

Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications

Sahar Tahvili
RISE SICS Västerås AB
Västerås, Sweden
sahar.tahvili@ri.se

Leo Hatvani
Mälardalen University
Västerås, Sweden
leo.hatvani@mdh.se

Michael Felderer
University of Innsbruck
Innsbruck, Austria
michael.felderer@uibk.ac.at

Wasif Afzal
Mälardalen University
Västerås, Sweden
wasif.afzal@mdh.se

Mehrdad Saadatmand
RISE SICS Västerås AB
Västerås, Sweden
mehrdad.saadatmand@ri.se

Markus Bohlin
RISE SICS Västerås AB
Västerås, Sweden
markus.bohlin@ri.se

ABSTRACT

One of the challenging issues in improving the test efficiency is that of achieving a balance between testing goals and testing resources. Test execution scheduling is one way of saving time and budget, where a set of test cases are grouped and tested at the same time. To have an optimal test execution schedule, all related information of a test case (e.g. execution time, functionality to be tested, dependency and similarity with other test cases) need to be analyzed. Test scheduling problem becomes more complicated at high-level testing, such as integration testing and especially in manual testing procedure. Test specifications at high-level are generally written in natural text by humans and usually contain ambiguity and uncertainty. Therefore, analyzing a test specification demands a strong learning algorithm. In this position paper, we propose a natural language processing (NLP) based approach that, given test specifications at the integration level, allows automatic detection of test cases' semantic dependencies. The proposed approach utilizes the Doc2Vec algorithm and converts each test case into a vector in n-dimensional space. These vectors are then grouped using the HDBSCAN clustering algorithm into semantic clusters. Finally, a set of cluster-based test scheduling strategies are proposed for execution. The proposed approach has been applied in a sub-system from the railway domain by analyzing an ongoing testing project at Bombardier Transportation AB, Sweden.

KEYWORDS

Software testing, Test scheduling, NLP, Dependency, Clustering, Doc2Vec, Optimization, HDBSCAN

ACM Reference Format:

Sahar Tahvili, Leo Hatvani, Michael Felderer, Wasif Afzal, Mehrdad Saadatmand, and Markus Bohlin. 2018. Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications. In *Proceedings of Requirements Engineering and Testing (RET'18)*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

For over a decade, applications of natural language processing (NLP) techniques were considered in different domains such as machine learning, deep learning, artificial intelligence, and static analysis.

Moreover, using the NLP techniques might be a suitable approach for improving the efficiency of documentation processes, which can be applied in different phases of software development life cycle (SDLC) such as requirement analysis, planning, and also testing. Since, in many cases, both requirements and test cases are formulated in natural text, NLP techniques can be used for analyzing the details of a textual specification. Some specific information such as required time for executing a test case, the test goal, functionality to be tested and the relationship between test cases can be detected by analyzing the textual specifications. One challenge for improving a testing process is detecting the dependency between test cases. By having an overview of dependencies between test cases, we can prioritize, select, and schedule test cases for execution. Some previous studies show that the dependency between test cases is too complex and will impact the result of a test execution [1]. In our previous work [15] we proposed a method for measuring the degree of dependency between test cases. In the proposed approach, the dependency degree has been measured manually between test cases. Since the number of required test cases for testing a system is rather large, a manual approach is not suitable to solve the problem efficiently. In the present position paper, we propose an NLP based approach for detecting the semantic dependency between test cases based on the textual test specifications. We also present a set of cluster-based strategies for scheduling tests. This paper provides the following contributions: (1) detecting the semantic dependency between test cases through analyzing test specifications, (2) clustering test cases based on the semantic dependency, and (3) proposing a set of cluster-based test scheduling strategies for execution.

2 BACKGROUND AND RELATED WORK

The lack of accurate test specifications analysis may lead to an inefficient testing process. Knowing execution time of test cases, requirement coverage and the dependency between test cases, are some required information that testers have to receive in the early stage of testing. Paying no attention to dependencies between test cases may lead to sequential failure of test cases and thereby waste of testing resources. In [13] we showed that the dependencies among test cases give partial information on the verdict of a test case from the verdict of another one. Moreover, Arlt et al. in [1] showed that test cases will be failed after each other if the testers do not care about the dependency between test cases. In the level of integration testing, individual software modules will be combined and

tested as a group, therefore, the problem of detecting the dependency between test cases becomes clearer. Test optimization is the main purpose of using test case dependency information, where test cases will be prioritized, scheduled and automatized by using this. Indumathi et al. in [7] classified the functional dependency between test cases into open and close dependency and prioritized test cases based on dependency structures. In the proposed classification, $TC2$ must be executed at any time but before $TC1$ if there is an open dependency between them. However, in a close dependency $TC2$ can be executed if $TC1$ has been executed before $TC2$. The dependency structures between test cases has been extracted manually in [7] thus the proposed approach is not scalable to handle a large set of test cases. Component dependency model (CMD) represents another type of dependency between test cases, introduced by Caliebe et al. in [3]. The proposed method in [3] is applicable for the component-based systems and dependencies can be identified through analyzing system structure and architecture, combined with additional information extracted from the system requirements.

3 SEMANTIC-DEPENDENCY MODEL

None of the researched dependencies in software testing domain, handles the semantic relationships between test cases. Generally, the semantic relationships are the associations that exist between the meanings of sentences or the meanings of phrases [12]. In this paper, we define a new type of dependency between test cases as:

Definition 3.1. A semantic dependency between test cases represents the conceptual associations between test cases which are not necessarily equivalent or hierarchical.

In other words, test case $TC1$ and $TC2$ are semantically dependent on each other, if both contain conceptual regularities or fundamental semantic relations. Moreover, detecting the semantic relationship between test cases which are conceptually associated requires deeper vocabulary analysis, therefore, the required input for identifying the semantic relationships is a test specification. Generally, a manual test case has multiple test steps including preconditions, test activities (action, expected reaction) and post-conditions. Sometimes test cases have some mutual steps, such as same preconditions which bring a system to an acceptable state before testing. Executing those test cases as a group can lead to an optimal usage of testing resources. The semantic dependency between manual test cases can be summarized as:

- Test cases have similar preconditions, initial state or post-conditions

The preconditions for a test case specify the setup needed for the test case to be executed successfully. Moreover, preconditions include the state that a system and its environment must satisfy before executing a test case [2]. The post-condition is a statement which describes the conditions that will be true when a test case has been executed successfully. The manual test cases are written by humans and are not easy to detect the similar preconditions and neither the post-conditions in a test specification. Furthermore, making a system ready for testing (running the preconditions in a test case) is a time and cost consuming process. Therefore, when a system is reaching an acceptable state, more test cases which

require the same system state can be tested as a group together. However, though running those test cases which require same post-conditions, we are able to save more time.

- Test cases test same functionality of a software

The main function which will be tested by a manual test case, is usually described in textual details into several test steps. Moreover, a functionality of a software to be tested, can be divided within different test cases. Creating a group of test cases which test the same functionality of a software can help testers to save more time. Test cases which have a semantic dependency might test same functionality of a software application and can be tested at the same time.

4 THE APPROACH

In this section, we describe our approach for scheduling manual test cases for execution based on the semantic dependency between them. The proposed approach is based on the valuation of test specification, as well as clustering of the dependent test cases. Finally, different test scheduling strategies, based on the semantic dependency between the test cases, will be proposed. Our approach is based on two main algorithms, which are able to handle a large set of data. Moreover, the testers can easily re-run the proposed algorithms, when new test specifications are added to the set. The steps of our proposed approach are shown in Figure 1, where the blue color represents the NLP stage and the green color shows the cluster analysis stages. In the following text, we describe the steps of the proposed approach:

Step 1: detecting the semantic relationships between manual test cases using the Doc2Vec algorithm.

Step 2: cluster the semantic dependent test cases using HDBSCAN algorithm.

Step 3: propose a set of non-clusterable test cases as independent test cases.

Step 4: use one of the cluster-based scheduling strategies for test execution.

As illustrated in Figure 1, the required input for running our approach is a test specification. By running the Doc2Vec algorithm, a set of vectors (which represent each test case) in a n -dimensional space will be generated. Thereby, HDBSCAN algorithm classifies test cases into several clusters. Finally a set of scheduling strategies will be proposed for execution. In other words, the expected output of the proposed approach is a set of semantic dependent test cases, arranged for execution, based on scheduling strategies.

4.1 Document Embedding Using Doc2Vec

The Doc2Vec algorithm consists a set of shallow and two-layer neural networks models which are designed to produce document embedding [10]. The basis of the Doc2Vec is based on learning representations forward in such a way that Doc2Vec takes as its input a large corpus of text and generates a unique vector in n -dimensional space for each unique document in the corpus [8]. From here, we can measure the similarity between two documents by calculating the Cosine similarity of the two vectors that represent the corresponding documents. The similarities in the n -dimensional space can be extracted to do comparison [8] where the words with similar meaning end up lying close to each other. Furthermore, the

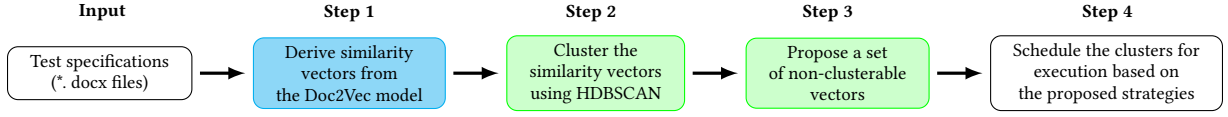


Figure 1: The steps of the proposed approach

Doc2Vec uses vector arithmetic to work with analogies, for instance the famous example: *Doctor - Man + Woman = Nurse* and also *USB - Port + Display = HDMI*. In this study we train the Doc2Vec algorithm in such a way that each test specification has been considered as an input document. As the first step, Doc2Vec extracts the semantic dependency between test specifications automatically, through utilizing deep linguistic patterns which have been defined over the dependency grammar of sentences [8].

4.2 Clustering with HDBSCAN

The second step of our proposed approach will deal with clustering. After running the Doc2Vec algorithm, a set of high dimensional vectors (which represents each test specification) will be generated. The vectors which have less distances to each other will be classified as a cluster. This step, will be performed by a fast and robust algorithm called HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), which is able to handle large high-dimensional data sets and separate clusters from the noise. The HDBSCAN measures the distance between the vectors and provides a set of clusters and also a set of non-clusterable vectors. Other clustering algorithms are designed to cluster every vectors to some cluster, which indicates their inability to handle noise in the clustering process, or, in some case, the algorithms cannot properly process high-dimensional data. In this work, we interpret the non-clusterable vectors as independent test cases, which can be executed in no particular order. Furthermore, each cluster consists of a set of test cases which have a semantic dependency and must be tested together at the same time.

4.3 Cluster-Based Test Scheduling Strategies

Determination of the possible strategies for test scheduling and selecting a proper strategy is assigned in the last step of our approach. In this section, we propose a set of cluster-based test case scheduling strategies based on the semantic dependency between test cases. The following definitions are applicable:

Definition 4.1. Let $C := \{C_1, C_2, \dots, C_n\}$ where, each C_j is a cluster, $j = 1, 2, \dots, n$.

Definition 4.2. Let cardinality of each C_j be K_j , where $K_j \in \mathbb{N}$ and $K_j > 1$. We define K_j as a cluster size.

In other words, the size of cluster indicates the number of test cases per cluster.

Definition 4.3. $\mathcal{P} := \{P_1, P_2, \dots, P_m\}$ where each P_j is a unique property for all member of C_j ; that is $\forall x \in C_j, \exists ! P_{s_j}$ such that $x \in P_{s_j}$, for every $j \in \{1, 2, \dots, n\}$.

In this work, we utilize the functional requirement group (FG) as a cluster property, where, every single test case belongs to one functional group and tests a specific part of a system under test. For

instance, *brake system* and *air supply* are two different functional groups.

Definition 4.4. We define t_j as a time function for every C_j such that $t_j : C_j \rightarrow (0, \infty)$.

The time function in the present work, represents the required time that a cluster of test cases takes for execution (the sum of test cases execution time). Since each test cases take different time for execution, then t_j will be changed for every single C_j . We need to consider that, execution time for test cases can be predicted by performing some regression analysis on previous executed test cases [14]. If there is no execution data available for test cases, we can assume all test cases have the same execution time. Using the proposed definitions, we define the following cluster-based strategies for test case scheduling:

- **Strategy 1 :** C_i has a higher priority than C_j if and only if $K_i > K_j$, where $i, j \in \{1, 2, \dots, n\}$, moreover this strategy can be called as *increasingly ordering*. However, strategy 1 can be defined as *decreasingly ordering* such that: C_i has a higher priority than C_j if and only if $K_i < K_j$, where $i, j \in \{1, 2, \dots, n\}$

In other words, the clusters will be ranked in strategy 1 based on their size. However, if two (or more) clusters have a same size ($K_i = K_j$), then strategy 1 is not applicable and a new strategy should be considered.

- **Strategy 2 :** Let $C_i = \{x_{1,i}, x_{2,i}, \dots, x_{K_i,i}\}$, $C_j = \{x_{1,j}, x_{2,j}, \dots, x_{K_j,j}\}$, where $K_i = K_j$. If $x_{l,i} \in P_{s_{l,i}}$ and $x_{l,j} \in P_{s_{l,j}}$, for $l=1, 2, \dots, K_i$. We define: $\mathcal{I} := \{P_{s_{1,i}}, P_{s_{2,i}}, \dots, P_{s_{K_i,i}}\}$ and $\mathcal{J} := \{P_{s_{1,j}}, P_{s_{2,j}}, \dots, P_{s_{K_j,j}}\}$. Thus C_i has a higher priority than C_j if and only if $|\mathcal{I}| > |\mathcal{J}|$.

In strategy 2, the clusters which contain test case with different property (functional group) will be top ranked. In fact, if C_i includes two test cases with two different properties (2 unique FGs), has a higher priority than C_j with the same size (2 test cases) which test just one FG (test cases have a same property). However, strategy 2 is not applicable if two (or more) clusters contain test cases which have a same property ($|\mathcal{I}| = |\mathcal{J}|$) with the same size ($K_i = K_j$).

- **Strategy 3 :** We define $T_i := \sum_{k=1}^{K_i} t_i(x_{k,i})$, $x_{k,i} \in C_i$ and $T_j := \sum_{k=1}^{K_j} t_j(x_{k,j})$, $x_{k,j} \in C_j$, where $K_i = K_j$ and $k \in \{1, 2, \dots, K_i\}$. Thus C_i has a higher priority than C_j if and only if $T_i < T_j$. However if $T_i = T_j$, there is no priority ordering to those clusters.

In strategy 3, the clusters which take less time for execution will be top ranked. Strategy 3 is applicable when an estimation of execution time for test cases is available. As stated before, through historical analysis of previously executed test cases, we are able to predict the execution time for test cases [14]. Furthermore, if we assume same execution time for test cases we are faced with the situation that two (or more) clusters have a same size and same property, we can run those clusters without any order.

5 PROOF OF CONCEPT

The feasibility of the proposed approach has been done through studying an on-going project for underground subway train in Stockholm, called C30 project at Bombardier Transportation AB.

5.1 Case Study Report

The units of analysis in this case study are manual test cases at the integration testing level for an ongoing project called C30 project. A total of 29 test suites, which are designed for testing different functional groups, have been extracted from the DOORS database at Bombardier. The extracted test cases are converted into vectors using paragraph-vectors [4] implementation of the Doc2Vec model, and then the test cases have been clustered by applying HDBSCAN [5] algorithm. Table 1 lists the functional groups and the number of associated test cases for testing various FG. Moreover, these test cases are run at a sub-system level, meaning that they are more time consuming to run than tests at unit level [6]. Therefore, executing dependent test cases which require same setup and conditions can help testers significantly reduce the testing time.

No.	Functional group	Test Case	Steps	Words
1	ATP	6	35	458
2	Air Supply	2	19	844
3	Bogie	4	29	779
4	CCTV	7	28	476
5	DVS	1	5	141
...
28	Vehicle Coupler	11	40	488
29	Windscreen	4	34	345

Table 1: Functional group with associated test cases (partial)

Moreover, Table 2 represents the clusters of test cases which are generated by HDBSCAN.

Cluster	Test Case	Functional Groups
C_u	126	Independent test cases
C_1	6	Drive and Brake Function, DVS, ATP, Traffic Radio Functions
C_2	4	Drive and Brake Function, Safe Exterior Access, CCTV
C_3	9	Safe Exterior Access, Train Inauguration
C_4	3	Safe SITS
C_5	7	Safe Bogie, General Requirements
C_6	8	Train Inauguration Functions, Drive and Brake Function
C_7	6	Exterior and Interior Access Function, Safe SITS
...

Table 2: The proposed clusters for scheduling (partial)

In Table 2, 126 test cases are identified as independent test cases. A total of 93 clusters are determined for dependent test cases (not all clusters are shown in Table 2). As explained earlier, test cases in Table 2 have been created for integration testing, where individual FGs will be combined and tested as a group, moreover, most of the generated clusters in Table 2 are included test cases from different FGs. However, some clusters such as C_4 contain test cases from just one FG. In this study, we just rank some of the generated clusters in Table 2 according to the proposed strategies in Section 4. However, Table 3 represents the number of 7 clusters which are ranked based on three different strategies.

As stated before, a method for estimating the execution time for manual test cases has been proposed by us previously [14], thereby, strategy 3 is applicable in this work. As we can see in Table 3, the

Strategy	Cluster
Strategy 1	$C_3, C_6, C_5, C_7, C_1, C_2, C_4$
Strategy 2	$C_1, C_2, C_3, C_5, C_6, C_7, C_4$
Strategy 3	$C_5, C_4, C_6, C_1, C_2, C_7, C_3$

Table 3: The proposed scheduling strategies

execution time for 7 tests cases in cluster C_5 is less than 3 test cases in cluster C_4 , therefore C_5 has a higher priority than C_4 . However, independent test cases have been clustered in C_u which can be executed without any particular order or other criteria such as execution time and requirement coverage can be considered for ranking them.

6 DISCUSSION & FUTURE EXTENSIONS

To detect different type of dependencies between test cases, various phases of a software development life cycle (SDLC) such as design, requirements and testing need to be analyzed. In this paper, we strive to propose a solution for such a situation that other information (requirement, internal signals, software architecture, etc.) are assumed to be not available. Moreover, executing a set of test cases which require the same system and environment setting is extractable from test specification. In the future, we are going to analyze the system requirement specifications (SRS) in order to detect other type of dependencies between test cases. However, other clustering algorithms such as Fuzzy C-Means, K-Means can be utilized for the clustering part of the proposed approach. Finally, a decision support system can be designed to schedule test cases for execution, based on the dependencies between test cases, requirement coverage and execution time.

7 SUMMARY & CONCLUSION

In this position paper, we proposed a cluster-based approach for scheduling integration test cases based on the semantic dependency between test specifications. The proposed approach has been applied to an industrial use case at Bombardier Transportation AB. The results of the proof of concept indicate that the concept of semantic dependency exists between integration test cases from different functional groups and can be detected through text analysis. Additionally, test cases have been divided into several clusters based on their semantic dependencies. Moreover, three different scheduling strategies have been proposed by us, where test cases will be given a different order for execution based on the proposed strategies. Finally, we interpret non-clusterable vectors as independent test cases.

REFERENCES

- [1] S. Arlt, T. Morciniec, A. Podelski, and S. Wagner. 2015. If A Fails, Can B Still Succeed? Inferring Dependencies between Test Results in Automotive System Testing. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. 1–10. <https://doi.org/10.1109/ICST.2015.7102593>
- [2] M. Bertrand. 1997. *Object-oriented Software Construction*. Prentice-Hall, Inc.
- [3] P. Caliebe, T. Herpel, and R. Geman. 2012. Dependency-Based Test Case Selection and Prioritization in Embedded Systems. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. 731–735. <https://doi.org/10.1109/ICST.2012.164>
- [4] L. Hatvani. 2018. Paragraph-vectors implementation. <https://github.com/inejc/paragraph-vectors>. (2018).
- [5] L. Hatvani. 2018. The used implementation of the HDBSCAN algorithm. <https://github.com/scikit-learn-contrib/hdbscan>. (2018).
- [6] H. Hemmati, L. Briand, A. Arcuri, and Sh. Ali. 2010. An enhanced test case selection approach for model-based testing: An industrial case study. In *Proceedings*

- of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'10). ACM, New York, NY, USA.
- [7] C.P. Indumathi and K. Selvamani. 2015. Test Cases Prioritization Using Open Dependency Structure Algorithm. *Procedia Computer Science* 48 (2015), 250 – 255. <https://doi.org/10.1016/j.procs.2015.04.178> International Conference on Computer, Communication and Convergence (ICCC 2015).
 - [8] Q. Le and T. Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Eric P. Xing and Tony Jebara (Eds.), Vol. 32. PMLR, Beijing, China, 1188–1196. <http://proceedings.mlr.press/v32/le14.html>
 - [9] I. Medeiros, N. Neves, and M. Correia. 2016. DEKANT: A Static Analysis Tool That Learns to Detect Web Application Vulnerabilities. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA 2016)*. ACM, New York, NY, USA, 1–11. <https://doi.org/10.1145/2931037.2931041>
 - [10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR abs/1301.3781* (2013). arXiv:1301.3781 <http://arxiv.org/abs/1301.3781>
 - [11] P. Runeson, , and H. Martin. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (19 Dec 2008), 131. <https://doi.org/10.1007/s10664-008-9102-8>
 - [12] V. C. Storey. 1993. Understanding semantic relationships. *The VLDB Journal* 2, 4 (01 Oct 1993), 455–488. <https://doi.org/10.1007/BF01263048>
 - [13] S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal, and D. Sundmark. 2016. Cost-Benefit Analysis of Using Dependency Knowledge at Integration Testing. In *The 17th International Conference On Product-Focused Software Process Improvement*. <http://www.es.mdh.se/publications/4438->
 - [14] S. Tahvili, M. Saadatmand, M. Bohlin, W. Afzal, and Sh. Hasan Ameerjan. 2017. Towards Execution Time Prediction for Test Cases from Test Specification. In *43rd Euromicro Conference on Software Engineering and Advanced Applications*. <http://www.es.mdh.se/publications/4758->
 - [15] S. Tahvili, M. Saadatmand, S. Larsson, W. Afzal, M. Bohlin, and D. Sundmark. 2016. Dynamic Integration Test Selection Based on Test Case Dependencies. In *The 11th Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques*. <http://www.es.mdh.se/publications/4298->