Testing Environmental Models supported by Machine Learning

Valdivino Alexandre de Santiago Júnior
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos, SP, Brazil
valdivino.santiago@inpe.br

Leoni Augusto Romain da Silva
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos, SP, Brazil
augustoromain@gmail.com

Pedro Ribeiro de Andrade Neto
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos, SP, Brazil
pedro.andrade@inpe.br

ABSTRACT

In this paper we present a new methodology, DaOBML, to test environmental models whose outputs are complex artifacts such as images (maps) or plots. Our approach suggests several test data generation techniques (Combinatorial Interaction Testing, Model-Based Testing, Random Testing) and digital image processing methods to drive the creation of Knowledge Bases (KBs). Considering such KBs and Machine Learning (ML) algorithms, a test oracle assigns the verdicts of new test data. Our methodology is supported by a tool and we applied it to models developed via the TerraME product. A controlled experiment was carried out and we conclude that Random Testing is the most feasible test data generation approach for developing the KBs, Artificial Neural Networks present the best performance out of six ML algorithms, and the larger the KB, in terms of size, the better.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; • **Software and its engineering** → **Software testing and debugging**; *Empirical software validation*; • **Applied computing** → *Environmental sciences*;

KEYWORDS

Combinatorial Interaction Testing, Model-Based Testing, Random Testing, Machine Learning, Environmental Modeling, Empirical Software Engineering, Digital Image Processing

ACM Reference Format:
Valdivino Alexandre de Santiago Júnior, Leoni Augusto Romain da Silva, and Pedro Ribeiro de Andrade Neto. 2018. Testing Environmental Models supported by Machine Learning. In *III Brazilian Symposium on Systematic and Automated Software Testing (SAST '18), September 17–21, 2018, SAO CARLOS, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3266003.3266004

1 INTRODUCTION

Environmental modeling is an important activity where models mean coupled nature-society systems in different ways [6]. The motivation for the creation of such models is to understand how human actions impact on natural systems, and they hence support planners and policy makers to propose recommendations for a sustainable development.

In order to develop a software product to support the modeling of nature-society interactions, several steps should be followed such as conception, structuring, calibration, and validation. Such tools usually provide a real-time visualization interface of simulation of complex outputs (e.g. images (maps) or plots). This is the case of TerraME, a product to implement environmental models[1] [6, 12].

The real-time views provided by tools such as TerraME certainly help an expert to determine if there are problems (defects) within the environmental models that created such outputs. However, more complex environmental models may make the expert's manual task to identify if a certain model is defective rather difficult. Thus, Software Engineering, and particularly the software testing process, can contribute a lot towards asserting that an environmental model has defects. And one path to do this is the automation of software testing [9] by analyzing the complex outputs generated by environmental models. It is a hard task to perform test automation, specifically related to the oracle activity, when the evaluated output takes a complex form such as an image [9, 17] but it is a valuable direction to help creating better nature-society interaction models.

Based on this motivation, in this paper we present a new methodology, called Test **Da**ta Generation and **O**racle via Knowledge **B**ase and **M**achine **L**earning (DaOBML), to generate test data and to perform the oracle task for environmental models whose outputs are complex artifacts such as images (maps) or plots. Our approach suggests several test data generation techniques (Combinatorial Interaction Testing (CIT) [3, 4, 22], Model-Based Testing (MBT) [2, 25], Random Testing (RT) [7]) and digital image processing methods to drive the creation of Knowledge Bases. Considering such bases and Machine Learning (ML) algorithms, a test oracle assigns the verdicts of new test input data. Our test oracle proposes six ML algorithms, Artificial Neural Network (ANN), Decision Trees (DT), Support Vector Machine (SVM), K-Nearest Neighbors (K-NN), Random Forests (RF), and Naive Bayes (NB) [23], to decide about the verdicts of new test data submitted to the environmental models[2]. Our methodology is supported by a tool, also called DaOBML, and

SAST '18, September 17–21, 2018, SAO CARLOS, Brazil

ACM ISBN 978-1-4503-6555-0/18/09…$15.00
https://doi.org/10.1145/3266003.3266004

[1] A model is more within the conceptual level while its implementation (source code) can be seen as one way to concretize it. However, in this work, the term "environmental model" refers to the source code of a certain programming language which implements a model developed to study a phenomenon.

[2] Even though Random Forests (RF) also operate by constructing decision trees, we have made a distinction here between traditional solutions, such as the C4.5 algorithm, and RF. In this case, the traditional solutions we call Decision Trees (DT) to differentiate from RF.

we applied it to models developed via the TerraME product. A controlled experiment was carried out to realize about the suitability of the Knowledge Bases created by each test data generation approach to support the test oracle, about the performance of the six ML algorithms, and whether the size of the Knowledge Base influences the performance of the test oracle.

This paper is organized as follows. Section 2 shows an overview of environmental modeling. Our methodology, DaOBML, is in Section 3. Section 4 describes the controlled experiment we conducted. Results and analysis of the empirical evaluation are in Section 5. Section 6 presents relevant related studies, and in Section 7 we present our conclusions and future directions of this research.

2 ENVIRONMENTAL MODELING

Environmental modeling uses mathematical and computational models to represent and investigate the environment. In this paper, we focus on computational models. There are different world views, also called paradigms, to guide the development of environmental models. Given a paradigm, the system under study can then be represented according to the definitions available for that paradigm. The modeler needs to take into account the advantages and the limitations of the available paradigms in order to choose one that best addresses the problem under study.

TerraME is a toolkit to implement models [6, 12]. It is an open-source tool distributed under the GNU LGPL license. In TerraME, models can be implemented using three modeling paradigms: Systems Dynamics (SD), Cellular Automata (CA), and Agent-Based Modeling (ABM). It provides concepts that work as building blocks for model development, allowing the user to specify the spatial, temporal, and behavioral parts of a model.

TerraME uses the Lua language [14] for its programming interface. The modeler can use a clear and expressive language that calls demanding operations in C++, hidden from him. This provides a good trade-off between computational efficiency, coding velocity, and readability. Figure 1 shows the output (a map) of a deforestation model for the Brazilian Amazonia implemented in TerraME.

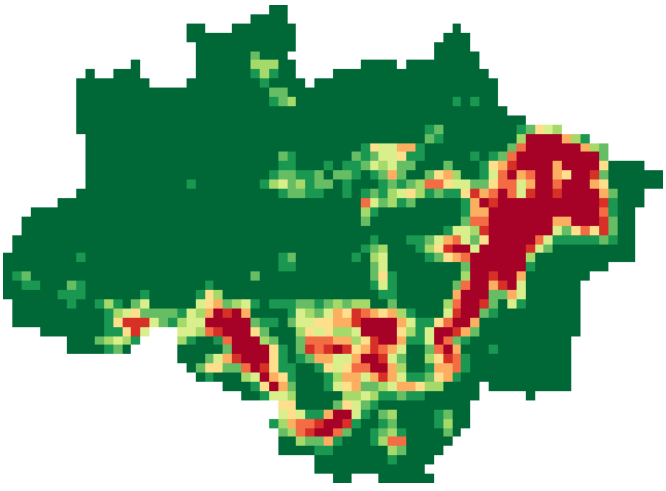


Figure 1: Map of a deforestation model for the Brazilian Amazonia implemented in TerraME

Areas in red in Figure 1 mean a further intensification of deforestation. Thus, one of the main outputs that TerraME generates is a map which is a sort of image, and this is the type of output addressed in this work.

3 THE DAOBML METHODOLOGY

The DaOBML methodology is a black box testing strategy particular suited to system, acceptance, and regression testing levels. Figure 2 presents our methodology. The main idea behind DaOBML is that if an environmental model (source code) has defects, the outputs (usually some sort of image) produced by this model are generally incorrect and we may infer, based on such incorrect outputs, that there are flaws in the source code.

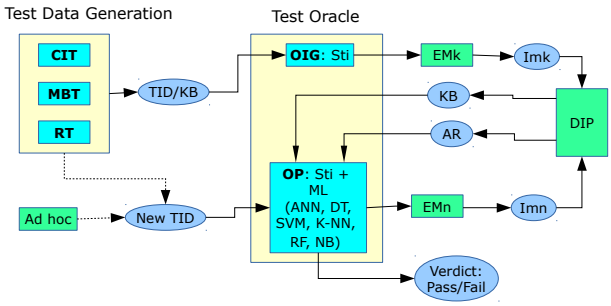


Figure 2: The DaOBML methodology. Caption: TID = Test Input Data; KB = Knowledge Base; OIG = Oracle Information Generator; OP = Oracle Procedure; Sti = Stimulation; EM(k/n) = Environmental Models; Im(k/n) = Images; AR = Actual Results; DIP = Digital Image Processing

We will explain our methodology considering two flows, upper and lower, where the upper flow needs to be carried out before the lower one. In the upper flow, the first step is the test input data generation. We rely on three distinct testing techniques: CIT, MBT, and RT. These techniques can be used in isolation or they can combined. But note that here the idea of applying these test data generation techniques is related to the creation of an appropriate Knowledge Base so that the oracle can assign a coherent verdict (see TID/KB in Figure 2). Thus, at first, test data generation helps the test results evaluation (oracle problem) activity of the testing process.

The context of this situation is where we have a set of environmental models that are sufficient trusted to generate the expected results. In DaOBML, the Knowledge Base (KB in Figure 2) can be indeed considered as the equivalent of the expected results. Such trusted models can be denoted as "gold standard" models. Hence, we need to test new versions of such models with more features or even other different environmental models but which have relatively similar characteristics of the ones which created the Knowledge Base. We assume that we have not enough images (maps) in the Knowledge Base or we have no base at all. Note that this situation of insufficient or even not existence of a set of appropriate images is very likely to occur when we are dealing with new research environmental projects or even with software developed for new medical equipments.

Thus, the reasoning is to drive the creation, via a systematic procedure (test generation strategy), of a Knowledge Base to support the test oracle operation. Moreover, the idea of three distinct testing techniques is to perceive that eventually for certain types of environmental models, for instance, a CIT-based approach can generate a more suitable Knowledge Base compared with MBT and RT-based ones.

Once the test input data are created, the Oracle Information Generator (OIG) component of the test oracle then stimulates (Sti) a set of Environmental Models (EMk) with these test data in order to, at the end, generate the expected results (Knowledge Base). In the case of environmental modeling software, the outputs are in many cases graphical outcomes or images (Imk) such as two-dimensional maps and plots. However, such images need to be processed so that their attributes can be gathered and hence helping the test oracle to assign the correct verdicts via ML approaches.

Note that the outputs of the Digital Image Processing (DIP) method are a set of transformed images. But, note that each entry of the Knowledge Base can be a transformed image itself but it can also be only the features extracted from the transformed image [9]. In the latter case, the Knowledge Base demands less storage space. Hence, when we mention an "entry" (i.e. a record) of the Knowledge Base, this means an output of the environmental modeling tool (image, map) which can be represented as an image itself or only as its features.

In the lower flow, new test input data (New TID) need to be submitted to the environmental models. Note that DaOBML does not suggest any particular test data generation technique for these new test input data: the professional may use an ad hoc strategy or even to rely on one of the techniques (CIT, MBT, RT) that were used to generate test input data to support the development of the Knowledge Base. The dashed arrows in Figure 2 mean this set of possibilities a test designer may have at his/her hand.

The Oracle Procedure (OP) of the oracle then stimulates the Environmental Models (EMn) with these new test input data in order to produce another set of images (Imn) which are also processed by the DIP method to obtain the Actual Results (AR). The same remark we have just made about the Knowledge Base, where each record can be an image or only its features, applies. The environmental models considered here (EMn) are usually different from the ones used in the upper flow (EMk). However there are circumstances where it makes sense that EMn = EMk, when we want to evaluate the capability of the methodology, particularly related to the strengths of the ML algorithms, to return the correct class (environmental model), since the new test input data are completely different from those that were used to generate the Knowledge Base.

With these two artifacts, KB and AR, the OP component uses one out of six ML algorithms, as shown in Figure 2 (ANN, DT, SVM, K-NN, RF, NB), to decide about the verdict of the new test data. We selected such algorithms given their wide acceptance in the academic community as classifiers. The verdict (Pass/Fail) is given by the OP component via a particular ML approach which checks if a given transformed image (e.g. a map), i_n, derived via new test input data matches some image, i_k, within the Knowledge Base. An environmental model passes a test if such a match is found, i.e. $i_n \sim i_k$. Otherwise, we say that the test fails and the environmental model is defective. Naturally, this "matching" does not mean a complete equality between the two images, i_n and i_k, since we are exploring the ability of the classifiers (ML algorithms) to predict the correct class due to new test input data as it has been done for so long within data mining/pattern recognition applications.

3.1 Implementation

We have implemented a tool, also called DaOBML, to partially support our methodology. The tool automatically executes the set of test input data suggested by a particular technique (CIT, MBT, RT) for driving the generation of the images (maps) that will compose the Knowledge Base. Our tool implements some steps of the digital image processing such as the conversion of color images into grayscale, and the use of the Sobel operator as edge detector. For the conversion into grayscale, we transform a color image (map) into an image with 16 shades of gray (4-bit grayscale). Note that even though 4-bit grayscale is considered an old color depth system, it is still adequate for the type of image produced by the environmental modeling software we selected (TerraME).

The Sobel operator is a gradient-based edge detection method which encounters edges using a horizontal mask and a vertical mask. Considering that convolution is the process of multiplying each intensity value of an image with its local neighbors, weighted by the mask, in the Sobel operator an image is scanned from left to right and top to bottom of the image using the horizontal and vertical masks, separately [19]. In the current version of the DaOBML tool, all images that are input to the OP component of the test oracle, the ones within KB and AR, have their edges detected. To help in the digital image processing steps, we have taken into account the JavaCV library [24].

Regarding the six ML algorithms suggested by our methodology, we have embedded the Weka data mining software [31] within our tool so that the user does not need to call it separately in order to know the verdict of the new test data. Moreover, note that each map produced by TerraME has only its features, after digitally processed, stored into the Knowledge Base in the main input format of the Weka software (`arff`). The images (maps) are created and after processed (edges detected) they are discarded. The bottom of line is that, in the current implementation of DaOBML, a Knowledge Base is indeed a single `arff` file where we concatenate all the partial contributions of the test input data suggested by the testing generation techniques. This single `arff` file is automatically created by our tool.

4 EMPIRICAL EVALUATION

In this section, we present the description of an empirical evaluation to assess several points of our methodology. In fact, this empirical evaluation is classified as a controlled experiment [32].

4.1 Objective and Definitions

The objective of this assessment is to evaluate several aspects related to DaOBML. First, we want to realize how suitable are the Knowledge Bases created by the test data generation approaches so that the test oracle can assign True Positive verdicts. For CIT, we selected the T-Tuple Reallocation (TTR) greedy algorithm [3, 4] and the Advanced Combinatorial Testing System (ACTS) tool [5] configured with the In-Parameter-Order General (IPOG) greedy

algorithm. For MBT, we relied on the Hierarchy-based translation from Statecharts into Model Checking and Specification Patterns Properties for Testing (HiMoST) [25] method which generates software test cases via Model Checking. But, in this case, we started right from the model of the NuSMV Model Checker and not from the Statechart model. We considered pure RT and not its adaptive variation.

Other issue is the performance of the ML algorithms, i.e. which out of the six solutions (ANN, DT, K-NN, NB, RF, and SVM) presents more correct verdicts. This is a very important goal of this evaluation since providing recommendations of approaches to practitioners is quite advisable.

Any strategy supported by a Knowledge Base is naturally influenced by such base. Hence, we wonder if the size of the Knowledge Base influences the performance of the test oracle. We created two types of bases: one smaller and one larger. For each type of base, we combined this possibility with the test data generation techniques resulting in 9 different Knowledge Bases as shown in Table 1. Note that the MBT technique created a single smaller base while ALLT means that the base was created by combining the maps of all the previous approaches: CIT/TTR + CIT/ACTS + MBT + RT. In this case, the single MBT base was considered for creating both the ALLT smaller and ALLT larger bases. We checked for duplicates of maps when combining the bases to form the ALLT Knowledge Bases. The size of the smaller base is around 25 entries and the larger one is around 250 entries. These numbers are not equal for all the approaches because of the characteristics of the test generation techniques and the environmental models we selected.

Table 1: Knowledge Bases. Caption: TDG = Test Data Generation

TDG	Smaller	Larger
ALLT	✓	✓
CIT/TTR	✓	✓
CIT/ACTS	✓	✓
MBT	✓	
RT	✓	✓

Regarding the metrics to assess the accuracy of our strategy, we considered the True Positives, i.e. the correctly classified instances. Our set of samples is composed of the following models (programs) of the TerraME 2.0-RC5 product:

(1) 13 models of the CA package: `Anneal`, `Banded Vegetation`, `Fire`, `Growth`, `Life`, `Oscillator`, `Parasit`, `Snow`, `Parity`, `Interspecific Competition`, `Excitable`, `Wolfram`, and `Solid Diffusion`;
(2) 9 models of the ABM package: `Growing Society`, `Heat Bugs`, `Labyrinth`, `Life Cycle`, `Overpopulation`, `Single Agent`, `Sugarscape`, `Schelling`, and `Predator Prey`.

As a matter of example, Figure 3 shows one output (map) of the `Interspecific Competition` model of the CA package stored in the RT smaller base. This model shows how species juxtaposition in space can lead to different population dynamics among competitor species [29]. This map is the final result of the simulation where we can see the competitive interaction of five grass species: *Agrostis stolonifera* (orange cells), *Holcus lanatus* (dark green cells), *Cynosurus cristatus* (dark blue cells), *Poa trivialis* (light blue cells), and *Lolium perenne* (dark red cells).

Figure 3: One output map of the `Interspecific Competition` environmental model stored in the RT smaller base

4.2 Research Questions and Variables

We defined the following Research Questions (RQs) to answer:

(1) **RQ_1** - How suitable are the Knowledge Bases created by each test data generation approach suggested by DaOBML to support the test oracle: ALLT, CIT, MBT, and RT?
(2) **RQ_2** - Which of the ML algorithms (ANN, DT, K-NN, NB, RF, SVM) presents the best performance?
(3) **RQ_3** - Does the size of the Knowledge Base influence the performance of the test oracle?

The independent variables are the test data generation techniques, the ML algorithms, and the size of the Knowledge Bases. The dependent variable is the number of True Positives obtained after applying the DaOBML methodology. In this evaluation, a True Positive means a Pass verdict issued by the test oracle.

4.3 Description of the Experiment

We started by generating the test input data in order to create the 9 Knowledge Bases (see Table 1). For both CIT approaches, CIT/TTR and CIT/ACTS, we considered 2 values per parameter for the smaller base and 4 values per parameter for the larger base. In both cases, we have the strength (t) equals to 2. For instance, let us consider the `Banded Vegetation` [10] model of the CA package. One parameter of this model is `dryCoeff`, a coefficient between 1.2 and 3.5 to change the state of a cell to dry. Hence, we selected two values of this parameter for creating the smaller bases due to TTR

and ACTS, and 4 values for the larger bases. This model has other parameters (`plantCover`, `wetCoeff`, etc.) in which we followed the same reasoning. Each combination of values produced by a CIT algorithm, where each parameter contributes with one value, is input to an environmental model and thus this combination results in one map which will be stored into the Knowledge Base, after detecting the edges. Since the outputs of both algorithms are not deterministic, the precise size of the bases (smaller, larger) is not known a priori.

As we have mentioned in Section 4.1, the HiMoST method [25] was the one used as MBT strategy. For each sample (environmental model), we developed a model directly in the notation of the NuSMV Model Checker rather than starting with a Statechart model. We followed the guidelines for properties formalization in Computation Tree Logic (CTL) [8] and test case generation proposed in HiMoST. We selected the Absence Pattern and Global Scope of the Specification Patterns System proposed by Dwyer et al. [11]. With respect to RT, the choice of values of the parameters was completely random. Moreover, we selected pure RT and not Adaptive RT [7].

It is important to state that not all models of the CA and ABM packages are suitable so that we could generate test input data for them. For instance, to generate test input data via the CIT algorithms with $t = 2$, the environmental model must have at least 3 free parameters so that we can assign different values (2 or 4) to them. If a model does not meet this requirement, hence we were not able to generate test data via these approaches.

After developing all 9 Knowledge Bases, the next activity was to select new test data where we considered 25 new test input data in accordance with an Ad hoc approach (see Figure 2) to submit to each environmental model. However, these 25 sets of values were determined for each model where it was possible to generate test input data according to a certain testing technique. Such new test input data are completely different from the ones suggested by each testing approach to drive the creation of the Knowledge Bases. For instance, we could generate test data for the `Banded Vegetation` model taken into account all 9 configurations of test data generation techniques and size of Knowledge Bases. Thus, we determined 25 new sets of values, which generated 25 maps, for the `Banded Vegetation` model and these values are not equal to any of the set of values suggested by CIT, MBT, and RT.

Next, the ML algorithms (ANN, DT, K-NN, NB, RF, SVM) came into play to give the verdicts of the new test data. Note that here DaOBML suggests the adoption of a single ML algorithm and not a combination of them. The main reasoning of the verdict assignment is based on the "adequate" matching of the map due to new data with some map of the Knowledge Base. Moreover, we considered the default configuration of the classifiers within the Weka software in most cases but we emphasize these points below:

(1) ANN. We used the Multilayer Perceptron with the backpropagation algorithm for training;
(2) DT. We chose the C4.5 algorithm (J48 classifier);
(3) K-NN. We used the IBk classifier configured with 3 neighbors;
(4) SVM. We selected the LibSVM library with a linear kernel.

As we have previously mentioned, a True Positive is indeed a Pass verdict of a particular new test input data. The assignment of True Positives is as follows. Let us consider the RT larger base where all 22 samples created entries in this base. Let us consider an environmental model, e. g. `Fire`, and one new set of input data (values) for this model that will generate a new map, m_{nf}. Hence, we expected that every ML algorithm would classify this m_{nf} as `Fire` and not as, for example, `Banded Vegetation`. To assert that an ML algorithm classified m_{nf} as `Fire`, we looked at the highest percentage of correctly classified instances as shown in Weka's output. Hence, we expected that `Fire` was the class with the highest percentage of correctly classified instances among all 22 environmental models. In this case, we have a True Positive. Otherwise, we have a False Negative which conversely represents a Fail verdict. As a matter of illustration, in the RT larger base, the RF algorithm had only 11 True Positives out of the 25 new maps for the `Fire` model, and thus it presented 14 False Negatives. On the other hand, the ANN algorithm presented 21 True Positives ouf of 25, and 4 False Negatives.

Another remark about the verdict assignment is that some approaches that use features to help in the similarity analysis between images [9, 16, 21] adopt a threshold parameter, which indicates the maximum distance accepted to consider two images similar. We do not have this threshold parameter to tune because, as we have just said above, we give the verdict of the new test data by a relative comparison among the percentage of correctly classified instances of the ML algorithms. We decided to do this because one of the main questions we would like to answer is about the performance of the classifiers (ML algorithms), measured as giving the correct (Pass) verdicts within DaOBML, to give us indication of which out of the six ML approaches is better for further developments of our methodology.

We focused our data analysis considering only True Positives because, in our context, making analysis with False Negatives, as others have done [16], would not give us significant information. Since we are using the same environmental models to receive the new test input data as the ones for creating the Knowledge Bases, hence we expected that every ML algorithm within the test oracle would produce as verdict Pass (True Positive) for all new test input data. Moreover we decided to do this, same environmental models for creating the Knowledge Bases and to receive new and different test input data (EMk = EMn in Figure 2), again because knowing about the performance of the ML algorithms is one of the main issues we would like to answer. Since we have 25 new sets of values, we always have this situation:

$$FN = 25 - TP,$$

where FN = number of False Negatives, and TP = number of True Positives. Statistically speaking, it is enough to accomplish the analysis with True Positives where the higher its value (median, mean), the better.

Regarding RQ_1, in order to obtain the number of True Positives for each environmental model, we used the following formula:

$$em_f = (\sum_{p=1}^{b} \sum_{q=1}^{m} TP_{pq})/b,$$

where em_f is an environmental model identified by f (in our case, here we have $1 \leq f \leq 15$), p identifies a base related to a particular

test data generation technique where the maximum number of bases is b (here we have $b = 2, 4, 1, 2$ for ALLT, CIT, MBT, and RT, respectively), q is an ML algorithm and the maximum number of algorithms is m (in our case, $m = 6$), and TP_{pq} is the number of True Positives due to a particular ML q. Note that the maximum value of True Positives is 150 (25×6) for each sample.

Regarding RQ_2, the value of True Positives for an environmental model, em_f, is simply the sum of True Positives due to each ML algorithm. In this case, we were allowed to consider all 22 models, and since we have 9 Knowledge Bases, the maximum value of True Positives is 225 (25×9) for each sample. For answering RQ_3, we performed a pairwise comparison between the smaller and larger bases of each test strategy where here ACTS and TTR were considered in isolation, and took into account 15 models (CIT/ACTS, CIT/TTR) and all 22 models (RT, ALLT). Recall that, depending on the test data generation strategy and the model, we could not generate test input data. Here, the maximum value of True Positives is 150 (25×6) for each environmental model.

To properly answer all the RQs, we made use of appropriate statistical evaluation. For RQ_1 and RQ_2, we have more than two populations and thus we used the Friedman test plus the exact all-pairs comparisons post-hoc test [13] with Bonferroni and Holm p-value adjustment methods. For RQ_3, we performed a pairwise comparison between smaller and larger bases. Thus, we relied on the two-sided Wilcoxon signed-rank test or on its two-sided Asymptotic variation, in case of ties. In all cases, we defined the significance level $\alpha = 0.01$.

4.4 Validity

One of the threats to the conclusion validity is the reliability of the measures. Our results are associated with six tools/libraries: DaOBML, Weka (embedded within DaOBML), JavaCV (embedded within DaOBML), TTR, ACTS, and NuSMV. After all processing steps, the measures were automatically obtained in accordance with the ML algorithms of the Weka tool. We believe that replication of this study by other researchers will produce similar results and our study has a high conclusion validity.

The samples of our experiment were environmental models, coded in the TerraME's modeling language, and thus we neither had any human/nature/social factor nor unanticipated events to interrupt the collection of the measures once started to pose an internal validity. Hence, our controlled experiment has a high internal validity.

We have a threat to external validity related to the population, i.e. how significant is the set of samples used in the experiment. Overall, we investigated 22 environmental models of two packages of the TerraME product. These models serve as examples of how to use the several features of the TerraME's language and thus it is necessary to evaluate more environmental models to generalize our results. But, we believe that the results of this controlled experiment are interesting as discussed in Section 5.

5 RESULTS AND ANALYSIS

We now present and discuss the results of the empirical evaluation we conducted. Recall that we considered True Positives (Pass verdicts) as explained in Section 4.3. Data for this evaluation are available in [26].

5.1 RQ_1

Regarding the first RQ which aims to answer how suitable are the Knowledge Bases created by each test data generation approach to help the test oracle, we had four classes of bases: ALLT, CIT, MBT, and RT. The result of the Friedman test presented a p-value = 0.0007067, refuting the null hypothesis of equal population distributions. Results (p-values) of the exact all-pairs comparisons post-hoc test with Bonferroni and Holm p-value adjustment methods are shown in Table 2. Figures 4a and 4b present the boxplot and the meansplot, respectively.

Table 2: RQ_1 - Exact all-pairs comparisons post-hoc test with Bonferroni and Holm p-value adjustment methods

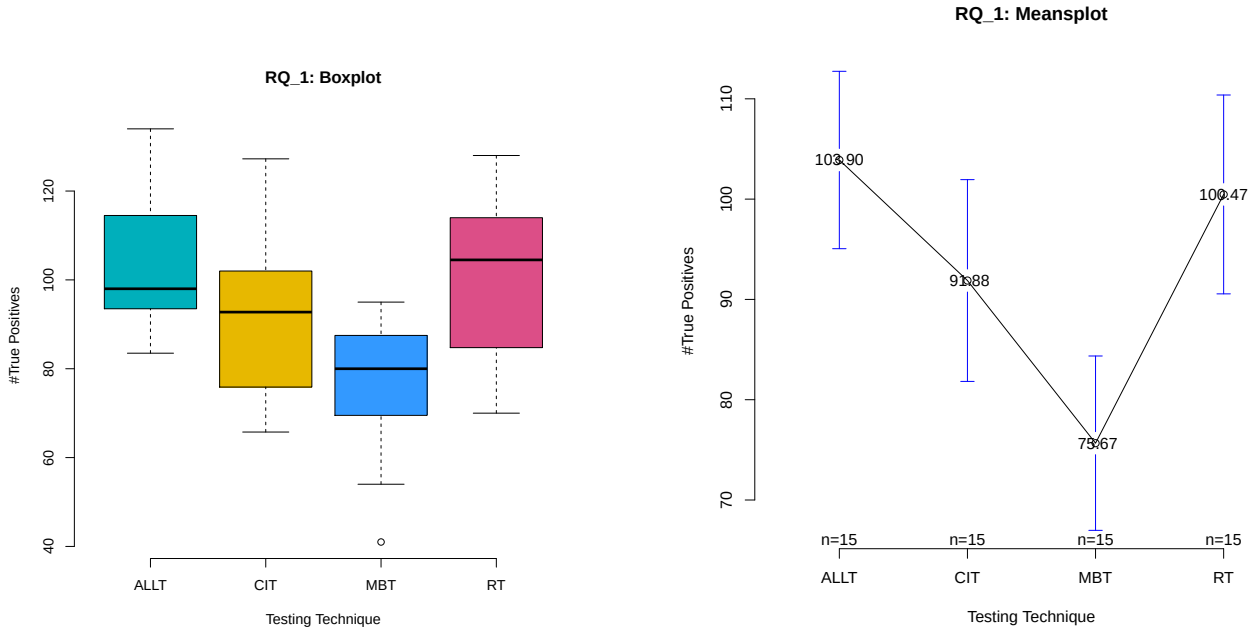
Comparison	Bonferroni	Holm
CIT × ALLT	0.83025	0.41513
MBT × ALLT	0.00051	0.00051
RT × ALLT	1.00000	0.72257
MBT × CIT	0.16312	0.10875
RT × CIT	1.00000	0.72257
RT × MBT	0.00628	0.00524

As presented in red in Table 2, in two cases the null hypothesis is rejected and there is difference between the populations: MBT × ALLT and RT × MBT. This happens with both adjustment methods, Bonferroni and Holm. As we can see in Figures 4a and 4b, the MBT technique is the worst technique (lower median and mean) in the pairwise comparison with ALLT and RT. Since there is no difference between any other pairs of techniques (including RT × ALLT) and in order to create the ALLT base is necessary to apply all the three testing techniques requiring more effort, we conclude that RT is the most suitable approach for creating a Knowledge Base to help the test oracle. According to this conclusion, the professional may simply select a set of random values for the parameters which is a more straightforward approach.

Despite criticisms of pure RT as we used, such as the next test cases/data to be selected can be less evenly distributed over the input domain because it does not make use of knowledge of previously executed test cases/data [7, 27], pure RT was the best solution in our evaluation. The simplicity of pure RT is one of its advantages for practical purposes.

5.2 RQ_2

To answer the question of which of the ML algorithms presents the best performance, first we need to realize that we have six classes, corresponding to the six ML algorithms that the DaOBML proposes: ANN, DT, K-NN, NB, RF, and SVM. Repeating the same procedure described in Section 4.3, the Friedman test resulted in a p-value = 2.442e-12, again rejecting the null hypothesis. Results (p-values) of the exact all-pairs comparisons post-hoc test with Bonferroni and Holm p-value adjustment methods are shown in Table 3. Figures 5a and 5b present the boxplot and the meansplot, respectively.



(a) Bloxpot **(b) Meansplot**

Figure 4: RQ_1 - Boxplot and Meansplot

Table 3: RQ_2 - Exact all-pairs comparisons post-hoc test with Bonferroni and Holm p-value adjustment methods

Comparison	Bonferroni	Holm
DT × ANN	6.8e-13	6.8e-13
K-NN × ANN	0.0136	0.0091
NB × ANN	2.1e-09	1.8e-09
RF × ANN	1.0e-09	9.6e-10
SVM × ANN	0.0013	0.0010
K-NN × DT	0.0052	0.0038
NB × DT	1.0000	1.0000
RF × DT	1.0000	1.0000
SVM × DT	0.0574	0.0344
NB × K-NN	0.1242	0.0580
RF × K-NN	0.0747	0.0399
SVM × K-NN	1.0000	1.0000
RF × NB	1.0000	1.0000
SVM × NB	0.5903	0.1968
SVM × RF	0.4820	0.1928

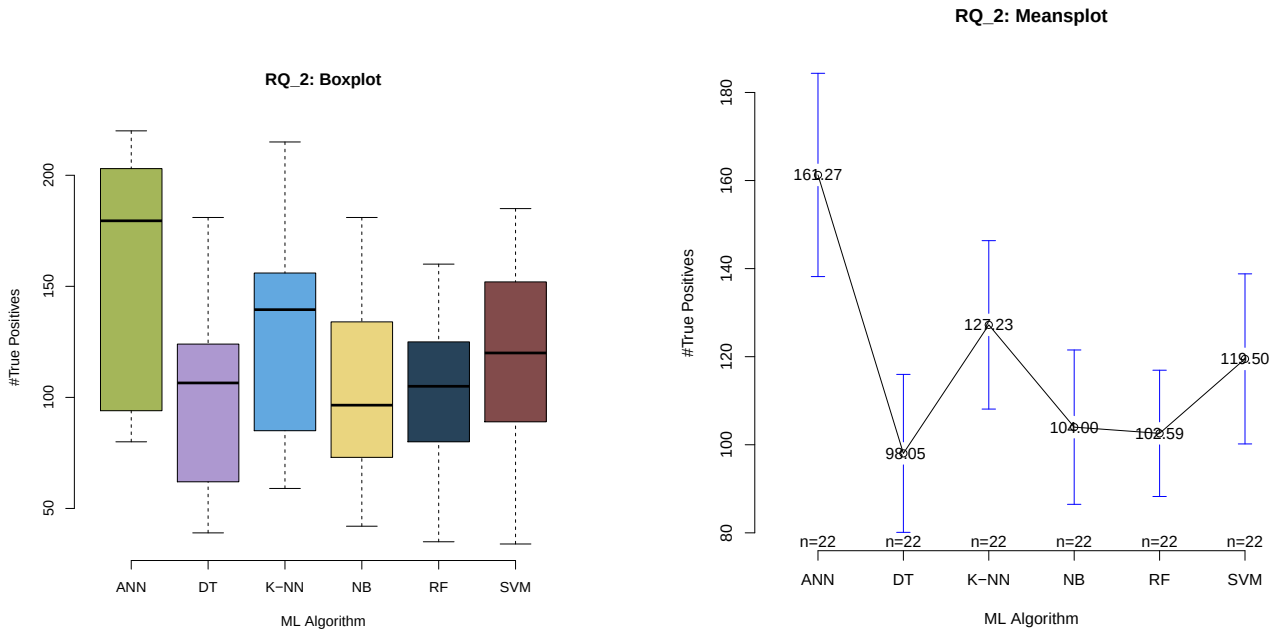
In Table 3, we see that there is significant statistical difference when comparing ANN with all the other ML approaches with the Holm adjustment method. Considering the Bonferroni method, only the comparison K-NN × ANN accepted the null hypothesis (there is no difference) while all the other pairwise evaluations involving the ANN algorithm presented differences. Even so, in this case, the adjusted p-value is a little bit superior (0.0136) than the significance level (0.01). Regarding the other pairwise comparisons, only K-NN × DT presented significant statistical difference.

Median and mean of the ANN are the highest and K-NN is in the second place as shown in Figures 5a and 5b. Based on these facts, we can state that ANN is the ML algorithm that presented the best performance overall. As we have mentioned, we used as ANN the classic Multilayer Perceptron with the backpropagation algorithm for training. This classic ANN has been proving successful in several application domains due to its adaptive learning, ability to deal with complicated or imprecise data, among other points. It is one of the most well known classifier and, for this experiment, proved to be the best solution.

5.3 RQ_3

As we have stated in Section 4.3, in order to answer this question we performed a pairwise comparison between the smaller and larger bases of each test strategy where here ACTS and TTR were considered in isolation. Therefore, we accomplished four comparisons with both types of bases: CIT/ACTS, CIT/TTR, RT, and ALLT. We detected ties in all four pairwise evaluations and hence the two-sided Asymptotic Wilcoxon signed-rank test was selected. Table 4 presents the p-values.

We noticed that only in the ACTS case there is no difference between the populations. In all the other three comparisons, the Wilcoxon test determined that there are differences. In all cases, the larger base due to a certain testing technique has always the highest mean and median compared with the respective smaller base. Hence, we conclude that as larger the Knowledge Base, in terms of size,



(a) Bloxpot **(b) Meansplot**

Figure 5: RQ_2 - Boxplot and Meansplot

Table 4: RQ_3 - Asymptotic Wilcoxon signed-rank test

Comparison	p-value
CIT/ACTS: smaller × larger	0.07007
CIT/TTR: smaller × larger	6.104e-05
RT: smaller × larger	5.484e-05
ALLT: smaller × larger	9.537e-07

the better. Although this conclusion may be regarded as obvious, it is important to stress that in other rigorous comparisons [20], working with complex output systems, the size of the bases was not that important. Thus, we decided to investigate in our context if the size of the bases was indeed relevant, and it ended up happening.

6 RELATED WORK

In this section, we present some recent and relevant studies related to our research. VISOR is an automated test oracle that uses an image processing pipeline which includes a series of image filters that align the compared images and remove noise to eliminate differences caused by scaling and translation [17]. The context of their research is the usage of captured images of Digital TV screens, which are prone to illumination, translation, scaling variations, and noise. Their approach is more sophisticated than ours but note that scaling is not a problem in our context because we work with outputs with the same pixel resolution, and zooming does not make much sense here. It is very probable that illumination is not also something to be concerned because lighting is basically the same (we are not capturing images via a camera). Thus, we were able to develop a simpler and less costly strategy compared with their method.

Content-based image retrieval has been employed to address the oracle problem for systems which produce complex outputs such as image or audio [9, 16, 20, 21]. The authors named their approach as feature-based test oracle which relies on the exploitation of features extracted from reliable outputs considered as a reference. They then measure the similarities between the outputs produced by an application being tested and the reference base. One drawback of their method is that it is required to tune a threshold to decide whether two images are similar. The calculation of this threshold is not a trivial task in practice which may demand considerable effort from the professional. Our approach does not have a threshold parameter to tune. Moreover, they have been using only Euclidean distance as similarity measure although many other solutions are available and seem more appropriate such as the Bin Overlapped Similarity Measure [28].

The effectiveness of Metamorphic Testing (MT) in detecting faulty Sobel edge detection programs was studied in [30]. MT is a way to generate new test input data and detect faults in applications even if the correct expected output is unknown, i.e. when we have no oracle available. The authors used camera captured images from published image libraries as test inputs. Approaches that are based on MT suffer the problem of identifying the metamorphic relations. These relations can be very complicated even for simple programs. On the other hand, we exploit the capabilities of ML algorithms as classifiers and their strengths to infer conclusions based on complex data.

In the automotive industry domain, image processing and visual testing techniques have been used for test oracle automation [1]. The authors addressed the Model-In-the-Loop (MIL) step of development of the Electronic Vehicle Information Center (EVIC), a component which has an interactive display to provide information to the driver such as fuel consumption. Hence, they took snapshots of the interactive display at certain times during test execution. Test results evaluation is accomplished via techniques such as coefficient of correlation between the actual and a reference image, optical character recognition (text), or other custom visual feature extraction for more complex items. We used the Sobel operator to detect the edges of the actual and reference images and not coefficient of correlation. Moreover, we relied on ML algorithms to give the verdict of the test cases.

In the context of Web applications, test oracle automation has been much approached although most of the techniques accomplishes analysis of code and they are not a black box technique via image matching. However, the feature-based test oracle we have previously mentioned [9] has been used as an image matching strategy for Web applications testing.

Other work within the Web domain proposed a pairwise image comparison (production page × staging page) to locate Web page layout faults while neglecting insignificant variations [27]. Their approach is suitable for regression testing, uses Adaptive RT [7] to select regions to be compared in pairs of images, and is based on the characteristics of failure patterns of browser layouts. Cross-browser compliance is not an issue in our case since we assume that the environmental modeling tools are desktop and not Web applications. Moreover, our methodology proposes RT not to select portions of images to be compared but rather to be a systematic way to create a reference base to help the test oracle.

Another research within the Web domain presented a technique for detecting presentation failures in Web applications, and localize the HTML elements that are likely to be responsible for the failures [18]. They used computer vision techniques to compare a Web page rendered in a browser with its oracle information and identify difference pixels. Fault localization is an issue with black box approaches, and thus it is interesting to address this point like the researchers did. However, it is not clear how generic their approach can be since they did fault localization by mapping difference pixels to HTML elements. In other words, it is probable that their technique is not suitable, under this respect, for other programming languages. In the detection phase, they identified presentation failures by comparing the screenshot of the actual page, as rendered in a browser, with its expected appearance, the oracle information. But, they performed this phase with Perceptual Image Differencing (PID) which requires not only one but four parameters to be defined by the professional. As we have previously mentioned, tuning of parameters can be very demanding in practical terms.

A framework to test image processing applications was proposed in [15]. It includes test data generation (images) via symbolic evaluation, execution of test data, and test results evaluation (test oracle) via MT and SVM (ML algorithm). In order to apply their approach, it is necessary to instrument the source code of the application aiming to indicate the interested code areas executed during a test run. Although used by researchers, code instrumentation can be dangerous because it can insert additional faults or interfere on the behavior of the application. Regarding the test oracle, we have already pointed out the problem of identifying the metamorphic relations within MT. They also proposed SVM as an oracle (MT or SVM can be selected) but our methodology and supported tool suggested up to six ML algorithms to choose, giving more flexibility to the tester.

We can assert that there are two main differences between our methodology and the studies presented in this section. First, DaOBML suggests a set of ML algorithms as the main component of the test oracle. Only one research study [15] used a single ML algorithm (SVM) as test oracle while we suggested up to six and performed an empirical evaluation where ANN was the best. We believe that the potentialities of ML and Artificial Intelligence methods are very significant and thus Software Engineering processes, such as testing, should take them more into consideration. Second, our methodology proposes the use of test case/data generation techniques (CIT, MBT, RT) as a systematic procedure to create or to improve Knowledge Bases to support the operation of the test oracle. Hence, we aimed at improving the quality of bases via techniques to generate test suites.

7 CONCLUSIONS

This paper presented a new methodology with tool support, DaOBML, to help improving the quality of environmental models where the outputs are complex artifacts such as images (maps) or plots. We believe that Software Engineering processes, such as testing, with all the techniques and methods that have been studied can give valuable contributions to develop better software systems which produce complex outputs such as images or even when these outputs are not exact, as it might happen with scientific software.

We relied on three test data generation techniques (CIT, MBT, RT) to drive the creation of Knowledge Bases. These techniques may also be used as new test data generators as traditionally but the main goal is to develop new or to improve existent Knowledge Bases in a situation where we have a set of environmental models that are sufficient trusted to generate the expected results (Knowledge Bases). The core of the Oracle Procedure are six ML algorithms (ANN, DT, K-NN, NB, RF, SVM) which assign the verdicts of new test data.

Considering environmental models developed via the TerraME product, a controlled experiment makes us to conclude that RT is the most feasible test data generation approach for developing the Knowledge Bases to support the test oracle, ANNs prove again that is an interesting Artificial Intelligence approach since it achieved the best performance among all considered alternatives, and the size of the Knowledge Base is important so that the larger is the better.

Even though, at first, scaling variations and illumination are not an issue to us, we need to enhance the digital image processing steps of our methodology and tool. Filtering techniques can be used as well as we can consider other edge detectors such as the ANN itself and phase congruency methods. We also need to accomplish other controlled experiments increasing the number of environmental models as well as assessing models that are not closely similar to the ones that created the Knowledge Bases, so that we can generalize our results. Mutation testing can also help in these

other experiments by creating faulty versions of the environmental models and hence we can better evaluate the effectiveness of our approach. Like any other black box approach related to system and acceptance testing levels, fault localization is an issue within DaOBML, and we aim at working in this direction where we can rely on supervised or unsupervised ML algorithms for this purpose. In addition, we will try to develop this fault localization strategy in a generic manner so that it can be suitable to several programming languages used for environmental modeling.

ACKNOWLEDGMENTS

This research was supported in part by a PIBIC grant (Process: 142440/2017-5) from the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq), Brazil.

REFERENCES

[1] D. Amalfitano, A. R. Fasolino, P. Tramontana, and S. Scala. 2014. Towards Automatic Model-in-the-loop Testing of Electronic Vehicle Information Centers. In *Proceedings of the 2014 International Workshop on Long-term Industrial Collaboration on Software Engineering (WISE '14)*. ACM, New York, NY, USA, 9–12. https://doi.org/10.1145/2647648.2656427

[2] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. Mcminn. 2013. An Orchestrated Survey of Methodologies for Automated Software Test Case Generation. *Journal of Systems and Software* 86, 8 (Aug. 2013), 1978–2001. https://doi.org/10.1016/j.jss.2013.02.061

[3] J. M. Balera and V. A. Santiago Júnior. 2016. A Controlled Experiment for Combinatorial Testing. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing (SAST)*. ACM, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/2993288.2993289

[4] J. M. Balera and V. A. Santiago Júnior. 2017. An algorithm for combinatorial interaction testing: definitions and rigorous evaluations. *Journal of Software Engineering Research and Development* 5, 1 (28 Dec 2017), 41. https://doi.org/10.1186/s40411-017-0043-z

[5] M. N. Borazjany, L. Yu, Y. Lei, R. Kacker, and R. Kuhn. 2012. Combinatorial Testing of ACTS: A Case Study. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. 591–600. https://doi.org/10.1109/ICST.2012.146

[6] T. G. S. Carneiro, P. R. Andrade, G. Câmara, A. M. V. Monteiro, and R. R. Pereira. 2013. An extensible toolbox for modeling nature–society interactions. *Environmental Modelling & Software* 46 (2013), 104 – 117. https://doi.org/10.1016/j.envsoft.2013.03.002

[7] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T.H. Tse. 2010. Adaptive Random Testing: The ART of test case diversity. *Journal of Systems and Software* 83, 1 (2010), 60 – 66. https://doi.org/10.1016/j.jss.2009.02.022

[8] E. M. Clarke, E. A. Emerson, and A. P. Sistla. 1986. Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems* 8, 2 (April 1986), 244–263. https://doi.org/10.1145/5397.5399

[9] M. E. Delamaro, F. L. S. Nunes, and R. A. P. Oliveira. 2013. Using concepts of content-based image retrieval to implement graphical testing oracles. *Software Testing, Verification and Reliability* 23, 3 (2013), 171–198. https://doi.org/10.1002/stvr.463

[10] D. L. Dunkerley. 1997. Banded vegetation: development under uniform rainfall from a simple cellular automaton model. *Plant Ecology* 129, 2 (01 Feb 1997), 103–111. https://doi.org/10.1023/A:1009725732740

[11] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. 1999. Patterns in Property Specifications for Finite-state Verification. In *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. ACM, New York, NY, USA, 411–420. https://doi.org/10.1145/302405.302672

[12] Earth System Science Center (COCST/INPE). 2018. TerraME: Multiparadigm Modeling Toolkit. Available from: http://www.terrame.org/doku.php. Access in: May 18, 2018.

[13] R. Eisinga, T. Heskes, B. Pelzer, and M. Te Grotenhuis. 2017. Exact p-values for pairwise comparison of Friedman rank sums, with application to comparing classifiers. *BMC Bioinformatics* 18, 1 (25 Jan 2017), 68. https://doi.org/10.1186/s12859-017-1486-2

[14] R. Ierusalimschy, L. H. Figueiredo, and W. C. Filho. 1996. Lua – an extensible extension language. *Software—Practice & Experience* 26, 6 (June 1996), 635–652. https://doi.org/10.1002/(SICI)1097-024X(199606)26:6<635::AID-SPE26>3.0.CO;2-P

[15] T. Jameel, L. Mengxiang, and L. Chao. 2016. A framework of automatic testing of image processing applications. In *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. 312–317. https://doi.org/10.1109/IBCAST.2016.7429896

[16] M. C. Júnior, R. A. P. Oliveira, M. A. G. Valverde, M. P. Jackowski, F. L. S. Nunes, and M. E. Delamaro. 2017. Feature-Based Test Oracles to Categorize Synthetic 3D and 2D Images of Blood Vessels. In *Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing (SAST)*. ACM, New York, NY, USA, Article 11, 6 pages. https://doi.org/10.1145/3128473.3128484

[17] M. F. Kiraç, B. Aktemur, and H. Sözer. 2018. VISOR: A fast image processing pipeline with scaling and translation invariance for test oracle automation of visual output systems. *Journal of Systems and Software* 136 (2018), 266 – 277. https://doi.org/10.1016/j.jss.2017.06.023

[18] S. Mahajan and W. G. J. Halfond. 2015. Detection and Localization of HTML Presentation Failures Using Computer Vision-Based Techniques. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. 1–10. https://doi.org/10.1109/ICST.2015.7102586

[19] N. Nausheen, A. Seal, P. Khanna, and S. Halder. 2018. A FPGA based implementation of Sobel edge detection. *Microprocessors and Microsystems* 56 (2018), 84 – 91. https://doi.org/10.1016/j.micpro.2017.10.011

[20] R. A. P. Oliveira. 2017. *Test oracles for systems with complex outputs: the case of TTS systems*. Ph.D. Dissertation. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo (USP).

[21] R. A. P. Oliveira, A. M. Memon, V. N. Gil, F. L. S. Nunes, and M. Delamaro. 2014. An extensible framework to implement test oracles for non-testable programs. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE* January (2014), 199–204. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84935074236&partnerID=40&md5=778fb3296d6f16acf77a7e70702d9a9d

[22] J. Petke, M. B. Cohen, M. Harman, and S. Yoo. 2015. Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection. *IEEE Transactions on Software Engineering* 41, 9 (Sept 2015), 901–924. https://doi.org/10.1109/TSE.2015.2421279

[23] I. Portugal, P. Alencar, and D. Cowan. 2018. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications* 97 (2018), 205 – 227. https://doi.org/10.1016/j.eswa.2017.12.020

[24] S. Audet. 2018. JavaCV. Available from: https://github.com/bytedeco/javacv. Access in: May 18, 2018.

[25] V. A. Santiago Júnior and F. E. C. Silva. 2017. From Statecharts into Model Checking: A Hierarchy-based Translation and Specification Patterns Properties to Generate Test Cases. In *Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing (SAST)*. ACM, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/3128473.3128475

[26] V. A. Santiago Júnior, L. A. R. Silva, and P. R. A. Neto. 2018. Empirical Evaluation Data - SAST 2018. Available from: https://bit.ly/2v9ygj5. Access in: July 31, 2018.

[27] E. Selay, Z. Q. Zhou, and J. Zou. 2014. Adaptive Random Testing for Image Comparison in Regression Web Testing. In *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 1–7. https://doi.org/10.1109/DICTA.2014.7008093

[28] S. G. Shaila and A. Vadivel. 2016. Indexing and encoding based image feature representation with bin overlapped similarity measure for CBIR applications. *Journal of Visual Communication and Image Representation* 36 (2016), 40 – 55. https://doi.org/10.1016/j.jvcir.2016.01.003

[29] J. Silvertown, S. Holtier, J. Johnson, and P. Dale. 1992. Cellular Automaton Models of Interspecific Competition for Space –The Effect of Pattern on Process. *Journal of Ecology* 80, 3 (1992), 527–533. https://doi.org/doi:10.2307/2260696

[30] K. Y. Sim, D. M. L. Wong, and T. Y. Hii. 2013. Evaluating the Effectiveness of Metamorphic Testing on Edge Detection Programs. In *International Journal of Innovation, Management and Technology*, Vol. 4. 6–10.

[31] The Univesity of Waikato. 2018. Weka 3: Data Mining Software in Java. Available from: https://www.cs.waikato.ac.nz/ml/weka/. Access in: May 18, 2018.

[32] C. Zannier, G. Melnik, and F. Maurer. 2006. On the Success of Empirical Studies in the International Conference on Software Engineering. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 341–350. https://doi.org/10.1145/1134285.1134333