A flexible and expressive formalism to specify Metamorphic Properties for BIG DATA systems validation

Fernando Asteasuain[1,2]

[1] Universidad Nacional de Avellaneda, Argentina
fasteasuain@undav.edu.ar
[2] Universidad Abierta Interamericana - Centro de Altos Estudios
CAETI, Argentina

Abstract. BIG DATA systems represent a huge challenge for software engineering validations tasks since they have been classified as "non testable". Metamorphic Relationships (MR) have been proposed as a technique to overcome this problem. These relationships establish interactions between data that can be used to validate the expected behavior of the system. However, the process of exploring and defining MRs is a very arduous one, and an expressive and flexible specification language is needed to denote them. In this work we show how the Feather Weight Visual Scenarios (FVS) framework can be seen as an appealing tool to specify MRs. We exploit FVS features to model complex MR interactions and analysis, allowing the possibility to perform non trivial operations between MRs such as refinement and consistency checking. FVS is shown in action by introducing a proof of concept example focused on a machine learning system over biology cell images.

Keywords: Formal Verification, BIG DATA, Metamorphic Testing

1 Introduction

The term Software Engineering was coined in 1968 during the so-called "Software Crisis". It was born as a response to crucial aspects that were threatening the computing community such as the repetitive failures and delays of software projects. In a few words, Software Engineering constitutes a toolbox of methods, techniques and processes to build and develop quality software. Since its creation, this Software Engineering's toolbox evolved to cope with different paradigms and challenges that arose in the computing field.

Undoubtedly, one of the most relevant domain nowadays relates to BIG DATA, machine learning and data science systems. These kinds of systems feature distinctive characteristics that urge Software Engineering to evolve in order to guarantee the quality of the developed systems [5, 13, 7, 18, 12, 20, 14]. Some of these characteristics are new software architectures, new protocols of communications, new software interactions, stronger performance and availability concerns and volatile, unstructured, diverse and heterogeneous data, just to name

a few. In this Software Engineering evolution the phase of formal verification and validation is probably the one that needs more attention and contributions. According to [13, 9], only two of nearly one hundred analyzed approaches addressing new software engineering methods for big data were related to formal validation. For example, work in [5, 7] introduces a parallel and distributed tool to perform model checking in big data systems. In this sense, one of the most pinpointed items to be addressed for formal validation in BIG DATA is that these kinds of systems have been defined as "non testeable' software [18, 9] because the lack of a proper testing oracle to check their behavior. The problem can be stated as: How the new version of the system, which now includes the analyzed information, can be tested? How can the new system be checked against its expected behavior? How can the expected behavior be specified? How can the software engineer verify if the system is producing the expected outputs?

One solution to tackle this particular item is known as metamorphic testing [8, 19]. This technique is based on building relationships, called metamorphic relationships (MRs), between the data in the original system and the data in the newer version of the system. For example, for a system focusing on sentiment analysis one can build for every word two MRs: one for its synonyms and another for its antonyms. Under this MRs, the new system can be validated as follows: for every word in the original system the newer version must give a similar response for words in the synonyms list and the opposite response for words in the antonyms list. However, how to specify and define the MR's for every system is an extremely arduous and error-prone task. Some approaches say that defining a lot of MRs solves the problem. Nevertheless, others conclude that defining too many MRs could have a negative impact in the validation phase, since they increase the efforts needed to accomplish this task [9].

Employing expressive and flexible specification languages might be useful to properly explore, understand and define MRs. In particular, in this work we explore the FVS (Feather Weight Visual Scenarios) specification framework [2, 4, 3] as a mechanism to specify MRs for BIG DATA systems. FVS is a very simple yet powerful and expressive graphical language to denote the expected behavior of a system. The behavior can be specified using branching or linear properties, and refinement between specifications is also available. FVS specification can be synthesized providing a controller for the system under analysis. When synthesizing behavior, the controller is automatically built upon the expected behavior of the system and the environment it interacts with. Usually, the controller takes the form of an automaton which decides which actions to take based on the received information (mostly provided by external sensors). The controller is built using game theory concepts, obtaining a winning strategy that takes the system to an accepting state no matter which actions the environment chooses [10, 6]. Finally, FVS can be combined with parallel model checkers to cope with BIG DATA systems performance requirements. In summary, in this work we propose the following contributions for FVS as a formal language to specify MRs:

- MRs can be graphically denoted
- FVS expressive power is strong enough to build all the necessary MRs.

- The possibility to synthesize behavior and obtain a controller can be used to check consistency between all the MRs.
- In FVS complex relationships between MRs can be stated. Refinement between two FVS specifications is well established, as well as analyzing when a MR imposes more restrictions over the system than other candidate MR.

As a proof of concept example, we have analyzed a BIG DATA system introduced in [9]. This system proposes a machine learning service to categorize images of biology cells. Complex MRs are defined between images, taking into account features as size, volume and orientation. All of the MRs were defined using FVS, and we exploited FVS characteristics to understand the interactions between all the MRs. The rest of this work is structured as follows. Section 2 briefly presents FVS and explains how a controller can be obtained. Section 3 develops the proof of concept example and presents the obtained results. Section 4 analyzes some related and future work whereas Section5 enumerates the conclusions of this research.

2 Feather weight Visual Scenarios

In this section we will informally describe the standing features of FVS. The reader is referred to [2] for a formal characterization of the language. FVS is a graphical language based on scenarios. Scenarios are partial order of events, consisting of points, which are labeled with a logic formula expressing the possible events occurring at that point, and arrows connecting them. An arrow between two points indicates precedence. For instance, in Figure 1-(a) A-event precedes B-event. In Figure 1-b the scenario captures the very next B-event following an A-event, and not any other B-event. Events labeling an arrow are interpreted as forbidden events between both points. In Figure 1-c A-event precedes B-event such that C-event does not occur between them. Finally, FVS features aliasing between points. Scenario in 1-d indicates that a point labeled with A is also labeled with $A \wedge B$. It is worth noticing that A-event is repeated on the labeling of the second point just because of FVS formal syntaxis.

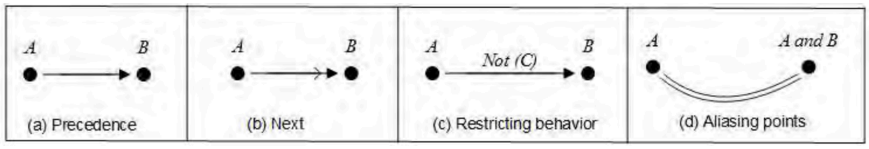

Fig. 1. Basic Elements in FVS

We now introduce the concept of FVS rules, a core concept in the language. The intuition is that whenever a trace "matches" a given antecedent scenario, then it must also match at least one of the consequent ones. In other words,

rules take the form of an implication: an antecedent scenario and one or more consequent scenarios. Graphically, the antecedent is shown in black, and consequent ones in grey. Since a rule can feature more than one consequent, elements which do not belong to the antecedent scenario are numbered to identify the consequent they belong to. An example is shown in Figure 2. The rule describes a requirement for a cooler system. If the cooler is off and the temperature exceeds a certain threshold then two things should happen afterwards: the cooler must be turned on and the observers must be notified.

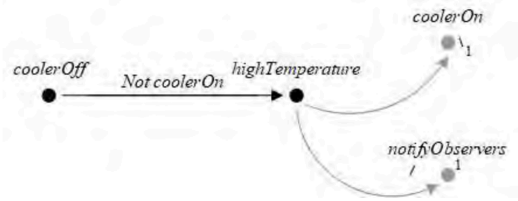

Fig. 2. An FVS rule example

2.1 Behavioral Synthesis in FVS

FVS specifications can be used to automatically obtain a controller employing a classical behavioral synthesis procedure. We now briefly explain how this is achieved while the complete description is available in [4]. Using the tableau algorithm detailed in [2] FVS scenarios are translated into Büchi automata. Then, if the obtained automata is deterministic, then we obtain a controller using a technique [15] based on the specification patterns [11] and the GR(1) subset of LTL. If the automaton is non deterministic, we can obtain a controller anyway. Employing an advanced tool for manipulating diverse kinds of automata named GOAL [21] we translate these automata into Deterministic Rabin automata. Since synthesis algorithms are also incorporated into the GOAL tool using Rabin automata as input, a controller can be obtained.

3 Proof of Concept Example: CMA System

The system under analysis is based on the BIG DATA service implementation described in [9]. This service, called Cell Morphology Assay (CMA) was designed for modeling and analyzing 3D cell morphology and mining morphology patterns extracted from diffraction images of biology cells. Study of 3D morphology can provide rich information about cells that is essential for cell analysis and classification [9]. Relying on different machine learning algorithms and big data tools images are analyzed and explored, and useful scientific information is obtained.

Validation of the system is carried out by defining a set of crucial metamorphic relationships (MR) establishing a proper bond between the original image and the one obtained for validation purposes. These bounds will define whether the system is actually doing a good enough job classifying the biology cells images. In this case, MRs were defined by domain experts users. In what follows, we describe the MR defined in [9] as FVS rules. Finally, a controller for the system is found.

3.1 MRs for the CMA system specified in FVS

The first MR says that if an artificial mitochondrion is added to a stack of original confocal image sections of a cell, then the newly added mitochondrion should be recognized in the new version. Similarly, if an artificial mitochondrion is removed from a stack of the original confocal image sections of a cell, then it must be excluded in the new version. For the FVS specification of this MR we define a set of events such as *image*(standing for the original image), *newImage*(standing for the new version of the image), *artMitoAdded* (standing for the addition of a new artificial mitochondrion), *artMitoRemoved* (a new artificial mitochondrion was removed), *artMitoRecognized* (standing for the recognition of the new artificial mitochondrion) and *artMitoExcluded* (standing for the exclusion of the new artificial mitochondrion). Figure 3 reflects this Inclusion/Exclusion MR.

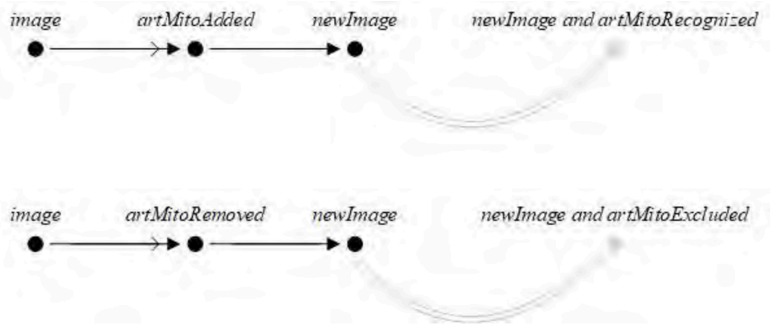

Fig. 3. Inclusion/Exclusion MR as FVS rules

The second MR defines a relationship between size and volume of the images. In a few words, if the size of a mitochondrion in the original image sections is increased then the volume of mitochondria is expected to increase in the new version. The FVS rule for this MR is shown in Figure 4.

The third MR relates lengths between images section. As explained in [9], there is a gap between image sections. This implies that a small mitochondrion may only appear in one section whereas a large one can appear in multiple sections. To specify this MR in FVS we assume that only two sections exist for each image. However, this approach can be easily extended to consider more

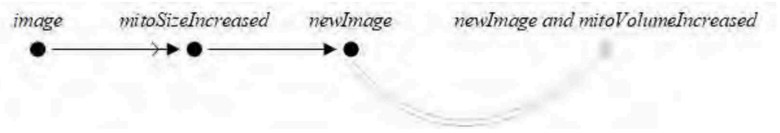

Fig. 4. FVS rule of the Size and Volume MR

sections if necessary. The Lenght-MR is shown in Figure 5. Note that the rule at the bottom of Figure 5 features three consequents: the mitochondrion could be placed in both sections (consequent 1), only in section 1 (consequent 2) or only in section 2 (consequent 3).

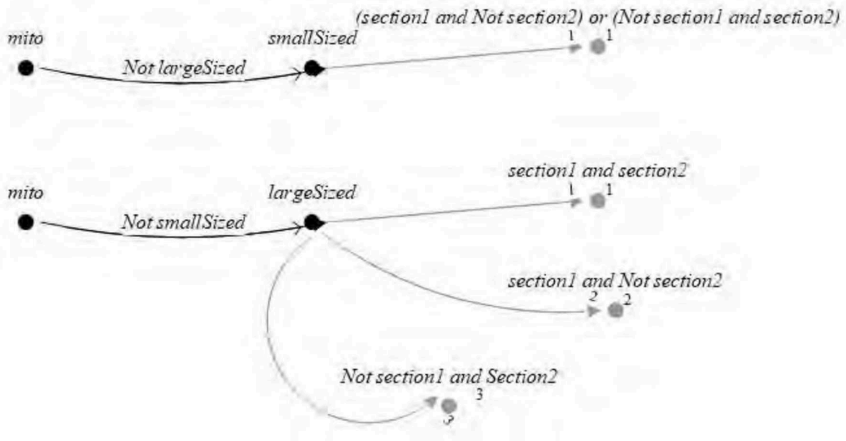

Fig. 5. FVS rule for the Length MR

The fourth MR establishes immutability of the generated shapes. Roughly speaking, the 3D structure of the original one should not be changed in the new version of the image. Considering only two possible shapes, Figure 6 sketches this important MR. As in the previous MR, this specification can be easily extended to consider more than two shapes.

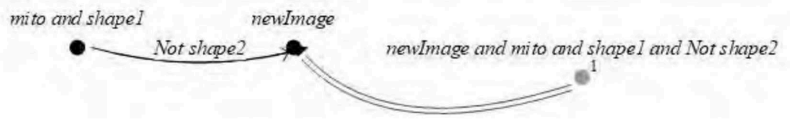

Fig. 6. FVS rule for the immutability MR

The fifth MR focuses on location aspects. Mitochondria that are close to the nuclear should remain in a section near to the nuclear and similarly, mitochondria close to the cell boundary should also appear close to the cell boundary in the new image. Two FVS rules are added to specify this behavior. These are shown in Figure 7.

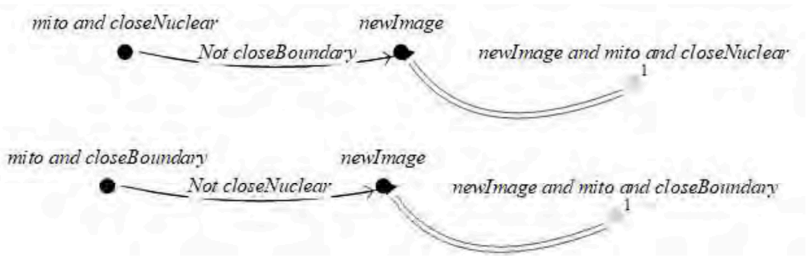

Fig. 7. FVS rules addressing Location MR

MR number six is a simple one. It says that when the size of the sphere in an image becomes larger, the brightness of the texture lines become slimmer. This is reflected in Figure 8.

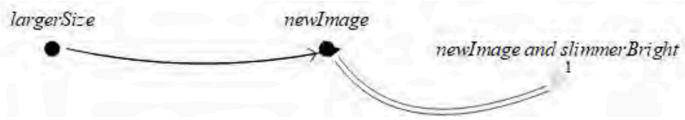

Fig. 8. Size and brightness MR in FVS

Finally, MR number seven establishes that for a sphere shape scatterer the textual pattern should be the same at all orientations. We considered two kinds of scatterers: sphere and irregular ones. The FVS rules tackling this MR are shown in Figure 9.

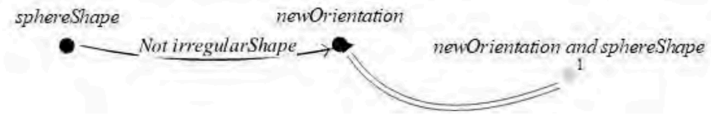

Fig. 9. Sphere shapes MR in FVS

Once all the MR were defined in FVS we were able to obtain a controller as explained in Section 2.1. Since a controller was found we can establish that

there were no inconsistencies in the MRs specifications. A partial view of the controller automaton is shown in Figure 10.

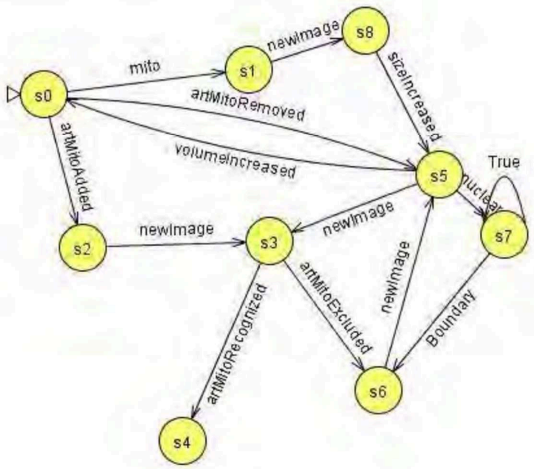

Fig. 10. A controller for the CMA System

3.2 Case Study Analysis, Remarks and Observations

It is relevant to point out that FVS was expressive enough to denote all the MRs defined in [9]. This reflects the richness of FVS's expressive power since this case proof of concept example establishes weighty and meaningful interactions between the images which can be hard to express. In addition, interest analysis of the set of MR can be gathered by analyzing the FVS specification scheme. First of all, visual information relating two or more MR (such as logical subsumption) can be simply noted by visually inspecting the scenarios. For example, both rules in Figure 5 denote equivalent antecedent scenarios but the rule at the bottom holds a "stronger" consequent, since it features more constraints. In consequence, this latter rule can be seen as a specialization of the former one. Secondly, consistency and completeness of the MR set can be stated by the implicit result of whether a controller for the MR specification is found or not. The presence of a controller for the system implies that there were no inconsistencies in all the defined MRs. And finally, other interesting interactions between MRs such as the concept of refinement can also be achieved in the given FVS specification for the system.

A rule featuring multiple consequents as the one shown in the bottom rule of Figure 5 allows the application of the refinement operation between different MR, since a new version with fewer consequents represents a refinement of the original rule [4].

4 Related and Future Work

Previous work in [3] can be seen as a foundation stone for the FVS framework, exhibiting its capabilities to synthesize behavior and to reason about linear an branching behavior together with the formal proofs of soundness and correctness of the approach. In this work we specifically apply FVS to model and verify metamorphic properties in an attempt to formally verify BIG DATA systems.

Work in [9] proposes a very interesting iterative technique to define MRs. Once MRs are defined, more of them can be generated using the notion of refinement between them. The approach was validated against the CMA machine learning system, which is able to categorize a huge amount of biology cells images. We believe FVS graphical flavor could be added to this iterative process to gain power and control to properly define the necessary minimum set of MR for each system. Contrary to FVS, the possibility to check consistency between MRs is not available in this approach.

Other approaches focused on metamorphic properties are [1, 8, 16]. These options are mostly focused on implementation details such as the actual framework tool to deploy the tests. Our approach is addressing a previous phase which is the exploration and specification of MRs.

Regarding future work, we would like to explore the interaction between FVS and other tools. For example, [1] extracts MR in runtime checking the different paths in the execution tree. In this context, FVS scenarios could be used as a monitor in the sense given by the model checking techniques. We also would like to investigate the possibility to automatically generate tests given the set of FVS rules defining the MR. This line of research involves the combination of FVS with automatic code and test generators like [17].

5 Conclusions

In this work we studied FVS as a specification language to denote metamorphic properties. FVS's flexible and expressive notation was able to denote complex MRs behavior. In addition, its formal semantics enables the possibility to perform comparison and formal operations between different MRs such as refinement, or simply comparing which MR imposes more constraints in the expected behavior of the system. We believe these are crucial activities in order to properly define the expected behavior of a BIG DATA system. The results obtained in the proof of concept example are promising enough to consolidate our tool in the formal validation field for BIG DATA systems.

References

1. M. Asrafi, H. Liu, and F.-C. Kuo. On testing effectiveness of metamorphic relations: A case study. In *2011 fifth international conference on secure software integration and reliability improvement*, pages 147–156. IEEE, 2011.
2. F. Asteasuain and V. Braberman. Declaratively building behavior by means of scenario clauses. *Requirements Engineering*, 22(2):239–274, 2017.
3. F. Asteasuain and L. R. Caldeira. A sound and correct formalism to specify, verify and synthesize behavior in big data systems. In *Argentine Congress of Computer Science*, pages 109–123. Springer, 2022.
4. F. Asteasuain, F. Calonge, M. Dubinsky, and P. Gamboa. Open and branching behavioral synthesis with scenario clauses. *CLEI E-JOURNAL*, 24(3), 2021.
5. C. Bellettini, M. Camilli, L. Capra, and M. Monga. Distributed ctl model checking using mapreduce: theory and practice. *CCPE*, 28(11):3025–3041, 2016.
6. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'Ar. Synthesis of reactive (1) designs. 2011.
7. M. Camilli. Formal verification problems in a big data world: towards a mighty synergy. In *ICSE*, pages 638–641, 2014.
8. T. Y. Chen, S. C. Cheung, and S. M. Yiu. Metamorphic testing: a new approach for generating next test cases. *arXiv preprint arXiv:2002.12543*, 2020.
9. J. Ding, D. Zhang, and X.-H. Hu. A framework for ensuring the quality of a big data service. In *2016 SCC*, pages 82–89. IEEE, 2016.
10. N. DIppolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesising non-anomalous event-based controllers for liveness goals. *ACM Tran*, 22(9), 2013.
11. M. Dwyer, M. Avrunin, and M. Corbett. Patterns in property specifications for finite-state verification. In *ICSE*, pages 411–420, 1999.
12. O. Hummel, H. Eichelberger, A. Giloj, D. Werle, and K. Schmid. A collection of software engineering challenges for big data system development. In *SEAA*, pages 362–369. IEEE, 2018.
13. V. D. Kumar and P. Alencar. Software engineering for big data projects: Domains, methodologies and gaps. In *IEEEBIGDATA*, pages 2886–2895. IEEE, 2016.
14. R. Laigner, M. Kalinowski, S. Lifschitz, R. S. Monteiro, and D. de Oliveira. A systematic mapping of software engineering approaches to develop big data systems. In *SEAA*, pages 446–453. IEEE, 2018.
15. S. Maoz and J. O. Ringert. Synthesizing a lego forklift controller in gr (1): A case study. *arXiv preprint arXiv:1602.01172*, 2016.
16. J. Mayer and R. Guderlei. An empirical study on the selection of good metamorphic relations. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 1, pages 475–484. IEEE, 2006.
17. I. A. Niaz and J. Tanaka. Code generation from uml statecharts. In *Proc. 7 th IASTED International Conf. on Software Engineering and Application (SEA 2003), Marina Del Rey*, pages 315–321, 2003.
18. C. E. Otero and A. Peter. Research directions for engineering big data analytics software. *IEEE Intelligent Systems*, 30(1):13–19, 2014.
19. S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on software engineering*, 42(9):805–824, 2016.
20. P. A. Sri and M. Anusha. Big data-survey. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 4(1):74–80, 2016.
21. Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, K.-N. Wu, and W.-C. Chan. Goal: A graphical tool for manipulating büchi automata and temporal formulae. In *TACAS*, pages 466–471. Springer, 2007.