Test Oracle using Semantic Analysis from Natural Language Requirements

Maryam Imtiaz Malik, Muddassar Azam Sindhu[a] and Rabeeh Ayaz Abbasi[b]
Department of Computer Science, Quaid-i-Azam University, Islamabad, 45320, Pakistan

Keywords: Natural Language Requirements, Software Testing, Test Oracle, Semantic Analysis.

Abstract: Automation of natural language based applications is a challenging task due to its semantics. This challenge is also confronted in the software testing field. In this paper, we provide a systematic literature review related to the semantic analysis of natural language requirements in the software testing field. The literature review assisted us in the identification of the substantial research gap related to the semantics-based natural language test oracle. To the best of our knowledge, we have not found any technique in which the semantics of test oracle from natural language requirements can be solved using Word Sense Disambiguation techniques. We have discussed our proposed approach to generate semantics-based test oracle from natural language requirements. Our proposed approach can be applied to any domain.

1 INTRODUCTION

Software testing is an important phase in the software development life cycle (SDLC). The software testing field has many research areas including test case generation, test case prioritization, software bug localization, test oracle, etc. In all areas, the requirements are given as input for proceeding the respective approach. These requirements are expressed either formally or informally. The informal software requirements are written in natural language by requirement engineers. Although, the natural language requirements are easy to understand without requiring any technical skills, but these requirements are ambiguous and often incomplete. Moreover, natural language requirements have another problem that is related to its semantics. The level of semantics can be taken broadly into two categories: word level and sentence level. As the meaning of words varies from context to context therefore the word and sentence level semantics should deal within a particular context.

Many researchers proposed different approaches regarding the semantics of natural language requirements. Therefore, in this position paper, we conducted a study in which semantics of natural language requirements in the software testing field are discussed. Further based on the research gap, we have also discussed our proposed approach.

In the rest of the paper, Section 2 provides the state of the art related to the semantics of natural language requirements in the software testing field. Section 3 discusses the motivation behind our research work. Section 4 and 5 discuss the research problem and proposed approach. And the last section 6 concludes the position paper along with discussing future directions of work.

[a] https://orcid.org/0000-0002-3411-9224
[b] https://orcid.org/0000-0002-3787-7039

2 STATE OF THE ART

In this section, we have discussed state of the art in the field of software testing related to the semantic analysis of natural language requirements. We have searched literature by the following set of keywords on Google Scholar.

- Natural Language Requirement
- Semantic analysis
- Software testing
- Semantic based Test oracle / Test oracle
- Test case generation
- Test case execution
- Natural Language Processing (NLP)

We further restricted our search to publication year since 2017, 2018, 2019, 2020 and any time. We also

Malik, M., Sindhu, M. and Abbasi, R.
Test Oracle using Semantic Analysis from Natural Language Requirements.
DOI: 10.5220/0009471903450352
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 2*, pages 345-352
ISBN: 978-989-758-423-7

checked the citation and references of most relevant papers. In the later subsections, we discuss the studies related to systematic literature review. Further, we discuss literature related to semantics of natural language in the respective field by categorizing them into seven broad categories that are discussed in the subsections.

2.1 Systematic Study

The authors of (Dadkhah et al., 2020) provide the literature review of software testing using semantic web.

(Garousi et al., 2018) discuss several papers related to natural language processing in the software testing field. (Garousi et al., 2018) also categorize some papers based on semantic analysis.

(Mustafa et al., 2017) also discuss literature for test case generation approaches from the models and specifications.

(Ahsan et al., 2017) discuss literature related to test case generation using natural language processing techniques.

2.2 Testing Framework

The authors of (Atefi and Alipour, 2019) have worked on an approach for automated testing of the conversational agent. The approach highlighted the test oracle for natural languages. The approach semantically compared the utterances using word2vec, Google's Universal Sentence Encoder and Facebook’s InferSent sentence embeddings.

(Wang et al., 2018) proposed an approach for automatically generating Object Constraint Language (OCL) constraints from the Use Case Modelling for System Tests Generation (UMTG) and domain model. The approach used semantic role labeling, Wordnet, and Verbnet for the generation of OCL constraints. These automated OCL constraints are used for system testing.

(Lin et al., 2017) proposed an approach in which the topic of the input field, GUI state, and clickable document object model (DOM) element are identified using latent semantic indexing. In the proposed approach the vectors are generated from DOM. These vectors are used for feature extraction by applying TF-IDF. The TF-IDF vector is used for singular value decomposition. The approach has used cosine similarity for assigning the topic to the DOM element.

(Masuda et al., 2015) proposed an approach for logic retrieval from Japanese specifications. The morphological and dependency parsing is used for logic retrieval.

(Sunil Kamalakar, 2013) proposed a behavioral driven development (BDD) approach for automatic generation of glue code from the natural language requirement. The approach takes application code/stub and BDD specification as an input. Reflection techniques are applied to code for the extraction of information related to class and methods. The probabilistic matcher is used for matching the specification with the properties of code. The probabilistic matcher includes edit distance, cosine, and cosine WordNet similarity, and Disco.

(Torres et al., 2006) proposed an approach for the extraction of natural language description from test cases in Communicating Sequential Processes. The ontology and case frames are used for storing domain classes and thematic roles respectively. Further, the lexicon is used for storing nouns, verbs, and modifiers.

2.3 Test Case Generation

(Mai et al., 2019) proposed an approach for the generation of security-based executable test cases. The test cases are generated using misuse case specification and test driver API. The approach used NLP techniques for security testing and also mapped the test API into ontology. Further, semantic role labeling is used for the generation of executable code.

(Wang et al., 2019), (Wang et al., 2015a), (Wang et al., 2015b) proposed an approach to generate system test cases from the use case specification. The authors have used the extended Restricted Use Case Modeling (RUCM) template along with the domain model as an input. The NLP techniques are applied to identify the RUCM steps, domain entities, alternative flows and references. Further, semantic role labeling is used to generate OCL constraints. These constraints are then used to generate a use case test model. The scenarios and input are generated from the use case test and domain model. These scenarios and input along with the mapping table are used to generate test cases.

(Moitra et al., 2019) proposed an approach to generate executable test cases from structured natural language requirement and domain ontology. These test cases are derived using formal analysis techniques.

In (Silva et al., 2019) the test cases are generated from requirements through colored Petri nets (CPN). The syntax tree and requirements frames are generated from controlled natural language requirements. These frames are used to create Data-Flow Reactive Systems (DFRS) and CPN models for test case generation.

(Mahalakshmi et al., 2018) proposed an approach for the generation of test cases from use cases. (Mahalakshmi et al., 2018) extracted the named-entity list from the use case by feeding features into the machine learning algorithm. The features are extracted using n-gram, term frequency, WordNet reference, etc. These named-entities are used to identify data elements in the decision table. Further, the scenario matrix and data elements are used to generate test cases.

(Anjalika et al., 2018) proposed an approach to generate test cases from user stories and ontology. The user story along with its corresponding epic number is provided as an input. The triplets are generated from user stories using natural language processing. The triplet contains the actor, action, and object. Further, human intervention is required to develop an ontology. The test cases are generated from these triplets and ontology.

(Mai et al., 2018) proposed an approach for the generation of executable test cases. The approach has taken misuse case specification, test driver API and an initial ontology as an input. The test diver API is mapped onto the provided ontology. They also used semantic role labeling for test input identification and executable test case generation. Further, the concrete test values are provided by the test engineers.

(Rane, 2017) proposed an approach to generate test cases from natural language requirements. They have used user story, test scenario description, dictionary and acceptance criteria as an input for the generation of test cases. The NLP techniques are applied to the input to generate frames. These frames are used for the generation of an activity graph and test cases. Although, the authors have claimed that the test cases are generated directly from user stories or through an activity diagram. But they have only discussed the generation of test cases through an activity diagram in detail.

(Carvalho et al., 2012) proposed an approach to analyze the CNLParser tool syntactically and produce a syntax tree as an output. In this approach, the output of Controlled Natural Language (CNL) parser is given as input to the Syntax Tree to Case Frame (ST2CF) tool. This tool is used to analyze the syntax tree semantically and generated the case frame. The case frame represents the thematic role of each element in a sentence based on the main verb. Further, the test vectors are generated using the Software Cost Reduction (SCR) tool. This approach is used for test case generation in (Carvalho et al., 2013), (Silva et al., 2016), (Carvalho et al., 2014b), (Carvalho et al., 2015) and model-based testing (Carvalho et al., 2014a).

(Santiago Junior and Vijaykumar, 2012), (de Santiago Junior, 2011) proposed an approach for the generation of abstract and executable test cases from natural language requirements. The approach modeled state chart for the generation of test cases. The modeled state chart is refined using the domain and word sense disambiguation (WSD). The WSD is used to remove the states containing self-transition. Besides, the pair of verbs is semantically analyzed using graph indegree and Jiang and Conrath similarity measures.

2.4 Test Case Prioritization

(Yang et al., 2017) provided an empirical performance evaluation of three NLP based test case prioritization techniques that are Risk, Div, and DivRisk. It is calculated using the average percentage of detected faults among the available test cases. In this study, the test cases are written in the Chinese language. These test cases are categorized into testing and training. The keywords from the training set are added into the dictionary of the testing set. These keywords are extracted using the Language Technology Platform (LTP) platform. Further, the nouns and verbs are selected and synonyms are manually identified. The words whose frequency is greater than average word frequency are added into the dictionary for performance evaluation. The result shows that Risk strategy has good performance as compared to others.

(Islam et al., 2012) proposed an approach to automatically prioritize test cases using the multi-objective function. (Islam et al., 2012) also prioritized the test cases based on the cost, code and requirement coverage. Moreover, the latent semantic indexing is used for traceability.

2.5 Software Duplicate Bug Detection and Localization

(Zhao and Harris, 2019) proposed an approach to generate assertions from the specification document. The sentences written in English language are semantically analyzed for the generation of formal assertions.

(Khatiwada et al., 2018) proposed an approach for automatic bug localization. (Khatiwada et al., 2018) semantically localize the software bug using normalized google distance and pointwise mutual information.

(Du et al., 2017) proposed an approach for classification of bug reports on the basis of fault types. The word2vec is used for classification.

(Lukins et al., 2010) proposed an approach for automatic bug localization. The approach has used the latent Dirichlet allocation technique for the identification of the topic and the query formation. These queries are created through the extraction of keywords from the bug report, title, and addition of words like synonyms.

(Runeson et al., 2007) proposed an approach to detect duplicate reports using NLP techniques. Among other NLP techniques and similarity measures, the synonyms are also used for duplicate detection.

Similarly, there are multiple studies including (Chen et al., 2018), (Yang et al., 2016), (Dit et al., 2008), (Kang, 2017) in which bug reports are detected using semantic similarity.

2.6 Test Oracle and Assertion Generation

(Goffi, 2018) also used their work (Blasi et al., 2017) in a thesis for generation of test oracles from Java comments present in source code. Further they have also suggested generation of test oracles from unstructured natural language as a future work.

The authors of (Blasi et al., 2017) extended their approach proposed in (Goffi et al., 2016). In this approach, the Java elements are compared syntactically and semantically with comments for the generation of the assertion. For semantical analysis, the approach has used Word Mover's Distance, Word2Vec and GloVe model.

Several NLP techniques are applied to the software artifact in (Ernst, 2017). The cosine similarity and TF-IDF are used to display error messages with the appropriate details. Further, edit distance or WordNet are used for the identification of the word similarity. The oracle assertions are also automatically generated from the Javadoc using parse trees, pattern matching, and lexical similarity. In (Ernst, 2017) English specifications are also converted into bashed commands through the translation process.

(Hu et al., 2011) proposed a semantic oracle using ontology and rule-based specification. The rules have antecedents and consequent which provide the test inputs and expected output respectively. These outputs are compared by oracle reasoner.

2.7 Functional Dependency and Traceability

(Tahvili et al., 2019) proposed an approach to detect and clustered the functionally dependent test cases. The authors have used Doc2vec for generating feature vectors. These feature vectors are clustered using the hierarchical density-based spatial clustering of applications with noise and fuzzy c-means clustering algorithms.

(Csuvik et al., 2019) proposed an approach for tracing the test with code using latent semantic indexing and word2vec.

3 MOTIVATION

The semantic analysis has extensive applicability in different areas including text mining, information retrieval, knowledge extraction etc. The extensive research in the literature revealed that semantic analysis also plays a vital role in the software testing field.

In the literature, the authors discussed a tool named DrQA for question answering (King et al., 2019) and (Makela, 2019). The tool used semantic analysis techniques like word embedding. The authors claim that the DrQA tool can be used to generate automated test oracle for testing AI systems. They also recommended that the models can be created from requirements and specifications to automate the test oracle (Makela, 2019). DrQA tool and different semantic analysis techniques within the software testing field motivated us to compare expected and observed outputs semantically from natural language requirements.

4 RESEARCH PROBLEM

In this position paper, we have provided an extensive literature on the applicability of semantic analysis in the software testing field. From the literature, it has been observed that there is an immense gap in providing semantic test oracle for natural language requirements-based test cases. We have retrieved only one relevant paper from literature in which utterances are semantically compared (Atefi and Alipour, 2019). In this paper, the testing framework is limited for conversational agents only. And the context module has used the design patterns. This approach relies on the determination of appropriate data set for semantic analysis.

In addition, many researchers proposed approaches for the generation of test cases from natural language requirements but these approaches do not perform semantic analysis of test oracle from natural language requirements.

There is no automated approach in which the testing framework leads from natural language requirements to semantic-based test oracle. Therefore, we are planning to generate automated semantic test oracle for natural language requirements.

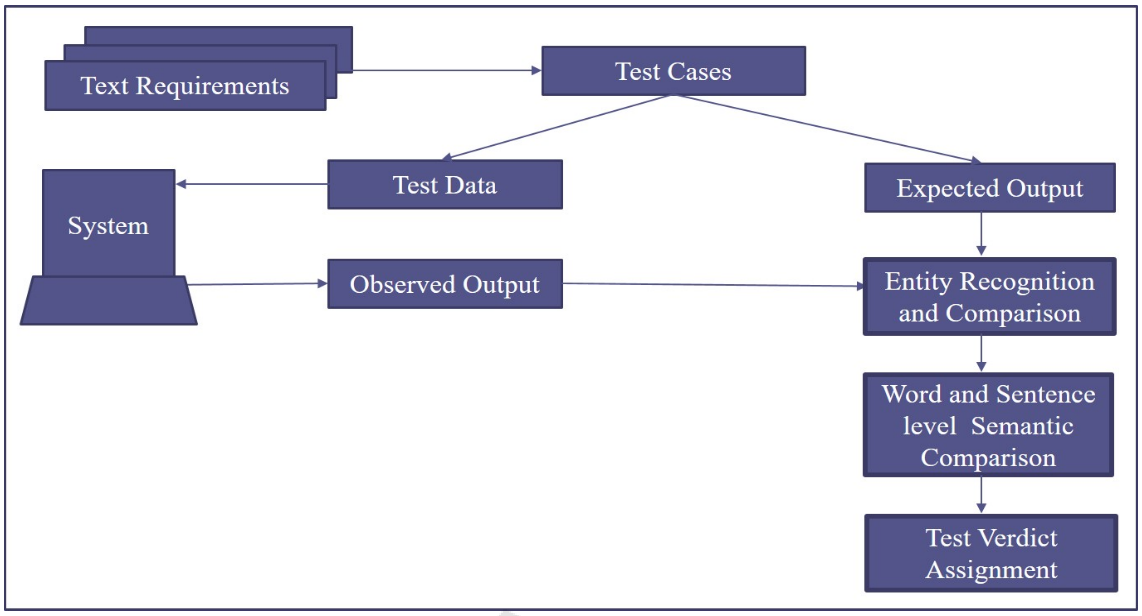

Figure 1: Proposed Approach.

5 PROPOSED APPROACH

Our proposed approach will generate test cases from natural language requirements and then compare the expected and observed outputs. Our proposed approach is not restricted to some particular domain. And it does not require additional dataset for semantic comparison. Figure 1 shows our proposed approach.

The proposed test oracle part is divided into multiple phases. The first phase is responsible for the identification of entities from outputs that do not require semantic similarity comparison. These entities may comprise a person's name, place, digits, etc. The named entity recognition or semantic role labeling will be applied for the identification of these entities. These entities are omitted or replaced by a constant word for further processing. The test oracle then compares these entities through some other proposed methodology.

In the second phase, we will apply semantic similarity both at word and sentence level. Our approach establishes the context using requirements for semantic similarity. Relying on requirements saves the extra effort and time of testers.

We will use WSD techniques for semantic comparison. WSD deals with different senses of a word. (Navigli, 2012) discussed three types of WSD: Supervised WSD, Unsupervised WSD, and Knowledge-based WSD. Supervised WSD approaches have applied machine learning techniques, knowledge-based WSD has used knowledge resources, and Unsupervised WSD has used unlabeled corpora for identifying the word sense in a context.

In our proposed approach we will evaluate several knowledge-based techniques and select the most suitable technique. The intuition behind using the knowledge-based technique is to not rely on the corpus for semantic similarity. WSD is AI-complete problem (Navigli, 2009). Consequently, to achieve better accuracy, we may use knowledge-based techniques along with other WSD techniques. As a supervised learning method requires human intervention for labeling therefore, we are not using supervised WSD. In the future, we will also use the word2vec approach by training on requirement documents.

In the third phase, our proposed approach will assign a test verdict based on the comparison of expected and observed outputs.

The proposed test oracle is illustrated using the following expected and observed outputs:

Expected Output: John's data is deleted from the system.

Observed Output: John's data is removed from the system.

The first phase of the proposed approach identifies John as an entity from expected and observed outputs. These entities are compared and replaced with a constant word. In the second phase, the expected and observed outputs are semantically compared. The parser identifies 'delete' and 'remove' as semantically similar words. Finally, a test pass verdict is assigned.

6 CONCLUSION

In this position paper, we raised an important problem related to the semantics of natural language requirements. We have discussed literature related to the semantics of natural language requirements in the software testing field. It is observed from the literature that there is little research carried on the automation of test oracle semantically. We plan to address these research gaps as the research problem in the software testing field. To solve the mentioned research problems, we have proposed an approach to automate the semantic test oracle from natural language requirements. The worthiness of our research work can be clarified from the research gap which originates after an extensive literature review. Our proposed approach is the first step towards the use of WSD techniques for solving a significant problem. Our approach can be applied to all applications dealing with natural language requirements.

In the future, we will automate the proposed approach. We will also evaluate the proposed approach from several case studies. We will also extend the literature review by covering some other search engines with additional keywords.

REFERENCES

Ahsan, I., Butt, W. H., Ahmed, M. A., and Anwar, M. W. (2017). A comprehensive investigation of natural language processing techniques and tools to generate automated test cases. In *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, ICC '17, pages 132:1–132:10, New York, NY, USA. ACM.

Anjalika, H. N., Salgado, M. T. Y., and Siriwardhana, P. I. (2018). An ontology based test case generation framework.

Atefi, S. and Alipour, M. A. (2019). An automated testing framework for conversational agents.

Blasi, A., Kuznetsov, K., Goffi, A., Castellanos, S. D., Gorla, A., Ernst, M. D., and PEZZ, M. (2017). Semantic-based analysis of javadoc comments.

Carvalho, G., Barros, F., Carvalho, A., Cavalcanti, A., Mota, A., and Sampaio, A. (2015). Nat2test tool: From natural language requirements to test cases based on csp. In Calinescu, R. and Rumpe, B., editors, *Software Engineering and Formal Methods*, pages 283–290, Cham. Springer International Publishing.

Carvalho, G., Barros, F., Lapschies, F., Schulze, U., and Peleska, J. (2014a). Model-based testing from controlled natural language requirements. In Artho, C. and Ölveczky, P. C., editors, *Formal Techniques for Safety-Critical Systems*, pages 19–35, Cham. Springer International Publishing.

Carvalho, G., Falcão, D., Barros, F., Sampaio, A., Mota, A., Motta, L., and Blackburn, M. (2013). Test case generation from natural language requirements based on scr specifications. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1217–1222, New York, NY, USA. ACM.

Carvalho, G., Falcão, D., Barros, F., Sampaio, A., Mota, A., and Motta, L. (2012). Nlreq2tvectors: A tool for generating test vectors from natural language requirements. Technical report, Technical report, UFPE.

Carvalho, G., Falcão, D., Barros, F., Sampaio, A., Mota, A., Motta, L., and Blackburn, M. (2014b). Nat2testscr: Test case generation from natural language requirements based on scr specifications. *Science of Computer Programming*, 95:275 – 297. Special Section: ACM SAC-SVT 2013 + Bytecode 2013.

Chen, M., Hu, D., Wang, T., Long, J., Yin, G., Yu, Y., and Zhang, Y. (2018). Using document embedding techniques for similar bug reports recommendation. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 811–814.

Csuvik, V., Kicsi, A., and Vidács, L. (2019). Source code level word embeddings in aiding semantic test-to-code traceability. In *2019 IEEE/ACM 10th International Symposium on Software and Systems Traceability (SST)*, pages 29–36.

Dadkhah, M., Araban, S., and Paydar, S. (2020). A systematic literature review on semantic web enabled software testing. *Journal of Systems and Software*, 162:110485.

de Santiago Junior, V. A. (2011). *SOLIMVA: A methodology for generating model-based test cases from natural language requirements and detecting incompleteness in software specifications.* PhD thesis, PhD thesis, Instituto Nacional de Pesquisas Espaciais (INPE).

Dit, B., Poshyvanyk, D., and Marcus, A. (2008). Measuring the semantic similarity of comments in bug reports. *Proc. of 1st STSM*, 8:64.

Du, X., Zheng, Z., Xiao, G., and Yin, B. (2017). The automatic classification of fault trigger based bug report. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 259–265.

Ernst, M. D. (2017). Natural Language is a Programming Language: Applying Natural Language Processing to Software Development. In Lerner, B. S., Bodík, R., and Krishnamurthi, S., editors, *2nd Summit on Advances in Programming Languages (SNAPL 2017)*, volume 71 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Garousi, V., Bauer, S., and Felderer, M. (2018). Nlp-assisted software testing: a systematic review. *arXiv preprint arXiv:1806.00696*.

Goffi, A. (2018). *Automating test oracles generation*. PhD thesis, Universita della Svizzera italiana.

Goffi, A., Gorla, A., Ernst, M. D., and Pezzè, M. (2016). Automatic generation of oracles for exceptional behaviors. In *Proceedings of the 25th International*

Symposium on Software Testing and Analysis, ISSTA 2016, pages 213–224, New York, NY, USA. ACM.

Hu, L., Bai, X., Zhang, Y., Lu, H., Ye, H., and Hou, K. (2011). Semantic-based test oracles. In *2011 IEEE 35th Annual Computer Software and Applications Conference(COMPSAC)*, volume 00, pages 640–649.

Islam, M. M., Marchetto, A., Scanniello, G., and Susi, A. (2012). A multi-objective technique to prioritize test cases based on latent semantic indexing. In *2012 16th European Conference on Software Maintenance and Reengineering(CSMR)*, volume 00, pages 21–30.

Kang, L. (2017). Automated duplicate bug reports detection-an experiment at axis communication ab. Master's thesis.

Khatiwada, S., Tushev, M., and Mahmoud, A. (2018). Just enough semantics: An information theoretic approach for ir-based software bug localization. *Information and Software Technology*, 93:45 – 57.

King, T. M., Arbon, J., Santiago, D., Adamo, D., Chin, W., and Shanmugam, R. (2019). Ai for testing today and tomorrow: Industry perspectives. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 81–88.

Lin, J., Wang, F., and Chu, P. (2017). Using semantic similarity in crawling-based web application testing. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 138–148.

Lukins, S. K., Kraft, N. A., and Etzkorn, L. H. (2010). Bug localization using latent dirichlet allocation. *Information and Software Technology*, 52(9):972 – 990.

Mahalakshmi, G., Vijayan, V., and Antony, B. (2018). Named entity recognition for automated test case generation. *INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY*, 15(1):112–120.

Mai, P. X., Pastore, F., Goknil, A., and Briand, L. C. (2018). A natural language programming approach for requirements-based security testing. *29th IEEE International Symposium on Software Reliability Engineering (ISSRE 2018)*.

Mai, P. X., Pastore, F., Goknil, A., and Briand, L. C. (2019). Mcp: A security testing tool driven by requirements. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 55–58.

Makela, M. (2019). Utilizing artificial intelligence in software testing. Master's thesis.

Masuda, S., Iwama, F., Hosokawa, N., Matsuodani, T., and Tsuda, K. (2015). Semantic analysis technique of logics retrieval for software testing from specification documents. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 1–6.

Moitra, A., Siu, K., Crapo, A. W., Durling, M., Li, M., Manolios, P., Meiners, M., and McMillan, C. (2019). Automating requirements analysis and test case generation. *Requirements Engineering*, 24(3):341–364.

Mustafa, A., Wan-Kadir, W. M., and Ibrahim, N. (2017). Comparative evaluation of the state-of-art requirements-based test case generation approaches. *International Journal on Advanced Science, Engineering and Information Technology*, 7:1567–1573.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2).

Navigli, R. (2012). A quick tour of word sense disambiguation, induction and related approaches. In Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., and Turán, G., editors, *SOFSEM 2012: Theory and Practice of Computer Science*, pages 115–129, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rane, P. (2017). Automatic generation of test cases for agile using natural language processing. Master's thesis, Virginia Tech.

Runeson, P., Alexandersson, M., and Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. In *29th International Conference on Software Engineering (ICSE'07)*, pages 499–510.

Santiago Junior, V. A. d. and Vijaykumar, N. L. (2012). Generating model-based test cases from natural language requirements for space application software. *Software Quality Journal*, 20(1):77–143.

Silva, B. C. F., Carvalho, G., and Sampaio, A. (2016). Test case generation from natural language requirements using cpn simulation. In Cornélio, M. and Roscoe, B., editors, *Formal Methods: Foundations and Applications*, pages 178–193, Cham. Springer International Publishing.

Silva, B. C. F., Carvalho, G., and Sampaio, A. (2019). Cpn simulation-based test case generation from controlled natural-language requirements. *Science of Computer Programming*, 181:111 – 139.

Sunil Kamalakar, F. (2013). Automatically generating tests from natural language descriptions of software behavior. Master's thesis, Virginia Tech.

Tahvili, S., Hatvani, L., Felderer, M., Afzal, W., and Bohlin, M. (2019). Automated functional dependency detection between test cases using doc2vec and clustering. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 19–26.

Torres, D., Leitao, D., and Barros, F. (2006). Motorola specnl: A hybrid system to generate nl descriptions from test case specifications. In *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, pages 45–45.

Wang, C., Pastore, F., and Briand, L. (2018). Automated generation of constraints from use case specifications to support system testing. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 23–33.

Wang, C., Pastore, F., Goknil, A., Briand, L., and Iqbal, Z. (2015a). Automatic generation of system test cases from use case specifications. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, ISSTA 2015, page 385–396, New York, NY, USA. Association for Computing Machinery.

Wang, C., Pastore, F., Goknil, A., and Briand, L. C. (2019). Automatic generation of system test cases from use case specifications: an nlp-based approach.

Wang, C., Pastore, F., Goknil, A., Briand, L. C., and Iqbal, Z. (2015b). Umtg: A toolset to automatically generate system test cases from use case specifications. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 942–945, New York, NY, USA. Association for Computing Machinery.

Yang, X., Lo, D., Xia, X., Bao, L., and Sun, J. (2016). Combining word embedding with information retrieval to recommend similar bug reports. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 127–137.

Yang, Y., Huang, X., Hao, X., Liu, Z., and Chen, Z. (2017). An industrial study of natural language processing based test case prioritization. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 548–549.

Zhao, J. and Harris, I. G. (2019). Automatic assertion generation from natural language specifications using subtree analysis. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 598–601.