Towards an Automated Model-Based Test Generation Approach from Controlled Natural Language Specifications Written in Arabic

Moez Krichen¹, Seifeddine Mechti²
¹ Al-Baha University, KSA & ReDCAD Laboratory, University of Sfax, Tunisia
² MIRACL Laboratory, ISG Tunis, ISSEPS, University of Sfax,
moez.krichen@redcad.org, mechtiseif@gmail.com

Abstract. Model-Based Testing (MBT) technique is a quite interesting tool for evaluating and validating the behaviour of a large class of systems from different fields such as smart cities, internet of things, cloud architectures, web services, telecommunications, transportation, electronic games, etc. This tool is based on strong and sophisticated formal foundations which makes it perfectly appropriate for automated validation processes adoption. In this paper, we propose a new model-based testing approach which takes as input a set of requirements described in Arabic Controlled Natural Language (CNL) which is a subset of the Arabic language generated by a specific grammar. The semantics of the considered requirements is defined using the Case Grammar Theory (CTG). The requirements are translated into Transition Relations which serve as an input for test cases generation tools. We illustrate our approach by a simple case study from the automotive domain.

Keywords: Model-Based Testing (MBT), Arabic Language, Controlled Natural Language (CNL), Case Grammar Theory (CGT), Syntactic Analysis, Automatic Generation.

1 Introduction

Nowadays, testing [18] is a very important phase in the development of any computer or electronic systems for a large variety of systems, software and technologies in different domains and applications such as smart cities, internet of things, cloud architectures, web services, telecommunications, transportation, electronic games, etc. Indeed, these tests should contribute to the detection and elimination of different types of errors and anomalies which could cause enormous damage due to the great complexity and inter-connectivity of the systems used in our daily life. Different types of tests exist such as unit testing [10], integration testing [40], system testing [5], regression testing [13], performance testing [37], security testing [25,23,24], etc. These different test scenarios can be executed either in system development time or at runtime after the system has been put into operation. In general, this testing phase corresponds to a very costly phase in terms of time and resources.
The majority of software requirements specifications are written in natural language, making requirements engineering artifacts a great starting point for the creation of software products.

According to [42] for the automation of the testing procedure, it is possible to identify three important phases:

1. Automatic generation of test scenarios from requirements descriptions;
2. Automatic execution of test scenarios;
3. Automatic valuation of results.

In this work, we focus mainly on the first phase. The model-based testing (MBT) is an efficient methodology for automating these steps. In this case, test scenarios are extracted from behavioral models [11] written using pure formal specification languages or semi-formal languages (such as UML [38]).

Testing practitioners frequently breakdown the system in multiple ways or utilize scenarios to generate test cases for model-based systems, after which models are produced for each scenario. These intermediate models are used to produce the test scenarios. The complexity and time necessary to discover scenarios, the accompanying intermediate models, and, as a result, the generated test cases are among the drawbacks of model-based techniques. Typically, the initial requirements specifications are ideal inputs to begin the testing process since they allow for cost reduction and increased efficacy citeNogueira2014.

Our approach is intended to work as follows. Initially, it should parse the textual system requirements to check whether they conform to the CNL structure. Our CNL is a precise and non-ambiguous subset of the Arabic language. After parsing, we need to provide a semantic interpretation for the requirements [2]. This technique was initially proposed by [8]. From the obtained case frames, the requirements' semantics need to be translated into an internal model representation. Based on this model, we will be able to generate test scenarios using the its SMT solver and the RT-Tester.

Section 2 gives a short overview about the Arabic Language characteristics. Section 3 is dedicated for test generation phase description. Section 4 addresses related work, and Section 5 presents conclusions and future work.

2 Characteristics of the Arabic Language

Arabic [14] is a Semitic language spoken by more than 400 million people across 22 countries in the Middle East and North Africa. With such a number of native speakers, it is considered as the fifth most common spoken worldwide language. The specificities of this language differentiate it from other languages (Indo-European languages) and make it worthy of study. These specificities occur at many levels: morphology, syntax and semantics as illustrated below. The writing and reading systems of Arabic go from right to left and contain in all 28 letters. Concerning the words, they can be divided into three basic types: Nouns, Verbs and Particles . It is based on a trilateral root system, which means that words
(verbs and nouns) are derived from roots of three letters (consonants). Various vowels, prefixes and suffixes are added to the initial roots to form different meanings surrounding that root. Such a system makes it easy to understand the meaning of unseen words by looking at their root.

Moreover, Arabic is known for its morphological richness. For instance, there are up to 1000 synonyms to refer to a camel, 500 for a lion, 52 synonyms for darkness, and 34 for the rain. Arabic, unlike many other languages (Indo-European languages), contains two types of sentences: nominal sentences (or clauses) and verbal sentences (or clauses), the main difference between the two types is the start of the sentence. In a verbal sentence, the sentence starts with a verb which is followed by a subject while a nominal sentence starts with a noun (or a pronoun) which is the subject and is flowed by a predicate (noun, adjective, preposition and noun, or verb).

Fig. 1: The Adopted Grammar for Encoding the CNL Used for Writing the Informal Specifications.

3 Specification Analysis and Test Generation

Our work is mainly inspired by the works of [6,7,44]. In our work, we adopt “Natural Language Processing” (NLP) for parsing every system requirement with respect to our Arabic-CNL (Figure 1). For all valid requirements, the parser produces “corresponding syntax trees” (ST).
A “Controlled Natural Language” is a subset of a given natural language which uses a limited set of grammar rules and a predefined lexicon which contains the vocabulary of the considered application domain. CNLs are mainly considered for avoiding textual ambiguity and complexity. The Arabic-CNL was defined for producing unambiguous requirements in Arabic language. We will adopt the Case Grammar linguistic approach [2] for representing the semantic meaning of the Natural Language. The obtained representation is then translated into a new model the formal semantics of which is represented using a transition relation.

The behavior of the system is modeled by means of state machines, described using “the RT-Tester Internal Model Representation”. For producing test scenarios with appropriate test data, the concurrent state machine semantics is modeled using transition relations which associates pre-states, current time and other variables with post-states. An example of a specification written in Arabic and its translation into English is proposed in Figure 2. The proposed specification conforms to the considered CNL (Arabic-CNL). The decomposition of the considered specification is illustrated in Figure 3.

---

**Fig. 2: Translation of The Considered Specification into English Using Google Online Translator.**
4 Related Work

In [41], the authors presented a technique aiming at model-based test case generation while taking into consideration natural language requirements deliverables. The technique is aided by an application that allows natural language specifications to be automatically translated into Statechart models. In [34], the authors addressed the problem of automatically creating executable security test scenarios from natural language security requirements.

In [9], an ongoing research aiming at developing an appropriate technique for producing natural language descriptions test scenarios using a formal language was proposed. An initial prototype was developed and applied for the case of a project from aerospace field. In [16], the researchers presented a technique which generates MBT test scenarios from NL requirements. These requirements are translated into state charts and test scenarios are derived from these state charts.

In [17], engineers and experts write requirements using structured NL scenarios. These scenarios are then translated automatically into executable test cases which provide backward and forward traceability. In [4], the authors proposed
a tool for analysing requirements. This tool accepts NL requirements, makes a
classification of them, makes interactions with users for refining them, trans-
forms automatically NL requirements into a logical description for validating
them and ultimately it produces test scenarios.

In [43], the authors proposed a methodology for generating test cases which
takes as input a set of restricted NL requirement. This methodology automati-
cally transforms these requirements to executable Petri-Nets. The latters are
adopted as an input for the generation of test cases. In [19], a tool called Kirby
was developed for generating automatically executable test cases from structured
English requirements.

In [15], the authors applied ML techniques for identifying semi-structured re-
quirements descriptions and proposed a rule-based methodology for their trans-
formation into Cause Effect (CE) Graphs. This study demonstrated that the
proposed technique allowed about 85 \% time savings for the creation of the
test models without any quality degradation.

Concerning the Arabic Language, the authors of [12] tried to adopt a formal
techniques to Arabic, as a preliminary phase towards its computation. Similarly
in [1], a formal model for extracting ontological relations for Arabic language was
proposed. To the best of our knowledge, no previous work in the literature was
dedicated for model based testing for specifications written in Arabic language.

5 Conclusion and Future Work

In this work, we have proposed a first attempt for the development of an au-
tomatic test generation methodology from specifications written in Arabic lan-
guage. This work is still in a preliminary phase and should be improved in
different directions.

First, we have to implement a test generation prototype against the consid-
ered specification class to experimentally validate the proposed approach. This
prototype should be tested in different fields and disciplines in order to estimate
its efficiency and its performance.

Moreover our approach may be extended for the case of distributed systems
made of several components [3,30,22,21,20]. In this case, many test generators
and testers will be needed and an optimal coordination [28] between these differ-
ent components will be required. We may also take into account behavioral and
structural adaptations of the considered systems under test [29] which means
how to adapt our test scenarios in these situations.

It will be also useful to consider test cases written with conformance to
TTCN-3 [27] standard (Figure 4) and to cover other types of test scenarios such
as security testing [39] and load testing [26,32,31,33].

Finally, it will be interesting to consider a larger class of Arabic requirements
with less restrictions on the structure of the specifications. This may be achieved
by considering specific machine learning techniques [35,36].
Fig. 4: TTCN-3 Reference Architecture.

References

1. Al Zamil, M.G., Al-Radaideh, Q.: Automatic extraction of ontological relations from arabic text. J. King Saud Univ. Comput. Inf. Sci. 26(4), 462–472 (Dec 2014). https://doi.org/10.1016/j.jksuci.2014.06.007, http://dx.doi.org/10.1016/j.jksuci.2014.06.007
2. Allen, J.: Natural Language Understanding (2Nd Ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA (1995)
3. Benharrat, N., Gaston, C., Hierons, R.M., Lapitre, A., Gall, P.L.: Constraint-based oracles for timed distributed systems. In: Testing Software and Systems - 29th IFIP WG 6.1 International Conference, ICTSS 2017, St. Petersburg, Russia, October 9-11, 2017, Proceedings. pp. 276–292 (2017). https://doi.org/10.1007/978-3-319-67549-7_17, https://doi.org/10.1007/978-3-319-67549-7_17
4. Boddu, R., Lan Guo, Mukhopadhyay, S., Cukic, B.: Retna: from requirements to testing in a natural way. In: Proceedings. 12th IEEE International Requirements Engineering Conference, 2004. pp. 262–271 (2004). https://doi.org/10.1109/ICRE.2004.1335683
5. Borjesson, E., Feldt, R.: Automated system testing using visual gui testing tools: A comparative study in industry. In: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. pp. 350–359. IEEE (2012)
6. Carvalho, G., de Almeida Barros, F., Carvalho, A., Cavalcanti, A., Mota, A., Sampayo, A.: Nat2test tool: From natural language requirements to test cases based on csp. In: SEFM (2015)
7. Carvalho, G., Barros, F., Lapschies, F., Schulze, U., Peleska, J.: Model-based testing from controlled natural language requirements. In: Artho, C., Olveczky, P.C. (eds.) Formal Techniques for Safety-Critical Systems. pp. 19–35. Springer International Publishing, Cham (2014)

8. Carvalho, G., Falcão, D., Barros, F., Sampaio, A., Mota, A., Motta, L., Blackburn, M.: Test case generation from natural language requirements based on scr specifications. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. pp. 1217–1222. SAC ’13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2480362.2480591, http://doi.acm.org/10.1145/2480362.2480591

9. Cristiá, M., Plüss, B.: Generating natural language descriptions of z test cases. In: Proceedings of the 6th International Natural Language Generation Conference. pp. 173–177. INLG ’10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010). http://dl.acm.org/citation.cfm?id=1873738.1873761

10. Daka, E., Fraser, G.: A survey on unit testing practices and problems. In: 2014 IEEE 25th International Symposium on Software Reliability Engineering. pp. 201–211. IEEE (2014)

11. El-Far, I.K., Whittaker, J.A.: Model-Based Software Testing. American Cancer Society (2002). https://doi.org/10.1002/0471028959.sof207, https://onlinelibrary.wiley.com/doi/abs/10.1002/0471028959.sof207

12. El-Sayed, H.: Arabic between formalization and computation. International Journal of Languages, Literature and Linguistics 1, 25–29 (01 2015). https://doi.org/10.7763/IJLLL.2015.V1.6

13. Engström, E., Runeson, P.: A qualitative survey of regression testing practices. In: International Conference on Product Focused Software Process Improvement. pp. 3–16. Springer (2010)

14. Fehri, A.F.: Key features and parameters in Arabic grammar. John Benjamins Publishing Company (2012)

15. Fischbach, J., Junker, M., Vogelsang, A., Freudenstein, D.: Automated generation of test models from semi-structured requirements (2019)

16. George, N.: Model based test case generation from natural language requirements and inconsistency , incompleteness detection in natural language using model-checking approach (2013)

17. Hoisl, B., Sobernig, S., Strembeck, M.: Natural-language scenario descriptions for testing core language models of domain-specific languages. MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development pp. 356–367 (01 2014)

18. Hynninen, T., Kasurinen, J., Knutas, A., Taipale, O.: Software testing: Survey of the industry practices. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 1449–1454. IEEE (2018)

19. Kamalakar, S., Edwards, S., Dao, T.: Automatically generating tests from natural language descriptions of software behavior. ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering pp. 238–245 (01 2013)

20. Krichen, M.: Model-Based Testing for Real-Time Systems. Ph.D. thesis, PhD thesis, Universit Joseph Fourier (December 2007) (2007)

21. Krichen, M.: A formal framework for conformance testing of distributed real-time systems. In: International Conference On Principles Of Distributed Systems. pp. 139–142. Springer (2010)
22. Krichen, M.: A formal framework for black-box conformance testing of distributed real-time systems. International Journal of Critical Computer-Based Systems 3(1/2), 26–43 (2012)
23. Krichen, M., Alroobaea, R.: A new model-based framework for testing security of IoT systems in smart cities using attack trees and price timed automata. In: 14th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2019 (2019)
24. Krichen, M., Cheikhrouhou, O., Lahami, M., Alroobaea, R., Maâlej, A.J.: Towards a model-based testing framework for the security of internet of things for smart city applications. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications. pp. 360–365. Springer, Cham (2017)
25. Krichen, M., Lahami, M., Cheikhrouhou, O., Alroobaea, R., Maâlej, A.J.: Security testing of internet of things for smart city applications: A formal approach. In: Smart Infrastructure and Applications. pp. 629–633. Springer, Cham (2020)
26. Krichen, M., Maâlej, A.J., Lahami, M.: A model-based approach to combine conformance and load tests: an e-health case study. International Journal of Critical Computer-Based Systems 8(3-4), 282–310 (2018)
27. Lahami, M., Fakhfakh, F., Krichen, M., Jmaiel, M.: Towards a ttcn-3 test system for runtime testing of adaptable and distributed systems. In: IFIP International Conference on Testing Software and Systems. pp. 71–86. Springer (2012)
28. Lahami, M., Krichen, M., Bouchakwa, M., Jmaiel, M.: Using knapsack problem model to design a resource aware test architecture for adaptable and distributed systems. In: IFIP International Conference on Testing Software and Systems. pp. 103–118. Springer, Berlin, Heidelberg (2012)
29. Lahami, M., Krichen, M., Jmaiel, M.: Runtime testing approach of structural adaptations for dynamic and distributed systems. International Journal of Computer Applications in Technology 51(4), 259–272 (2015)
30. Lahami, M., Krichen, M., Jmaiel, M.: Safe and efficient runtime testing framework applied in dynamic and distributed systems. Science of Computer Programming 122, 1–28 (2016)
31. Maâlej, A.J., Hamza, M., Krichen, M., Jmaïel, M.: Automated significant load testing for ws-bpel compositions. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. pp. 144–153. IEEE (2013)
32. Maâlej, A.J., Krichen, M.: A model based approach to combine load and functional tests for service oriented architectures. In: VECoS. pp. 123–140 (2016)
33. Maâlej, A.J., Krichen, M., Jmaïel, M.: Model-based conformance testing of ws-bpel compositions. In: 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops. pp. 452–457. IEEE (2012)
34. Mai, P.X., Pastore, F., Goknil, A., Briand, L.C.: A natural language programming approach for requirements-based security testing. In: 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE). pp. 58–69 (Oct 2018). https://doi.org/10.1109/ISSRE.2018.00017
35. Mechti, S., Abbassi, A., Belguith, L.H., Faiz, R.: An empirical method using features combination for arabic native language identification. In: 13th IEEE/ACS International Conference of Computer Systems and Applications, AICCSA 2016, Agadir, Morocco, November 29 - December 2, 2016. pp. 1–5 (2016). https://doi.org/10.1109/AICCSA.2016.7945787, https://doi.org/10.1109/AICCSA.2016.7945787
36. Mechti, S., Khoufi, N., Belguith, L.H.: Improving native language identification model with syntactic features: Case of arabic. In: Intelligent Systems Design and Applications - 18th International Conference on Intelligent Systems Design and Applications, ISDA 2018, Vellore, India, December 6-8, 2018, Volume 2 (2018). https://doi.org/10.1007/978-3-030-16660-1_20, https://doi.org/10.1007/978-3-030-16660-1_20

37. Molyneaux, I.: The art of application performance testing: from strategy to tools. " O'Reilly Media, Inc." (2014)

38. OMG: OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1 (August 2011). http://www.omg.org/spec/UML/2.4.1

39. Ouchani, S., Debbabi, M.: Specification, verification, and quantification of security in model-based systems. Computing 97(7), 691–711 (2015). https://doi.org/10.1007/s00607-015-0445-x, https://doi.org/10.1007/s00607-015-0445-x

40. Jaffar-ur Rehman, M., Jabeen, F., Bertolino, A., Polini, A.: Testing software components for integration: a survey of issues and techniques. Software Testing, Verification and Reliability 17(2), 95–133 (2007)

41. Santiago Júnior, V.A.D., Vijaykumar, N.L.: Generating model-based test cases from natural language requirements for space application software. Software Quality Journal 20(2), 95–133 (Mar 2012). https://doi.org/10.1007/s11219-011-9155-6, http://dx.doi.org/10.1007/s11219-011-9155-6

42. Sarmiento, E., d. P. Leite, J.C.S., Almentero, E.: C l: Generating model based test cases from natural language requirements descriptions. In: 2014 IEEE 1st International Workshop on Requirements Engineering and Testing (RET). pp. 32–38 (Aug 2014). https://doi.org/10.1109/RET.2014.6908677

43. Sarmiento, E., Leite, J.C., Almentero, E., Alzamora, G.S.: Test scenario generation from natural language requirements descriptions based on petri-nets. Electronic Notes in Theoretical Computer Science 329, 123 – 148 (2016). https://doi.org/https://doi.org/10.1016/j.entcs.2016.12.008, http://www.sciencedirect.com/science/article/pii/S1571066116301153, cLEI 2016 - The Latin American Computing Conference

44. Schnelte, M.: Generating test cases for timed systems from controlled natural language specifications. In: 2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement. pp. 348–353 (July 2009). https://doi.org/10.1109/SSIRI.2009.58