering, the most used supervised learning techniques we could determine from research are *Support Vector Machines (SVM)*, *k-Nearest Neighbor Classification* and *Naive Bayes Classification*. Those techniques are especially useful to classify requirements based on a predefined categorization (i.e. quality requirements) [21], [10]. *SVMs* on the one hand try to determine a classification model via a separating hyperplane. With *k-NN* on the other hand, it is possible to create a classifier in form of a voting process where the k closest elements determine the class of an input sample.

2) Unsupervised learning: Unfortunately, specific unsupervised learning methods are rarely mentioned in literature but we could identify several instances were clustering algorithms are applied. Such methods are used to group data based on not well specified commonalities.

Techniques like *Principal Component Analysis (PCA)* or factor analysis as an example allow for reducing large data sets such as speech, unstructured text or visual data (i.e. extracting their *principal components*) as well as determine potential causes (unobserved *factors*) for variation in the data. Furthermore, *Independent Component Analysis (ICA)* can be applied to determine a sort of *blind* separation of the data into subcomponents.

Most often, clustering is applied after natural language processing aligned textual data sets to then determine the main keywords and attributes [22], [23]. We could find one study by Sheshasayee et al. who specifically apply *C-Means* and *K-Means* clustering for requirements engineering [24]. Chatzipetrou et al. apply *Principal Component Analysis (PCA)* in their preprocessing phase to determine those components within the data that explain the remainder of the data [25].

3) Hybrid-learning: The last section of the machine learning perspective unites all approaches that are combinations of supervised or unsupervised techniques or use other artificial-intelligence-based methods to transform the data. Notable examples include *natural language processing* techniques such as *bag or words*, *word unigrams* or *tf–idf* to handle unstructured requirements and extract keywords from them [26], [10]. Rahman et al. as an example use a *word vectorization* technique to preprocess their natural language documents before applying *deep learning* for classification [27].

C. Functional Perspective

To get a better understanding of how the different machine learning techniques are actually applied, we chose to categorize the functional aspects of the different approaches. Here we distinguish between the actual results that are generated via models and the usage scenarios.

1) Feature extraction: The general term "feature extraction" should incorporate any means of automatically generating information from (potentially unstructured) data sets. As such, natural language processing tools are a popular choice to determine requirements from stakeholder or user requests [28], [22].

2) Feature ranking: This category very much aligns with the requirements prioritization phase and mostly uses pair-wise

evaluation methods or the *Analytic Hierarchy Process* to rank requirements [14], [11], [25].

3) Classification: Requirements classification is actually the area for which we could find the most examples. As it is a typical step in the requirements process to determine which requirements belong to which categories (i.e. functional vs. non-functional, quality requirement categories), there are many existing models that help in automating this process [8], [10], [27], [21]. The different methods that may be applied are further explained in section IV during the classification step.

4) Clustering: For clustering, as already mentioned, we could identify a few approaches that apply techniques like *C-Means*, *K-Means* or *PCA* to raw requirements data [25], [24]. But clustering in general is widely used to reduce the input dimensions [22], [23].

5) Metric determination: The last category of application domains is the calculation of metrics from a given requirements set. This is most often done in combination with a priori classification of quality requirements which are then combined with a scoring algorithm [6]. The abstract framework in section IV goes into more detail about our proposed quality metrics.

IV. Developing a Machine Learning Process Pipeline

Based on the defined taxonomy, this section now fits the identified models and techniques into an abstract framework. This pipeline-based structure covers the key aspects of requirements engineering such as elicitation, prioritization and validation as well as describes the individual steps and transitions in between. In the process, we also mention the links to our taxonomy and describe how the pipeline steps correlate with the requirements engineering process.

Figure 2 shows the overall architecture of the model. Our process model is divided into two core parts: *Classification* and *Evaluation of Requirements*. The former describes ways of collecting requirements from heterogeneous data sets and mentions clustering and classification techniques from machine learning to produce a categorized set of requirements. The latter part then takes such a structured data set and proposes machine learning techniques to extract general quality metrics and subsequently evaluate the project requirements. For this evaluation we propose a simple scorer based on a set of extensible quality metrics.

A. Classification of Requirements

The framework first shows some measures to obtain actual data from a variety of data sources and then process it accordingly. The latter part mostly deals with alignment of the samples and early extraction of important features from large data sets. This data is then fed into a classification function (either for training a model or on an already trained classifier) to gain a category vector for the complete data set. The following sections describe this process in more detail as well as mention the corresponding machine learning methodologies that are applicable.

1) Requirements Data Sets: The initial step of the pipeline describes the data collection process. There we have to distinguish between different sources, such as user speech, stakeholder specifications or requests, notes or stories issued by users (e.g. in open source environments). Additionally, existing requirements documents may be used to guide similar projects.

As those data sources vary in structure, they have to be aligned, to allow further processing. In case of speech recognition, as an example, this could include extracting the textual representation of the spoken samples, whereas SRS documents usually are already in a predefined structure. The full data set is then collected in a database to be used in the subsequent processing steps.

2) Preprocessing: To gain information from unstructured data, we propose to use *natural language processing* techniques in order to extract keywords from written entries. This should give structure to the different data samples by removing redundant words (conjunctions, prepositions etc.) and stripping the texts of punctuation and abbreviations [26].

Specific techniques that are used at this stage are *TF-IDF*, *Word Unigrams*, *Support Vector Machines (SVMs)*, *Bayes* and *k-Nearest Neighbor Classifiers* [10]. Other methods include alignment of data such as removal of special characters or stop words via *artificial neural networks* such as *convolutional neural networks* or *recurrent neural networks* [27].

3) Classification: In order to be able to classify the requirements, it is first important to extract crucial information from the samples. In this case this includes keywords from a predefined selection of topics or categories. The classification process therefore extracts the different requirements from the samples and then groups them accordingly.

Regarding the categorization, this model assumes a static set of classes which may be different for specific use cases: A generic approach would be to categorize according to *quality requirements* - security, usability etc. - or to group *functional* and *non-functional* requirements. Ideally, the classification may also include domain specific areas like legal or environmental aspects, distinct performance criteria or specifics in terms of security and vulnerability.

To actually apply machine learning techniques, we have to find a model for the data set. In machine learning this is usually a parameterized function that categorizes input data based on the learned classifier. Usually, a classifier is trained by enriching a sample data set with static labels and verifying intermediate results with so-called *verification* or *test samples*. In our case the labels stem from the set of static categories, hence the labeled data set consists of pairs of requirement samples and their target category.

Once such a model is created and it satisfies the requirements for the process - i.e. balance, variance, accuracy - it should be applied to the actual input data. This allows us to assign specific categories of our set to individual requirements and at this point enables us to reason about the actual data.

Data classification is a typical use case for supervised learning techniques such as *SVMs*, *Naive Bayes* or *k-NN* but

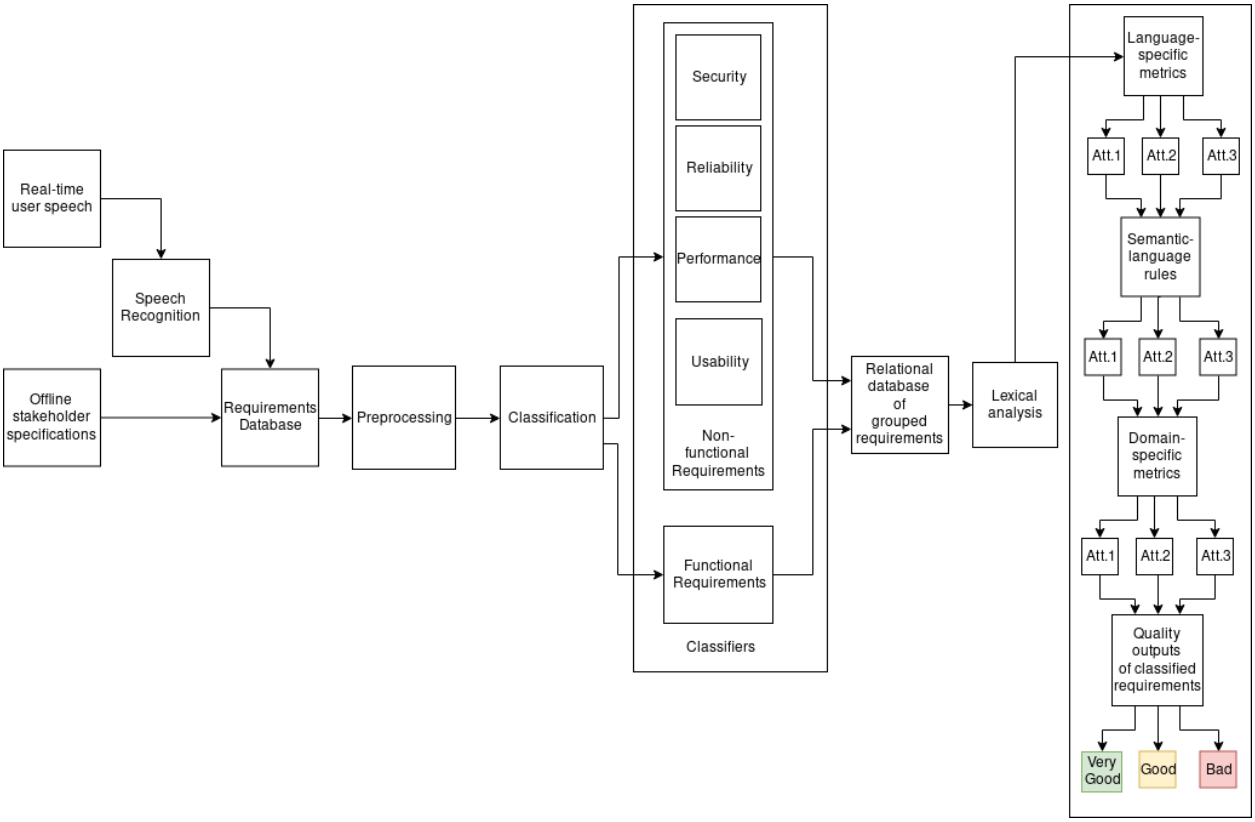

Figure 2. The abstract process pipeline for applying machine learning techniques in requirements engineering

research exists that also shows how to apply neural networks for determining requirements categories [21], [10], [27].

B. Evaluation of Requirements

Verification and validation of requirements is an important part of the requirements engineering process. Therefore, the second part of the abstract model deals with evaluating the (at this point) prepared and categorized requirements. For this purpose, we propose several quality metrics and categories that may be used for the evaluation. The other part of this section introduces machine learning concepts to actually derive such values from a labeled set of requirements.

1) Quality Metrics: For the quality metrics, we took inspiration from Parra et al., who propose a number of specific criteria on how to evaluate requirements based on their quality [6]. Genóva et al. additionally give a detailed description of quality metrics and elaborate on ways to measure those in textual data [29]. We suggest three main categories of quality metrics that cover the most important aspects:

- **Semantic language rules.** Metrics of this category evaluate requirements based on qualities like ambiguity, understandability and completeness. As an example, ambiguous requirements reduce the testability of the actual system and traceability in later stages of the development.
- **Language-specific metrics.** These rules give value to the grammatical correctness of requirements descriptions and evaluate properties like punctuation, usages of passive voice or readability.
- **Domain-specific metrics.** Requirements - while targeted for specific projects or problems - should be formulated in a more generic way and not over-specify individual domains. Metrics of this kind as an example evaluate the number of domain words or verbs that are used in requirements.

2) Estimating Quality Metrics: Determining quality metrics of specific requirements again requires a model for the target domain. There are different approaches on how to give value to requirement quality. One way would be to leverage discrete labels such as “good” or “bad” quality and use a classifier to categorize the requirements. While there are proposals to use existing generic data sets to train such a classification function [6], it is also possible to use data for a tailored domain as learning input.

3) Evaluation Criteria: The method we have preferred in this phase is to evaluate according to three major groups explained in Figure 3. Therefore, the percentage of the evaluation metrics shall play an important role to score the target requirements. Basically, requirements characterized with vocabulary

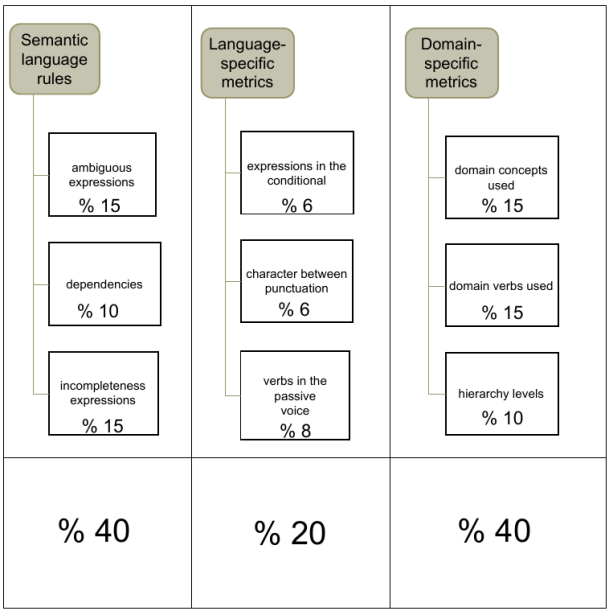

Figure 3. Evaluation criteria and scorer for quality metrics

in English are sensitive, hence building up a sentence in a proper way, makes them more precise, comprehensible and clear for the design and implementation process. Additionally, we listed the set of rules we established to calculate the quality of a requirement and give more information how this scoring table has been carried out. In order to identify those rules, we would use natural language processing methods and semantic reasoning algorithms.

At first, the textual sample for the target requirement is separated into words. This method is called the process of converting a sequence of characters (such as in a web page or book) into a sequence of tokens. After this conversion, initially all attributes of language specific metrics are applied to evaluate quality using the set contributions of each attribute.

Second, after language specific metrics are carried out on requirements, lexical components are compared according to levels of ambiguity, dependency among requirements and incompleteness of expressions. Afterwards, they are scored considering the percentage of each semantic rule listed in the overview figure. In conclusion, all lexical components derived from the requirements samples in the former phases are scored considering the domain concepts and domain verb used as well as the hierarchy level of verbs.

V. Conclusion

In the process of developing the abstract framework, we identified many different use cases for machine learning in requirements engineering from the literature. Most of the found techniques focus on classifying requirements or simplifying the elicitation process and describe models about how to automate individual steps.

We came up with a pipeline that combines many of those applications to, in the end, validate requirements based on specific quality metrics. The model should guide requirements engineers in integrating machine learning techniques into the process and show ways to automatically check the quality of a requirements data set. As this model was developed only theoretically, further research is required to give practical examples and evaluate its usability.

References

[1] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[2] C. Jones, *Software Engineering Best Practices*. McGraw-Hill, Inc., Aug. 2009. [Online]. Available: http://dl-1acm-1org-1my6qr5yk03e6.han.ubl.jku.at/citation.cfm?id=1594755

[3] B. Solemon, S. Sahibuddin, and A. A. A. Ghani, "Requirements Engineering Problems and Practices in Software Companies: An Industrial Survey," in *Advances in Software Engineering*, ser. Communications in Computer and Information Science, D. Ślęzak, T.-h. Kim, A. Kiumi, T. Jiang, J. Verner, and S. Abrahão, Eds. Berlin, Heidelberg: Springer, 2009, pp. 70–77.

[4] S. Wagner, D. Méndez Fernández, M. Felderer, and M. Kalinowski, "Requirements Engineering Practice and Problems in Agile Projects: Results from an International Survey," May 2017.

[5] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in *2013 21st IEEE International Requirements Engineering Conference (RE)*, Jul. 2013, pp. 52–61, iSSN: 1090-705X, 2332-6441.

[6] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Information and Software Technology*, vol. 67, pp. 180–195, Nov. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584915001299

[7] T. Iqbal, P. Elahidoost, and L. Lucio, "A Bird's Eye View on Requirements Engineering and Machine Learning," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. Nara, Japan: IEEE, Dec. 2018, pp. 11–20. [Online]. Available: https://ieeexplore.ieee.org/document/8719411/

[8] C. Castro-Herrera, J. Cleland-Huang, and B. Mobasher, "Enhancing Stakeholder Profiles to Improve Recommendations in Online Requirements Elicitation," in *2009 17th IEEE International Requirements Engineering Conference*. Atlanta, Georgia, USA: IEEE, Aug. 2009, pp. 37–46. [Online]. Available: http://ieeexplore.ieee.org/document/5328640/

[9] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Supporting the Requirements Prioritization Process. A Machine Learning approach." Jan. 2004, pp. 306–311.

[10] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," *Journal of Systems and Software*, vol. 138, pp. 108–123, Apr. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121217303096

[11] N. Mulla, "A New Approach to Requirement Elicitation Based on Stakeholder Recommendation andCollaborative Filtering," *International Journal of Software Engineering & Applications*, vol. 3, pp. 51–60, May 2012.

[12] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel, "Learning Operational Requirements from Goal Models," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 265–275. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2009.5070527

[13] N. Ramadan, A. A.Mohamed, and A. Abdelghany, "A Hybrid Machine Learning Model for Selecting Suitable Requirements Elicitation Techniques," *International Journal of Computer Science and Information Security (IJCSIS) – ISSN 1947-5500*, vol. 14, pp. 380–391, Jun. 2016.

[14] A. Perini, A. Susi, and P. Avesani, "A Machine Learning Approach to Software Requirements Prioritization," *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 445–461, Apr. 2013.

[15] K. P. Sycara, "Machine learning for intelligent support of conflict resolution," *Decision Support Systems*, vol. 10, pp. 121–136, 1993.

[16] A. Hayrapetian and R. Raje, "Empirically Analyzing and Evaluating Security Features in Software Requirements," in *Proceedings of the 11th Innovations in Software Engineering Conference on - ISEC '18*. Hyderabad, India: ACM Press, 2018, pp. 1–11. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3172871.3172879

[17] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 1, May 2010, pp. 155–164, iSSN: 1558-1225, 0270-5257.

[18] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 1, May 2010, pp. 95–104, iSSN: 1558-1225, 0270-5257.

[19] G. Barash, E. Farchi, I. Jayaraman, O. Raz, R. Tzoref-Brill, and M. Zalmanovici, "Bridging the gap between ML solutions and their business requirements using feature interactions," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*. Tallinn, Estonia: ACM Press, 2019, pp. 1048–1058. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3338906.3340442

[20] H. Sultanov, "Application of swarm and reinforcement learning techniques to requirements tracing," 2013.

[21] Z. Kurtanović and W. Maalej, "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 490–495, iSSN: 2332-6441.

[22] M. Ringsquandl and M. Schraps, "TAXONOMY EXTRACTION FROM AUTOMOTIVE NATURAL LANGUAGE REQUIREMENTS USING UNSUPERVISED LEARNING," 2014.

[23] V. Veerappa and E. Letier, "Clustering Stakeholders for Requirements Decision Making," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, D. Berry and X. Franch, Eds. Berlin, Heidelberg: Springer, 2011, pp. 202–208.

[24] A. Sheshasaayee and P. Sharmila, "Comparative study of fuzzy C means and K means algorithm for requirements clustering," *Indian Journal of Science and Technology*, vol. 7, pp. 853–857, Jan. 2014.

[25] P. Chatzipetrou, L. Angelis, P. Rovegard, and C. Wohlin, "Prioritization of Issues and Requirements by Cumulative Voting: A Compositional Data Analysis Framework," in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, Sep. 2010, pp. 361–370, iSSN: 2376-9505, 1089-6503.

[26] M. H. Osman and M. F. Zaharin, "Ambiguous software requirement specification detection: an automated approach," in *Proceedings of the 5th International Workshop on Requirements Engineering and Testing - RET '18*. Gothenburg, Sweden: ACM Press, 2018, pp. 33–40. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3195538.3195545

[27] M. A. Rahman, M. A. Haque, M. N. A. Tawhid, and M. S. Siddik, "Classifying non-functional requirements using RNN variants for quality software development," in *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation - MaLTeSQuE 2019*. Tallinn, Estonia: ACM Press, 2019, pp. 25–30. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3340482.3342745

[28] B. Mobasher and J. Cleland-Huang, "Recommender Systems in Requirements Engineering," *AI Magazine*, vol. 32, pp. 81–89, 2011.

[29] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno, "A Framework to Measure and Improve the Quality of Textual Requirements," *Requirements Engineering*, vol. 18, Mar. 2011.