

Test Case Selection Using CBIR and Clustering

Research-in-Progress

Everton Note Narciso

School of Arts, Sciences and Humanities - EACH
University of São Paulo - Brazil
evernarciso@usp.br

Márcio Eduardo Delamaro

Institute of Mathematics and Computer Sciences
University of São Paulo - Brazil
delamaro@icmc.usp.br

Fátima de Lourdes dos Santos Nunes

School of Arts, Sciences and Humanities - EACH
University of São Paulo - Brazil
fatima.nunes@usp.br

ABSTRACT

Choosing test cases for the optimization process of information systems testing is crucial, because it helps to eliminate unnecessary and redundant testing data. However, its use in systems that address complex domains (e.g. images) is still underexplored. This paper presents a new approach that uses Content-Based Image Retrieval (CBIR), similarity functions and clustering techniques to select test cases from an image-based test suite. Two experiments performed on an image processing system show that our approach, when compared with random tests, can significantly enhance the performance of tests execution by reducing the test cases required to find a fault. The results also show the potential use of CBIR for information abstraction, as well as the effectiveness of similarity functions and clustering for test case selection.

Keywords

Test case selection, content-based image retrieval, clustering, similarity functions.

INTRODUCTION

The methods used during the software testing phase directly affect the product's final quality. Manual and unplanned testing usually implies doubtful reliability in terms of software production, not meeting the desired requirements.

Automated testing is a process that seeks to minimize the subjectivity of manual testing and optimize the available resources. To create mechanisms that exploit software requirements with the least possible computational effort represents a major challenge, and in this scenario the tools for test case selection are vital to determine the test strategies.

The main objective to select test cases regards eliminating redundant or unnecessary test data. There are many approaches to select test cases for systems under object-oriented paradigm, embedded systems and systems with alphanumeric inputs/outputs in general (Engstrom, Skoglund and Runeson 2008; Yoo and Harman 2012). However, there is a gap with regards to selecting test cases for graphic domain systems, such as image processing systems.

To test a system using a large number of images requires high computational costs. Considering the impracticality of this alternative, we present a new approach for selecting test cases. Our approach focuses on image processing systems, which consists of applying Content-Based Image Retrieval (CBIR), similarity functions and clustering techniques in order to select more relevant test cases (images), based on their characteristics.

To achieve the proposed objective, the paper was organized as follows: Section 2 presents the basic concepts of test case selection, image processing, CBIR, similarity functions and clustering - fundamental topics to develop the proposal. Section 3 presents the methodology of the experiments. Section 4 shows the results. Section 5 presents the discussions and threats to validity. Section 6 presents the related works and Section 7 concludes the paper.

BACKGROUND

To develop the proposal, the contributions from different computing areas must be taken into account, such as image processing, CBIR and Software Engineering. This section presents the fundamental concepts regarding such topics.

Image Processing

Digital image processing consists of a set of techniques to capture, represent and transform images using a computer. The digital image is a representation of a scene in a discrete region, limited by a finite set of integer values that represent each of their dots. Mathematically, an image can be described as a function of light-intensity returned by an object. Most images consider the two-dimensional space, defined as a function $f(x,y)$, where x and y are the spatial coordinates, and the value of f in coordinates x,y provides the image intensity at the dot. Each value refers to a dot in the image and represents one color. These tiny dots are called pixels. A pixel is the smallest unit of an image, spatially, on which operations can be performed (Gonzales and Wintz, 1987).

Image processing is the entire set of operations that can be applied to a matrix that represents an image. The purpose to mathematically define the image is the possibility to manipulate its contents in order to make changes or to extract important information (Gonzales and Wintz, 1987).

Content-Based Image Retrieval (CBIR)

A CBIR system consists of a set of techniques that uses visual content, usually called characteristics, to find images according to user requests from an image used as model template. Figure 1 presents the typical structure of a CBIR system (Delamaro, Nunes and Oliveira, 2011).

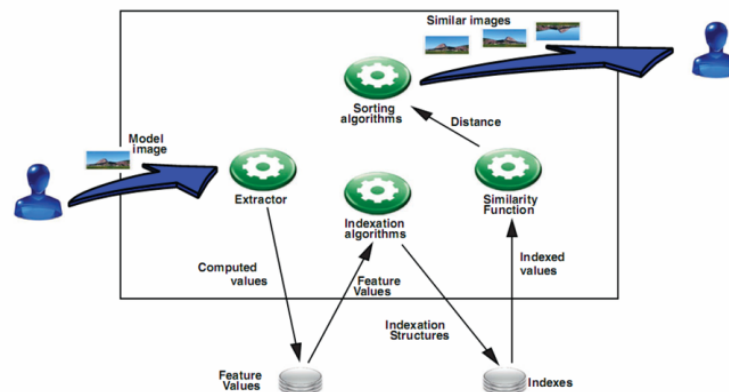


Figure 1. Typical Structure of a CBIR System.

Figure 1 shows that a user can specify a query by providing a model image and a set of similar images are returned as the system output. A CBIR system processes the information contained in the digital images and creates an abstraction of such images in terms of visual attributes. The characteristics can, for example, refer to color, texture or form, and can be represented by a numerical vector that contains the information of the image in question (Delamaro, Nunes and Oliveira, 2011). In a CBIR system the similarity among the images is calculated by using similarity functions that allow comparing characteristics extracted using image processing techniques. Considering that the heuristic in which the more dissimilar test cases are preferable for detecting faults, we can use the CBIR concepts and similarity functions to select an optimized subset of test cases, as described in the next sessions.

Test Case Selection

Test cases (TC) are input parameters, execution conditions, and expected results used for testing (IEEE, 1990). During the development phase of large and complex software, it is customary that the size of the test suites increases as the software requirements evolve. However, the execution of large test suites can be impractical because the time and resources limitations should be taken into account. To mitigate this problem, methods should be implemented to reduce the number of test cases to be run. In this context, the test case selection seeks to select an optimized subset of test cases from a finite test suite using some criterion of interest to eliminate the redundant or unnecessary test data and maximize fault detection rates (Hemmati and Briand, 2010).

Similarity Functions

The test case selection using similarity concepts is based on heuristics, in which the more dissimilar test cases are preferable for detecting faults (Chen, Kuo, Merkel and Tse, 2010). This approach uses similarity functions to measure the distance (or difference) between the test cases. The similarity functions should be suitable for representing and manipulating the input data (Hemmati, Arcuri and Briand, 2010). Some examples of similarity functions are Euclidean distance and the Coverage Manhattan distance (Hemmati and Briand, 2010). For test case selection, once the similarity function is defined, the method chooses a dissimilar and reduced subset of test cases, optimizing the testing process. Nevertheless, besides the similarity functions, we used the clustering techniques to grouping images and increase the power of inference about the test cases. The basic concepts of clustering are described in the next session.

Clustering

Clustering is a technique used to group similar objects into mutually exclusive groups. Thus, each group consists of objects that are similar to each other but dissimilar to the objects of other groups (Sapna and Mohanty 2010).

The clustering process identifies and classifies objects according to their characteristics, minimizing the intragroup variance and maximizing the intergroup variance (Hemmati *et al.*, 2010). The similarity criterion used in the clustering process can be, for example, the variability between two or more objects belonging to the same cluster, such as the mean, standard deviation, the Euclidean distance, and others. The result is a number of heterogeneous groups with homogeneous content. The classification and attainment of condensed data groups enables to increase the inference power on the sets observed.

METHODOLOGY

The experiment consists of using CBIR, similarity function and clustering techniques to select test cases for testing image processing systems. The theoretical foundation is mainly based on the heuristic that the more dissimilar test cases are preferable for detecting faults. The test cases, in this case, are the images used as input parameters for the software under test (SUT). The experimental test is determined in three steps: given the test suite, the algorithm extracts the characteristics of each image, performs clustering using similarity functions and selects the most dissimilar test cases. The result is an optimized subset of test cases. Figure 2 illustrates the selection process proposed.

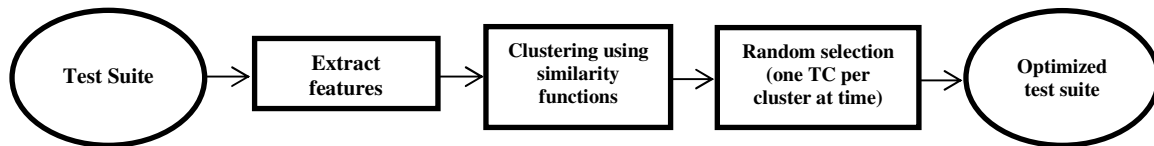


Figure 2. Selection Process Using CBIR and Clustering.

Delamaro *et al.* (2011) developed a framework that uses CBIR techniques for defining test oracles with graphical output systems, known as O-FIm (Oracle for Images). The O-FIm supports a set of classes developed in the Java language which enables extracting the visual content of images, creating characteristic vectors and calculating the distance between the processed images. Some O-FIm classes were adapted in order to develop our approach.

The image processing system used for testing was developed by our team in the Java language, intended to skelitize grayscale images in standard RGB. Thus, the skelitization process removes all redundant pixels, producing a new simplified image reduced to the width of a single-pixel, as illustrated in Figure 3.

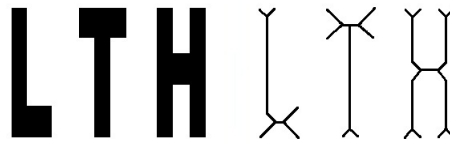


Figure 3. Illustration of the Skelitization Result Process.

The functional characteristics of SUT were empirically considered to define the contents of the abstracted images using CBIR techniques. The purpose of the system developed is to remove redundant pixels from an image and simplify it with the width of a single pixel. The skelitized images must be in the grayscale. Based on this information, the following characteristic extractors were defined: color (to differentiate the shades of gray), area (to determine the spaces filled in shades of gray) and perimeter (to determine the location of the external pixels of each image). In fact, the skelitization process directly affects the color characteristics, area and perimeter of the skelitized image.

In the CBIR process, each characteristic can be represented by a numerical value varying between zero and one. At the end of the extraction process, each image is represented by a numerical vector with three positions, called a characteristic vector, related to the three characteristics implemented. Algorithm 1 shows the pseudocode of the area extractor.

```

Input: Test suite (images)
1 for each image from test suite do
2   Width ← width of image
3   Height ← height of image
4   SizeOfImage ← Width * Height
5   Color ← any shade of gray
6   Area ← 0
7   for each pixel of image do
8     if pixel = Color then
9       Area ← Area + 1
10  VectorOfFeatures [ ] ← Area/SizeOfImage
11 return VectorOfFeatures [ ]
12 End of execution

```

Algorithm 1. Area Characteristic Extractor.

Defining the extractors and creating the characteristic vectors enables to systematically apply the similarity functions in order to find the most dissimilar images in terms of characteristics of interest. Figure 4 presents the similarity functions used in the experiments. Equation 1 is the Canberra distance; equation 2 is the Jeffrey distance; equation 3 is the Euclidean distance and equation 4 is the Chi-square.

$$Dis(A, B) = \sum_{i=0}^{n-1} \frac{|a_i - b_i|}{|a_i| + |b_i|} \quad (1)$$

$$DKL(A, B) = \sum_{i=0}^{n-1} b_i \log \frac{b_i}{a_i} \quad (2)$$

$$Dis_{L_2}(A, B) = \sqrt{\sum_{i=0}^{n-1} (a_i - b_i)^2} \quad (3)$$

$$Dis(A, B) = \sum_{i=0}^{n-1} \frac{(b_i - m_i)^2}{m_i}, m_i = \frac{a_i + b_i}{2} \quad (4)$$

The hierarchical clustering method, known as Agglomerative Hierarchical Clustering (AHC), was used to identify and group the most similar test cases (Sapna and Mohanty, 2010). Given a test suite of size S , a similarity matrix of size $(S*S)/2$ is created to represent all possible distances between the test case pairs. Note that the similarity matrix is symmetric. Due to the distance between the test cases A and B are the same distance as between B and A , there is no need to build a matrix of size $S*S$. In AHC, each test case represents an isolated cluster. At each clustering stage, the two most similar clusters are grouped into one cluster. Figure 4 represents the initial stages of an AHC process. In the first three stages of this example the closest clusters are (1)↔(2), (6)↔(8) and (13)↔(16), respectively. In these stages each cluster is isolated and grouped, generating new clusters (cluster 1, cluster 2 and cluster 3). At the end of the third stage, the closest clusters are (1,2)↔(6,8). In the fourth stage these clusters are grouped, generating a new cluster (1,2,6,8). The process is repeated until the desired stopping point is achieved (e.g. when a similarity level is achieved) or until all data are grouped into a single cluster.

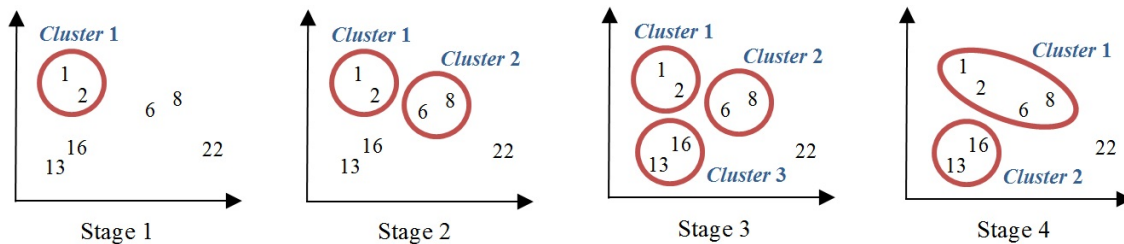


Figure 4. Representation of AHC Process.

In short, in our experiment the similarity functions were used to create similarity matrices M and the AHC was applied to these matrices. After the clusters were created, a test case was randomly selected for each cluster until reaching the desired test suite size.

Evaluation of Characteristic Extractors

This section presents the statistical evidence to support previous hypotheses that given the functional characteristics of SUT, in which the characteristic extractors called color, area and perimeter are suitable for abstracting the test cases, since such abstractions are relevant to find the data dissimilarity. In this scenario, the importance of using multiple characteristics is due to the fact that a given test case A may contain the same characteristic of another test case B (such as area), and in such cases it is necessary to find additional characteristics to measure the dissimilarity between A and B .

We empirically assume the hypothesis that independent, uncorrelated and nonlinear characteristics (also called variables) are preferable to reveal the dissimilarity between the test cases. We consider that dependent, strongly correlated or linear variables tend to reveal the same fault patterns, undesirable within the context of our experiment.

As three different characteristic extractors were used, a scatterplot was deployed to analyze the behavior of data and the correlation analysis and chi-square test were performed to evaluate the influence of each variable on the dissimilarity calculation. Thus, the scatterplot allows visualizing the behavior of data. The correlation allows analyzing the strength of the relationship among data and the chi-square test allows analyzing the dependence of data (Devore, 2012). In other words, the statistical tests performed should answer the following questions: 1) what is the influence of each variable on the dissimilarity calculation? 2) what is the effect that changes in any variable cause on the other variables?

All statistical tests were performed using the R software version 2.15.2 (R Core Team, 2012). The characteristic vectors were submitted to the software, which in turn returned the statistical results of the experiment. Figure 5 shows the scatterplot of color, area and perimeter characteristics.

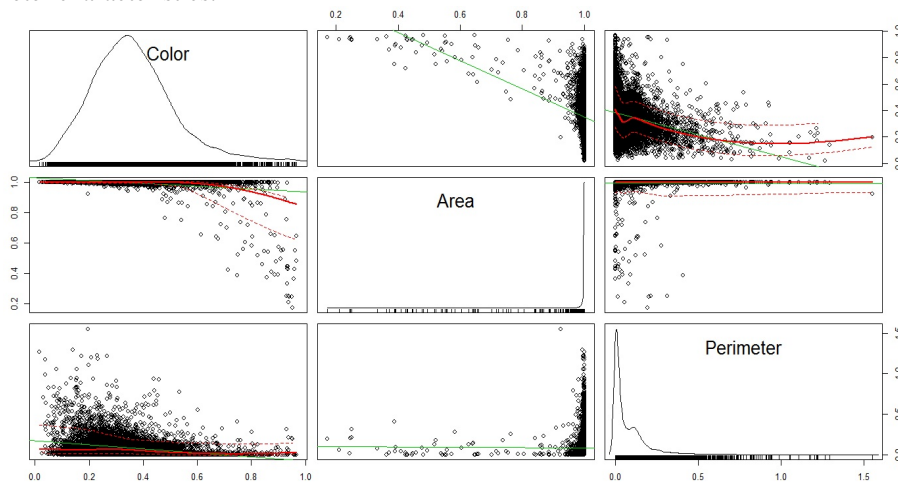


Figure 5. Scatterplot of Color, Area and Perimeter Characteristics.

Figure 5 is a scatterplot that shows the behavior of pairs of data (Color, Area and Perimeter). Each dot represents a test case and the straight line represents the regression line. If there are many dots following the line, there are indications that the data possibly has the same behavior patterns. Thus, the scatterplot depicting evidences that the behavior of the data (all variables) is nonlinear. Table 1 presents the correlation tests.

Correlation			
Characteristic	Color	Area	Perimeter
Color	1	0,31	0,29
Area	0,31	1	0,008
Perimeter	0,29	0,008	1

Table 1. Correlation Tests.

Table 1 depicts the correlation tests among the pairs of characteristics color, area and perimeter. The correlation analysis shows that the variables are weakly correlated (because the result is much less than one), indicating that modifying any variable can cause little or no changes in the other variables. The result of the p-value of the chi-square test is less than 0.5% - indicating that the variables are independent.

All tests carried out suggest that the variables are nonlinear, weakly correlated, independent and strongly influential on the dissimilarity calculation. Based on these statistics we assume that removing any variable (color, perimeter or area) can compromise the efficiency of the selection based on dissimilarity.

Finally, it should be noted that the statistical results are incipient and represent evidence to support the hypothesis that the characteristic extractors used are suitable for the experiment.

Evaluation Metrics

There are two skelization software versions, each with a single semantic seeded error and detected by a small range of test cases available in the test suite. The test suite consists of seven thousand nine hundred images manually and randomly taken from the Internet, without any restriction regarding its use. All images were reset to grayscale and RGB default. Only fifty test cases were able to reveal the seeded errors, representing 0.6% of the entire test suite.

The domain of software that works with images is complex. In the context of our experiment the heuristic on similarity-based selection is preferable than the heuristic on coverage-based selection (more specifically code coverage) because a single image can contain a lot of information (e.g. a small image of size 100 lines per 100 columns contains ten thousand pixels), and a small set of images covers most part of the SUT code, but cannot find the existing fault. Note that the test suite is comprised of seven thousand nine hundred images, and only fifty images are able to find the seeded errors. That is called data-sensitive faults, in contrast to coverage-sensitive faults.

Regarding the evaluation of the experiments, researchers have adopted different metrics to determine the effectiveness and the efficiency of the methods proposed, such as the fault detection rate (Hemmati *et al.*, 2010) and the model or code coverage (Mirarab, Akhlaghi and Tahvildari, 2011). Researchers also have adopted the number of test cases required to find the first fault as the key metric for evaluating the selection methods (Kuo, Chen, Liu and Chan 2007; Chen *et al.*, 2010). The general idea is that a fault at a given point of the code can cause other faults at different points of the code. Therefore, to find the first fault, the developer must repair the error that caused the fault before submitting the software to a new test, thereby avoiding other faults resulting from the first fault (Rothermel, Untch, Chengyun and Harrold, 2001). As mentioned, this is particularly useful in a domain susceptible to data-sensitive faults, as the one addressed in this article.

The evaluation metrics used in the experiment measures the number of test cases required to find the first fault. Our intention is to show that the proposed selection method can outperform the F-measure, which is the measure of number of test cases required to find the first fault at random. To obtain a statistically consistent result, the random selection was executed one hundred times and the simple average was considered as the final result of the experiments, which is compared to our selection using CBIR and clustering.

OBTAINED RESULTS

Given the test suites, running the two experiments using our new approach consists of selecting the test cases and submit them to SUT (each containing only one seeded error), as well as to the original software (considered correct). The results of both processes are compared (pixel by pixel), and the experiment is concluded when the execution of the test cases selected shows differentiated outputs, indicating there is a fault. Figure 6 illustrates the test running process through an activity diagram of Unified Modeling Language (UML).

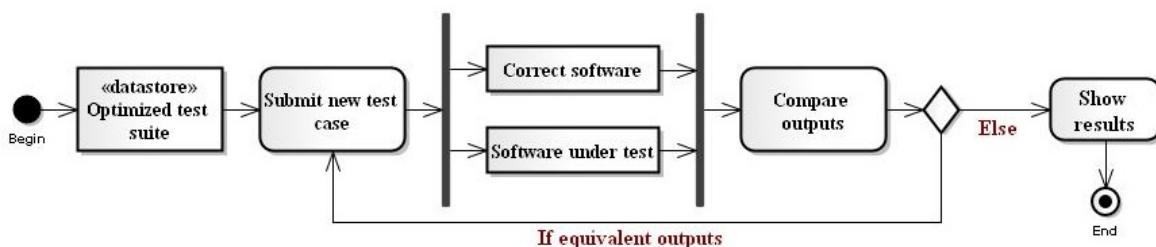


Figure 6. Test Running Process.

Figure 6 shows that the test cases are submitted to the execution of tests until a fault is detected. The same procedure was used for the random selection and for our selection using CBIR and clustering. Table 2 shows details about the tests and the results of the experiments.

Selection Method	Similarity Function	# TC required to find the first fault	Experiment
Random	None	148 (<i>F-measure</i>)	SUT1
		141 (<i>F-measure</i>)	SUT2
CBIR, similarity functions and clustering	Canberra	13	SUT1
	Jeffrey	14	
	Euclidean	25	
	Chi-Square	13	
	Canberra	31	SUT2
	Jeffrey	135	
	Euclidean	22	
	Chi-Square	19	

Table 2. Results of Test Case Selection Process.

Table 2 presents the overall results of the experiments. The column 3 shows the test times (the number of executed test cases until a fault is detected). Note that all tests based on CBIR and similarity functions outperformed the random tests. In SUT1, an average of 16 test cases was needed to find the first fault, which when compared to the *F-measure* represents only 11%. In SUT2, although the Jeffrey function showed an unsatisfactory result, an average of 52 test cases was needed to detect the first fault, which compared to the *F-measure*, represents only 37%.

The results of the experiments show the potential of the techniques used in the software testing process. First, using CBIR to abstract relevant information from images proved feasible and suitable. Second, the selection heuristic using similarity functions was effective. Third, using clustering to group the most similar test cases also showed good results.

DISCUSSION AND THREATS TO VALIDITY

A limitation of the experiments is the fact that the SUT used in our experiment is relatively small (~200 lines of code), and the extractors used were color, perimeter and area. Replication studies considering large software may reveal different patterns of effectiveness of our proposed method. Future works should perform tests on different image processing systems, as well as develop and implement new characteristic extractors that meet different demands to corroborate the good results achieved. Despite the need for developing and implementing new characteristic extractors, there is a framework to help the creation of automated oracles. This framework contains various classes of extractors and similarity functions to support graphic processing testing and is available at the Free-software Competence Center in the University of São Paulo (<http://ccsl.icmc.usp.br/pt-br/projects/o-fim-oracle-images>).

Regarding the results validation, the SUT2 showed a poor result when the Jeffrey distance was used for the selection. Out of 141 random selection test cases, it needed 135 test cases to find the first fault. This result indicates that the similarity functions used have a direct impact on the selection process proposed. In practical terms, it can be inferred that in the SUT2 the Jeffrey function did not represent the dissimilarity between test cases and, therefore, studies on this and other functions should be complemented and expanded, within the context of our approach.

A major advantage provided by our approach lies in the fact that extracting characteristics from images has to be performed only once. In fact, despite the computational cost and the time required for data abstraction, the characteristic vectors can be stored in a database and used later to select test cases independent of the SUT.

Another benefit is that redundant images (in terms of characteristics) can be deleted from the test suites. After executing CBIR and similarity functions, only the images stored that meet some predetermined dissimilarity criterion can be kept.

Scalability is the third potential gain of our approach. There is always the possibility to add a new image in the test suites, and in this case there is no need to reprocess (via CBIR) the previously included images.

We believe that our approach can be exploited in related areas, such as Search-based Software Engineering. Therefore, the test case selection can be treated as a high-complexity optimization problem in which search-based methods explore a range of possible solutions in order to find the best result combinations, with a reasonable computational cost (Hemmati *et al.*, 2010; Li, Harman and Hierons, 2007). Different search-based methods have been successfully used in various software

testing activities, such as genetic algorithms (Li *et al.*, 2007) and particle swarm optimization (Souza, Miranda, Prudencio, and Barros, 2011). The use of search-based methods for test case selection using CBIR and similarity functions can contribute to optimize the scalability, robustness and reliability of the tests.

Finally, we believe it is possible to apply our approach to the prioritizing process of test cases. Prioritizing will sequentially arrange the test cases, enabling to execute first the most important test data. Its goal is to find the presence of faults as early as possible, allowing a faster depuration of SUT (Li *et al.*, 2007). Similar to the search-based methods, prioritization has been successfully employed in research that is focused mainly on regression testing. Using CBIR and similarity functions to prioritize test cases is a promising research possibility, considering that many selection methods used can be adapted to prioritization (Bo, Zhang, Chan and Tse, 2009; Catal, 2012).

RELATED WORKS

Chen, Merkel and Tse (2010) presented a synthesis of important theoretical studies related to adaptive random testing (ART), a similarity-based test case selection method. In the ART method, test cases are randomly chosen from the original test suite, and an evaluation of the results is then performed so that only the test cases that meet the dissimilarity criterion are utilized. In this study the researchers also discuss relevant topics such as fault patterns, theoretical limits, random sequences and fault-based tests.

Engstrom, Skoglund and Runeson (2008) developed a systematic review on test case selection methods for regression testing. The researchers selected 28 of 2923 articles found in various scientific data sources. Despite the different selection methods used (such as genetic algorithms, simulated annealing, dataflow-based coverage, among others), the researchers found no evidence determining the superiority of any particular method with regards to all the methods evaluated.

Yoo and Harman (2012) presented a survey on the approaches to minimization, selection and prioritization of test cases for regression testing. The researchers discuss various methods used to reduce the test suite, the interrelationship between approaches, the characteristics, advantages and disadvantages of each topic under consideration.

Many test case selection methods have been sufficiently studied, analyzed and implemented, resulting in mature, viable and well established solutions. However, we found no studies that utilize CBIR and similarity functions for selecting test cases considering systems in the image processing domain.

CONCLUSION

This paper presented a new approach for test case selection using CBIR, similarity functions and clustering. The proposed selection method uses CBIR techniques to abstract relevant information from images, and then performs clustering using similarity functions. The most dissimilar images are selected and submitted to the software under test. The experiment results show the feasibility of the method presented, and in the context evaluated, its superiority compared to random selection. Our approach, focused on testing image processing systems, is promising in terms of data reduction in the test suites, generality (can be used in any image processing system) and scalability.

There are many well-established test case selection methods in the literature. However, we found no studies that utilize CBIR and similarity functions for test case selection. Additionally, our approach paves the way for other research possibilities, such as its application to Search-based Software Engineering and prioritization of test cases. Some future works can include: 1) creating a framework for selecting test cases using CBIR and similarity functions; 2) implementing new characteristic extractors and new similarity functions; 3) using different evaluation metrics, such as the number of test cases needed to find all the faults (in situations with more than one fault in SUT); and 4) conducting different software tests to analyze and evaluate the effectiveness of the proposed approach.

ACKNOWLEDGMENT

This research was supported by the State of São Paulo Research Foundation (Fundação de Amparo à Pesquisa do Estado de São Paulo-Fapesp) Process 2010/15691-0 and Brazilian National Council of Scientific and Technological Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq) Process 559931/2010-7 and the National Institute of Science and Technology-Medicine Assisted by Scientific Computing (Instituto Nacional de Ciência e Tecnologia - Medicina Assistida por Computação Científica-INCT-MACC).

The authors thank to Professor Delhi Tereza Paiva Salinas from the University of São Paulo for her help in statistical analysis.

REFERENCES

1. IEEE Standard Glossary of Software Engineering Terminology (610.12-1990). URL <http://standards.ieee.org/findstds/standard/610.12-1990.html>.
2. Delamaro, M. E., Nunes, F. L. S., Oliveira, R. A. P. (2011) Using concepts of content-based image retrieval to implement graphical testing oracles. *Software Testing, Verification e Reliability*. John Wiley e Sons. 1099-1689.
3. Hemmati, H., Arcuri, A., Briand, L. (2010) Reducing the cost of model-based testing through test case diversity. *IFIP International Federation for Information Processing*, pp.63–78.
4. Kuo, F. C., Chen, T. Y., Liu, H., Chan, W. K. (2007) Enhancing adaptive random testing in high dimensional input domains. *Proceedings of the 2007 ACM symposium on Applied computing (SAC '07)*. ACM, New York, USA, pp.1467-1472.
5. Bo, J., Zhang, Z., Chan, W. K., Tse, T. H. (2009) Adaptive Random Test Case Prioritization. *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*. IEEE Computer Society, Washington, USA, pp.233-244.
6. Catal, C. (2012) On the application of genetic algorithms for test case prioritization: a systematic literature review. *Proceedings of the 2nd international workshop on Evidential assessment of software technologies (EAST '12)*. ACM, New York, USA, pp.9-14.
7. Mayer, J. (2006) Adaptive random testing with randomly translated failure region. *Proceedings of the 1st international workshop on Random testing (RT '06)*. ACM, New York, USA, pp.70-77.
8. Hemmati, H., Briand, L. (2010) An Industrial Investigation of Similarity Measures for Model-Based Test Case Selection. *Software Reliability Engineering (ISSRE)*, pp.141-150.
9. Sapna, P. G., Mohanty, H. (2010) Clustering test cases to achieve effective test selection. *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India (A2CWIC '10)*. ACM, New York, USA, pp.15-8.
10. R Core Team (2012) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. URL <http://www.R-project.org/>.
11. Li, Z., Harman, M., Hierons, R. M. (2007) Search Algorithms for Regression Test Case Prioritization. *Software Engineering, IEEE Transactions on*, pp.225-237.
12. Souza, L. S. de, Miranda, P. B. C. de, Prudencio, R. B. C., Barros, F. A. de. (2011) A Multi-objective Particle Swarm Optimization for Test Case Selection Based on Functional Requirements Coverage and Execution Effort. *Tools with Artificial Intelligence (ICTAI)*, 23rd IEEE International Conference, pp.245-252.
13. Chen, T. Y., Kuo, F., Merkel, R. G., Tse, T. H. (2010) Adaptive random testing: The ART of test case diversity. *Journal of Systems and Software*, pp.60-66.
14. Engstrom, E., Skoglund, M., Runeson, P. (2008) Empirical evaluations of regression test selection techniques: a systematic review. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '08)*. ACM, New York, USA, pp.22-31.
15. Yoo, S., Harman, M. (2012) Regression testing minimization, selection and prioritization: A survey. *Software Testing Verification and Reliability*, pp.67-120.
16. Gonzales, Rafael S.; Wintz, Paul. (1987) *Digital Image Processing*. Addison-Wesley Publishing Company, USA.
17. Devore, Jay L. (2012) *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole, Boston, USA.
18. Free-software Competence Center in the University of São Paulo. URL <http://ccsl.icmc.usp.br/pt-br/projects/o-fim-oracle-images>.
19. Rothmel, G., Untch, R. H., Chengyun, C., Harrold, M. J. (2001) Prioritizing test cases for regression testing. *Software Engineering, IEEE Transactions on*, pp.929-948.
20. Mirarab, S., Akhlaghi, E. S., Tahvildari, L. (2011). Size-Constrained Regression Test Case Selection Using Multi-Criteria Optimization. *Software Engineering, IEEE Transactions on*, pp.1.