Proceedings of the Federated Conference on Computer Science and Information Systems pp. 657–660
DOI: 10.15439/2020F163
ISSN 2300-5963 ACSIS, Vol. 21

Community Detection in Model-based Testing to Address Scalability: Study Design

Alper Silistre*, Onur Kilincceker†, Fevzi Belli‡, Moharram Challenger§ and Geylani Kardas*
*International Computer Institute, Ege University, Izmir, Turkey. Email: alpersilistre@gmail.com, geylani.kardas@ege.edu.tr
†University of Paderborn, Paderborn, Germany. Mugla Sitki Kocman University, Mugla, Turkey. Email: okilinc@mail.upb.de
‡University of Paderborn, Paderborn, Germany. Izmir Institute of Technology, Izmir, Turkey. Email: belli@upb.de
§University of Antwerp and Flanders Make, Belgium. Email: moharram.challenger@uantwerpen.be

***Abstract*—Model-based GUI testing has achieved widespread recognition in academy thanks to its advantages compared to code-based testing due to its potentials to automate testing and the ability to cover bigger parts more efficiently. In this study design paper, we address the scalability part of the model-based GUI testing by using community detection algorithms. A case study is presented as an example of possible improvements to make a model-based testing approach more efficient. We demonstrate layered ESG models as an example of our approach to consider the scalability problem. We present rough calculations with expected results, which show 9 times smaller time and space units for 100 events in the ESG model when a community detection algorithm is applied.**

I. Introduction

SOFTWARE testing is of critical importance considering today's technological developments. The solution methods proposed in this area differ according to the software systems to be tested. Graphical user interfaces (GUIs) are an integral part of the web and mobile software systems. Many model-based approaches have been proposed varying based on the model used for testing the GUI functions. These models differ in terms of semantics and syntactic terms. Scalability is shown as one of the disadvantages of all model-based approaches. As the model grows, operations such as test production become more difficult, thus creating the problem of scalability.

Model-based testing is a testing method that executes test cases generated from the abstract view of a system under test (SUT). This abstraction specifies the behavior of the system so we can model it with one of the different models such as Finite State Machine (FSM) [1], Event Sequence Graph (ESG) [4] [5], Event Flow Graph (EFG) [2] [3], and Regular Expression (RE) [6][7]. The major benefit is, code-based testing is a time consuming and error-prone method to cover all cases of the SUT while we can generate and run test sequences efficiently with model-based testing. There are many automation tools to achieve this task and many papers worked on this topic for several decades.

Model-based approaches suggest a hierarchical and layered structure at the model generation stage. No specific approach has been found in the literature regarding the problem that may arise if the tester who created the model skipped this important issue. In this study, the community detection approach, which is frequently used in different areas, is proposed to eliminate this serious problem. Thus, the model expressed as layered will be hierarchical, that is, a layered structure by community detection method. In the scope of the study, an ESG model will be layered, and then automatic test set generation and test execution operations will be performed on this ESG model.

The current work in this paper is a design study. We briefly review the literature and provide proof of concept with a case study to support the proposed approach. Expected results with the scope of the current work are discussed, and rough theoretic calculations are presented.

The rest of the paper is organized as follows: Section 2 gives the related work on models used in GUI testing and community detection problem. Section 3 introduces the proposed approach. A case study to exemplify the proposed approach is given in section 4. Expected results and implications with possible threats to the validity of the proposed approach are presented in section 5. Finally, section 6 concludes the paper.

II. Related Work

This section briefly presents related literature regarding model-based GUI testing and community detection problem.

A. Model based GUI Testing

Shehady and Siewiorek [1], introduce a model to be used in model-based testing called VFSM (Variable Finite State Machine) for the GUI with fewer states than a formal FSM model. The VFSM and FSM are equivalent at its core. So, they show how to convert a VFSM to an FSM to create a test suite. The major benefit here is that the total state count is less which makes it more efficient in terms of end-to-end test generation and execution.

Memon et al. [2] come up with a new technique that helps automated test generation by using ESG models. Essentially, this technique is a planning algorithm that uses AI. The algorithm needs start and final states of the model and several defined operators to work on the model. It creates test sequences between these start and final states by looking into GUI interactions and events between these states.

Memon [3] shows event-flow model generation which he describes as one scalable model for using in the model-based testing area. Event-space exploration (ESES) strategies are used inside the proposed method. The event-flow model is a combination of different models which are assessed in the

paper. In order to reduce the cost of model creation, the process is also automated.

Belli [4] presents a new approach called the 'holistic' approach. He discusses that in order to test the GUI of a system properly, we must take incorrect test cases along with correct test cases into account. The GUI of an application should work without failing even the events are illegal. This is an important part of achieving complete system coverage.

Belli et al. [5] examine existing work on models that have been used in model-based GUI testing such as; ESG, EFG, etc., and analyze these model notations and how to create mutation from them. They also present ways to apply test generation from models and additional optimization techniques.

Kilincceker et al. [6] introduce RE as being a model-based testing method and use RE to model hardware design combined with RE-based test generation. RE is represented by an abstract syntax tree and a tree traversal algorithm is used in the test generation. RE-based coverage criteria are used to assess the adequacy of the testing method. Kilincceker and Belli [16] propose novel coverage criteria based on the analysis of a RE model. These coverage criteria are used to generate test sequences for testing GUI systems in [7] by means of random test generation. They also use to test GUI of mobile applications [13], hardware design [15], and web-based systems [16] combined with holistic testing.

B. Community Detection

Harary et al. [8] take into account the result of a Festinger's work [17] which was finding a clique in a group depending on whether certain elements satisfy required conditions and improve this with a new study to find all cliques in a group that has three or fewer cliques with a concept called unicliqual person. They then remove the restrictions with an "inductive reduction method".

Fortunato [9] explains community detection in graphs in very detail from main definitions to different methods to detect these communities, with example algorithms and techniques. He gives details about his ideas on the topic and mentions the given problem yet to be solved properly.

Leskovec et al. [10] examine different algorithms for network community detection to understand them better and compare them with each other in terms of performance and their capabilities. They also take biases in those algorithms into consideration while conducting their study. They show how complicated to detect communities in large network groups.

Sadi et al. [11] study community detection algorithms with a method that is using Ant Colony Optimization to reduce network graphs without losing its ability to solve the given problem. They work especially on the scalability part of the topic because the cost of computation is a lot in any given large network graph. They aim to reduce the number of nodes and then applying algorithms to work on this reduced graph.

To the best of our knowledge, there is no work to address scalability for model-based testing by utilizing the community detection algorithm. To this end, the current work aims to fill the gap by providing a community detection algorithm to address the scalability problem of model-based testing.

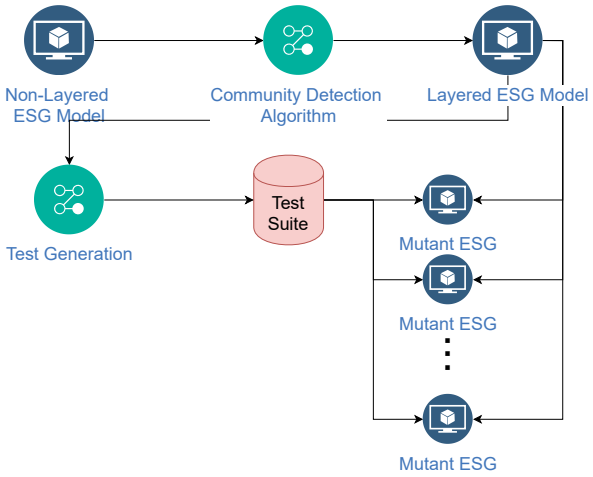

Fig. 1. The proposed approach

III. The Proposed Approach

The proposed approach offers a community detection algorithm on a non-layered ESG model to obtain a layered one. It also includes further necessary steps in conventional model-based testing approaches. These further steps, apply to the layered ESG model, are test generation and test execution as depicted in Fig. 1. By detecting communities in our model, we will treat them as sub-graphs. These sub-graphs will have their own process to generate test sequences with a model-based test generation and execution process. This sub-graph will be treated as a single node in the main graph. Since this sub-graph has nodes that are part of the community, extracting their interaction with other nodes in the main graph will be beneficial in terms of scalability because we do not need to create test sequences that involve nodes in a community to have a relationship with another node in a higher level graph.

Moreover, the test suite resulted from the test generation step becomes more compact and useful from the layered ESG model. Finally, the test suite is executed on mutant layered ESG models to evaluate their effectiveness using mutation score. To do this, the current work utilizes model-based mutation testing by means of appropriate model mutation operators given in [16].

IV. Case Study

The community detection step in the proposed approach is exemplified in a case study that is the internal part of a commercial tourist web page namely ISELTA [1]. The case study is the Special Module of ISELTA web page. Special Module enables agents to offer special advertisements.

Special Module of ISELTA contains arrival and departure dates, accommodation type, number of items, total price, description nation and international, name of the advertisement.

[1]ISELTA, Available at: http://iselta.ivknet.de/

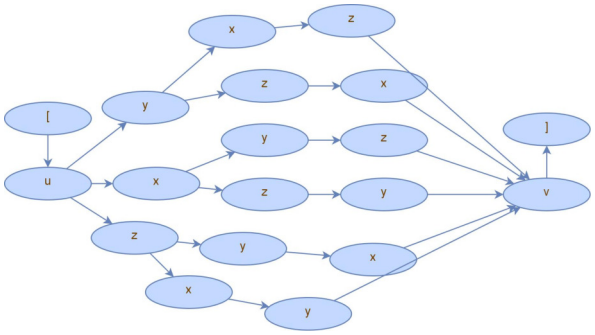

Fig. 2. ESG model of the Add Layer for ISELTA Special Module

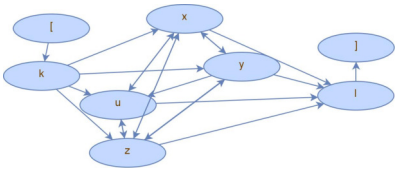

Fig. 3. ESG model of the Edit Layer for ISELTA Special Module

We represent events of "number of items", "total price", "description" and "name" input fields of the ISELTA Special Module in ESG models. Other events are neglected.

Special Module is represented as a non-layered ESG model that contains 23 nodes excluding opening and closing events namely pseudo-events representing the starting and finishing states of the ESG model.

Current work uses LEMON (Local Expansion via Minimum One Norm) [18] method for the detection of communities in the graph model of GUI. LEMON uses a local expansion method to find the communities. It detects communities using a sparse vector and is able to achieve the highest detection accuracy compared with state-of-the-art methods such as OSLOM [19], DEMON [20], LC [21]. We skip details of the definitions and algorithms for community detection algorithm due to lack of space in the current paper. The community detection algorithm is applied to the non-layered ESG model and results in two sub-layers and one upper layer. The upper layer contains 2 sub-layers (sESG) namely Add and Edit layers given in Fig. 2 and Fig. 3, respectively. The resulting ESGs include 17 events in the Add layer and 6 events in the Edit layer excluding pseudo-events. Instead of applying test generation on 23 events in the non-layered ESG model, the current work divides this non-layered ESG to sub-layers by using a community detection algorithm. The current work is expected to speed up the test generation to save time and potentially scale on large models. Moreover, the resulting test suites from the test generation become more compact and efficient.

V. Discussion

A. Expected results and implications

This section provides the expected results of the proposed approach in terms of scalability and efficiency. To do so, a rough calculation to measure scalability and efficiency will be carried on.

Let's assume that we have 100 events in a non-layered ESG. In the best case, the community detection algorithm detects 10 sub ESG events containing 10 events. Then, we have 10 events in the upper layer and 10 events in each sub layer resulting in 110 events in total. Assume that the test generation algorithm runs on O(n^2) time complexity and O(n^2) space complexity. The test generation algorithm results in 100^2 time units and 100^2 space units in the full resolution ESG model. However, for the layered ESG models with a community detection algorithm applied, the result is 10^2 time and space units for each sub-layer. Since we have 10 sub-layers, we will have 10^2 * 10 which will result in 1000 time and space units and another 10^2 from the upper layer. This results in 1000 + 100 equal 1100 time and space units. This provides about 9 times faster test generation and 9 times more compact test suites for 100 events in the best case if we can divide 100 events into 10 equal sub-layers. Additionally, any further increase in the number of layers will provide much smaller time and space units.

In the worst case, the community detection algorithm does not detect any layer and the non-layered and layered ESG models have the same number of events. However, the cost of the community detection algorithm is neglected from the calculation due to no additional cost on the test generation but to overall methodology. The computational complexity of a conventional community detection algorithm is O(m^2n) for a graph with n vertexes and m edges [12].

B. Threats to validity

1) Conclusion Validity: The size of our case study may be small to show an example of the defined approach in this paper. This is a potential threat to generalize the approach since multiple examples of bigger ESG models should be assessed to be sure about the robustness of the approach. With this, any unforeseeable problems might pop up, and we can address solutions to these possible problems when the total event size of the ESG model goes beyond the numbers given here. We plan to evaluate the proposed approach on medium and large size of case studies to cope with this threat.

2) Internal Validity: We have described how the model-based testing approach works on models which defined as an abstraction of the system. In theory, working with abstractions makes things efficient since we do not need to use a code-based white-box testing approach. This gives us the ability to test huge models otherwise would be time-consuming if we need to test them manually or with code. This may cause a threat to the internal validity of the approach because testing on the abstraction can never be full as it would run on the actual system. However, it is possible to execute generated test suites from models in actual systems using appropriate test automation tools (such as Selenium [2]) for code-based testing.

Another problem might be the problem of creating wrong or missing models (a model does not cover the whole SUT) from

[2]Selenium, Available at: https://www.selenium.dev/

a system if it is a complex one. This will naturally prevent us to test the whole system. Because of these reasons, we must depend on the correctness of the model for an efficient model-based testing approach.

3) External Validity: Model-based testing aims to detect behavioral and functional faults in a system. Using this method to identify problems in visual aspects and semantics of GUI widgets in a system is a threat to the external validity of the approach because model-based testing is not the first method that comes to mind for this. Code-based testing approaches are more applicable for testing these visual aspects of the system under test. However, the proposed approach is applicable for any testing approach that uses behavioral and sequential models rather than concurrent systems modeled such as by Petri-Nets.

4) Construct Validity: The theoretic advantages, discussed in the previous section, require to be validated on the case studies in terms of time and space complexity. However, the scenarios for average and worst cases result in additional cost of community detection algorithm (O(m^2n) for a graph with n vertexes and m edges [12]). This can be a potential threat to construct validity. On the other hand, the community detection algorithm reduces the total cost of time and space complexity comparing to the full resolution model where community detection is not applied.

VI. Conclusion

The study design given in this paper introduces a model-based testing approach combined with the community detection algorithm to cope with possible scalability problems.

A proof of concept with a small size case study is given to exemplify the use of the community detection algorithm in the current work. The community detection algorithm is applied to the full resolution ESG model to create a layered ESG model. The layered ESG model contains several small layers to improve the efficiency of further steps such as; test generation and execution. Moreover, expected results are introduced with rough theoretic calculations.

A community detection algorithm executes on the full resolution ESG model to obtain a layered ESG model. Then, the test sequences will be generated on this resulting layered ESG model. These test sequences will be executed on mutant models obtained from the original ESG model to evaluate the quality of the generated test sequences. To this end, we expect to provide 9 times faster and 9 times more compact test suites in the best case with respect to layered ESG model rather than full resolution ESG model when we assume that time and space complexity of test generation algorithm equal to O($n2$). Moreover, this also provides saving time in further test execution time. Thanks to 9 times more compact test suites, the time required during the test execution phase can be 9 times less. However, the proposed approach comes with an additional cost of the community detection algorithm. It can be noticed that the case study is trivial and presented to explain the idea. However, we plan to evaluate our approach on medium and large sizes of ESG models to assess these expected advantages by automating procedures.

References

[1] R. K. Shehady & D. P. Siewiorek, "A method to automate user interface testing using variable finite state machines," Proceedings of IEEE 27th International Symposium on Fault Tolerant Computing, Seattle, WA, USA, 1997, pp. 80-88, doi: 10.1109/FTCS.1997.614080.

[2] A. M. Memon, M. E. Pollack & M. L. Soffa, "Hierarchical GUI test case generation using automated planning," in IEEE Transactions on Software Engineering, vol. 27, no. 2, pp. 144-155, Feb. 2001, doi: 10.1109/32.908959.

[3] Memon, A. M. (2007). An event-flow model of GUI-based applications for testing. Software testing, verification and reliability, 17(3), 137-157.

[4] Belli, F. (2001, November). Finite state testing and analysis of graphical user interfaces. In Proceedings 12th international symposium on software reliability engineering (pp. 34-43). IEEE.

[5] Belli, F., Beyazıt, M., Budnik, C. J., & Tuglular, T. (2017). Advances in model-based testing of graphical user interfaces. In Advances in Computers (Vol. 107, pp. 219-280). Elsevier.

[6] Kilincceker, O., Turk, E., Challenger, M., & Belli, F. (2018, July). Regular expression based test sequence generation for HDL program validation. In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 585-592). IEEE.

[7] Kilincceker, O., Silistre, A., Challenger, M., & Belli, F. (2019, July). Random test generation from regular expressions for graphical user interface (GUI) testing. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 170-176). IEEE.

[8] Harary, F., & Ross, I. C. (1957). A procedure for clique detection using the group matrix. Sociometry, 20(3), 205-215.

[9] Fortunato, S. (2010). Community detection in graphs. Physics reports, 486(3-5), 75-174.

[10] Leskovec, J., Lang, K. J., & Mahoney, M. (2010, April). Empirical comparison of algorithms for network community detection. In Proceedings of the 19th international conference on World wide web (pp. 631-640).

[11] Sadi, S., Öğüdücü, Ş., & Uyar, A. Ş. (2010, July). An efficient community detection method using parallel clique-finding ants. In IEEE Congress on Evolutionary Computation (pp. 1-7). IEEE.

[12] Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. Scientific reports, 6, 30750.

[13] Mercan, G., Akgündüz, E., Kılınççeker, O., Challenger, M., & Belli, F. (2018). Android uygulaması testi için ideal test ön çalışması. CEUR Workshop Proceedings.

[14] Kılınççeker, O., & Belli, F. (2017). Grafiksel kullanıcı arayüzleri için düzenli ifade bazlı test kapsama kriterleri. CEUR Workshop Proceedings.

[15] Kilincceker, O., Turk, E., Challenger, M., & Belli, F. (2018, April). Applying the Ideal Testing Framework to HDL Programs. In ARCS Workshop 2018; 31th International Conference on Architecture of Computing Systems (pp. 1-6). VDE.

[16] Kilincceker, O., & Belli, F. (2019, November). Towards Uniform Modeling and Holistic Testing of Hardware and Software. In 2019 1st International Informatics and Software Engineering Conference (UBMYK) (pp. 1-6). IEEE.

[17] Festinger, L. (1949). The analysis of sociograms using matrix algebra. Human relations, 2(2), 153-158.

[18] Li, Y., He, K., Bindel, D., & Hopcroft, J. E. (2015, May). Uncovering the small community structure in large networks: A local spectral approach. In Proceedings of the 24th international conference on world wide web (pp. 658-668).

[19] Lancichinetti, A., Radicchi, F., Ramasco, J. J., & Fortunato, S. (2011). Finding statistically significant communities in networks. PloS one, 6(4), e18961.

[20] Coscia, M., Rossetti, G., Giannotti, F., & Pedreschi, D. (2012, August). Demon: a local-first discovery method for overlapping communities. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 615-623).

[21] Ahn, Y. Y., Bagrow, J. P., & Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. nature, 466(7307), 761-764.