Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

Tipo de artículo: Artículo original
Temática: Pruebas de software
Recibido : 22/02/2022 | Aceptado: 31/05/2022

Implementation patterns to extend the search-based reduction Model of MTest.search

Patrones de implementación para extender el Modelo de reducción basado en búsquedas de MTest.search

Alejandro Miguel Güemes Esperón [1] https://orcid.org/0000-0001-9704-9449
Martha Dunia Delgado Dapena [2] https://orcid.org/0000-0002-2601-3462
Mailyn Moreno Espino [3] https://orcid.org/0000-0002-7613-3382
José Miguel Loor Intriago [4] https://orcid.org/0000-0002-4752-1765

[1, 2, 3] Facultad de Ingeniería Informática, Universidad Tecnológica de La Habana "José Antonio Echeverría", Cujae. Calle 114, No. 11901 entre 119 and 127, Marianao, Código Postal: 19390, La Habana, Cuba.
[4] Universidad Técnica de Manabí. Avenida Urbina y Portoviejo, Código Postal: 130105, Manabí, Ecuador.
[1] aguemes@tesla.cujae.edu.cu, [2] marta@ceis.cujae.edu.cu, [3] my@ceis.cujae.edu.cu, [4] Jmloor@utm.edu.ec

ABSTRACT

The process of generating test cases contributes to the quality of software products, detecting as many errors as possible. The design stage of test cases from test values is a challenging task, so its automation is necessary to increase the effectiveness to detect errors and reduce cost. Metaheuristic algorithms such as genetic algorithms, simulated annealing, and tabu search have been successfully applied to solve the combinatorial explosion of automatic test suite generation. The MTest.search model for automatic unit test generation has domain model extension, test, and execution mechanisms defined. In this work, implementation patterns are presented to extend the search-based reduction model. This proposal allows incorporating new configurations of the optimization problem to generate combinations of test values obtaining reduced test suites. To validate the proposed solution, case studies were defined based on the

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

analysis of extensions implemented following the patterns defined in this work and the resulting test cases. In the implemented extensions, the significance criterion of the test values and of the paths/scenarios was introduced to obtain reduced test suites with greater potential to detect errors.

Keywords: software quality; software tests; reduced test suites; metaheuristic algorithms.

RESUMEN

El proceso de generación de casos de pruebas contribuye a la calidad de los productos de software, detectando la mayor cantidad de errores posibles. La etapa de diseño de los casos de pruebas a partir de valores de prueba, constituye una tarea desafiante, por lo que es necesaria su automatización para elevar la efectividad para detectar errores y disminuir el costo. Algoritmos metaheurísticos como algoritmos genéticos, recocido simulado y búsqueda tabú se han aplicado con éxito para resolver la explosión combinatoria de la generación automática de la suite de prueba. El modelo MTest.search para la generación automática de pruebas unitarias tiene definidos mecanismos de extensión del modelo de dominio, de prueba y de ejecución. En este trabajo se presentan patrones de implementación para extender el modelo de reducción basado en búsquedas. Esta propuesta permite incorporar nuevas configuraciones del problema de optimización para generar combinaciones de valores de pruebas obteniendo suites de pruebas reducidas. Para validar la solución propuesta se definieron casos de estudios a partir del análisis de extensiones implementadas siguiendo los patrones definidos en este trabajo y de los casos de pruebas resultantes. En las extensiones implementadas se introdujo el criterio de significación de los valores de prueba y de los caminos/escenarios para obtener suites de pruebas reducidas con mayor potencialidad para detectar errores.

Palabras Clave: calidad de software; pruebas de software; suites de pruebas reducidas; algoritmos metaheurísticos.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

Introduction

Software testing is the phase of the life cycle of software development whose objective is to discover the failures that have occurred in the rest of the phases, so that they can be solved, thus contributing to the quality of the product and reducing the cost derived from such failures (PRESSMAN, 2015). However, software testing is expensive (MYERS, 2012). Due to this, it is necessary to automate the testing process in order to reduce its cost and increase its effectiveness (MYERS, 2012) (OLIINYK, 2019) (SERNA, 2019).

Within the dynamic testing processes are included the activities of design and execution of tests (PRESSMAN, 2015). One of the activities that entail an intensive labor is the generation of combinations of test values used in the construction of the test cases of the software product in the design stage (FERNÁNDEZ, 2016) (SERNA, 2017) (FELBINGER, 2018).

There are several methods for the automatic generation of test data, which try to efficiently find a small set of such data that allows satisfying a given sufficiency criterion (FERNÁNDEZ, 2016) (SERNA, 2017) (ZHANG, 2020). This reduces the cost of software testing (MYERS, 2012) (BLÉ, 2020). Currently several scientific papers address the software testing development process. The generation of paths and combinations of test values to support the design of test cases remains problematic (POLO, 2017) (VERONA, 2018) (SHARMA, 2018) (MISHRA, 2019) (BALERA, 2019) (LOOR, 2019) (ZHU, 2019) (HARIKARTHIK, 2019) (MUSA, 2019) (ASHRITHA, 2020) (AL-SAMMARRAIE, 2020) (KHARI, 2021) (PANDEY, 2021). There are different tools such as JUnit and NUnit that allow unit tests to be executed automatically, but they lack functionalities that assist the developer in the design of test cases (SERNA, 2019) (JUNIT, 2021) (NUNIT, 2021).

In (DELGADO, 2017) a model MTest.search was defined, for the automatic generation of tests based on searches. MTest.search has optimization models that reduce the test suite, for the white box test case described in (ROJAS, 2016) and for the black box test described in (MACÍAS, 2016). In addition, extension mechanisms are proposed for requirement input domains expressed through user stories (VERONA, 2018), different source code and test code languages (LARROSA, 2019) (GÜEMES, 2021), and automated support with different software components (FERNÁNDEZ, 2016) (LARROSA, 2016) (LARROSA, 2018) (GÜEMES, 2018) (LARROSA, 2019-B) (GÜEMES, 2020) (HENRY, 2021), as well as plugins for the Eclipse environment. However, the search-based reduction model of MTest.search does not have extension

mechanisms that allow incorporating new criteria for the generation of reduced test suites. The objective of this work is to propose implementation patterns that allow the extension of said model.

Materials and methods

During the software development process, models are used to represent the artifacts necessary for the description of the product being developed (SERNA, 2017) (SERNA, 2019) (ZHANG, 2020). The Model for the Automatic Generation of Tests based on Searches (DELGADO, 2017) known as MTest.search, consists of Workflows for the execution of early tests in the production environment, Optimization Models for the reduction of functional test cases and units, and Integrated Automated Tools that support the execution of workflows.

MTest.search model

MTest.search contains the following elements: 1. Domain Model, with their respective source domain description models; 2. Test Model and Reduced Test Model; 3. Search-based reduction model; 4. Execution model, with their respective destination code description models. In the upper part, the workflows for carrying out the tests are shown, as well as the tools that support the different elements that make up the model. Since the MTest.search model is made up of different models and components, it is possible to define extensions and transformations.

The flexibility, extensibility and applicability of the Model to different productive and business environments is made possible through its extension mechanisms, which allow the adaptation of its models and tools by both advanced and beginner developers. Extensions have been defined for generating unit tests from a user story template as the input domain.

In (ASHRITHA, 2020) the input domain model was extended to software requirements expressed through User Stories. The search-based model reduction allows model reduction tests (DELGADO, 2017). It is

achieved using a defined optimization model. The model starts from the input variables or attributes of the functionality to be tested, the domain description of each attribute, the level of coverage to be achieved and the number of input variables or attributes (DELGADO, 2017). From the optimization process, test cases generated from the defined criteria are obtained, which guarantee the specified level of coverage. The objective function of the optimization problem for the generation of a reduced test suite maximizes the coverage of the generated suite according to the level of coverage defined by the user (FERNÁNDEZ, 2016) (DELGADO, 2017). The heuristic function, for the evaluation of the test cases that make up the suite, will be defined according to the type of test and penalty mechanisms can also be included (FERNÁNDEZ, 2016).

There are extensions to the reduction model for white box unit tests and black box functional tests. The algorithm used to generate the combinations of values reduces the input models to discrete values using different transformations (DELGADO, 2017). In this way, the set of input values is reduced to the most significant values for the test and from there the algorithm that executes the elements proposed in the defined optimization model is applied. With this proposal, the number of values to be tested is limited without diminishing the effectiveness of the test design (DELGADO, 2017). For the determination of the transformation vectors for each type of data, loop and condition test case design techniques are used, after having generated an initial transformation vector with the techniques of equivalent partitions and limit values (ROJAS, 2016) (DELGADO, 2017). The model can be extended for the different types of tests.

Figure 1 shows the architecture of the tools that support the model. The tools are grouped into three fundamental layers: Layer “Generation of test suites”: Responsible for the generation of unit test cases regardless of the input domain and the output format; Layer “Extension”: Contains the extensions developed for various input domains and output formats. In addition, mechanisms are provided in this layer to extend to other input domains and output formats; Layer “User environment”: Contains the tools customized to the user's specific production environments. Currently there is a plug-in for the Eclipse development environment (IDE, for its acronym in English), from version 3.5, and a desktop application for generating test cases from the description of requirements.

In addition, tools have been developed that support the model and contribute to the generation of unit test cases. In (MACÍAS, 2016) a functional test case value generation component (GeVaF) was developed maximizing the coverage of the scenarios where, finally, combinations of black box test values are obtained. However, being focused on functional tests, the generated values are not associated with the independent paths of the code. In addition, in (FERNÁNDEZ, 2016) a component was developed to generate combinations of values to perform unit tests (GeVaU) where the generated combinations take into account the independent paths of the unit to be tested.

In (LARROSA, 2016) (GÜEMES, 2018) (LARROSA, 2019-B) the elements associated with the design of unit test cases were identified automatically, algorithms for the generation of independent paths were developed; the test case generator component (GeCaP) integrated the algorithms developed with other solutions for the generation of values, in such a way that the unit test cases are generated automatically (LARROSA, 2018). For the integration of GeCaP with the other components of the model, it is necessary to use different exchange files (LARROSA, 2016).

Implementation patterns to extend the search-based reduction model of MTest.search

Three implementation patterns have been defined for the proposed extension mechanisms that contribute to the implementation of the user environment and extension layers in the defined MTest.search architecture.

Pattern 1: Incorporate new criteria in the optimization model for the generation of reduced test suites.
***Problem*:** An advanced user needs to incorporate in the MTest.search support tools, new criteria to be taken into account in the configuration of the optimization problem.

***Solution*:** To add new criteria in the configuration of the optimization problem, the following steps must be taken into account:

1. Create a new component that imports the GeVaU component from the *Generador de casos de pruebas* package and its dependencies. Implement a GeneradorValores (GV) class that implements the IGenerarValores interface. To do this, the method that is responsible for executing the problem of

generating combinations of test values must be redefined, and the method that reads the information from the VariablesValores.txt files (it contains the information regarding the values generated for each variable that intervenes in the specific unit to be tested) and Caminos.txt (contains the information corresponding to each independent path of the flow control graph of the unit under test).

In case of using the Biciam class library to define an optimization model that allows generating combinations of test values:

2. Import the Biciam class library to make use of metaheuristic algorithms to generate combinations of values.
3. Implement a class that inherits from TargetFunction and implements the IGenerarValores interface. Redefine the method that is responsible for the evaluation of the generated states.
4. Implement a class that inherits from Tester and implements the IGenerarValores interface. Redefine the methods for the configuration of the optimization problem, as well as the metaheuristic algorithms to be used in the problem.
5. Implement a class that inherits from M1P and implements the IGenerarValores interface. Redefine the method in charge of generating a new state by changing the value of the variables, and the one that generates a random state.
6. Implement a class that inherits from MiCodificacion and implements the IGenerarValores interface. Redefine the methods responsible for generating a random variable from a given state.
7. Implement the necessary classes to model new criteria to be taken into account within the defined optimization model.

***Consequences*:** The application of this pattern allows generating reduced test suites taking into account the criteria incorporated in the configuration of the optimization problem.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

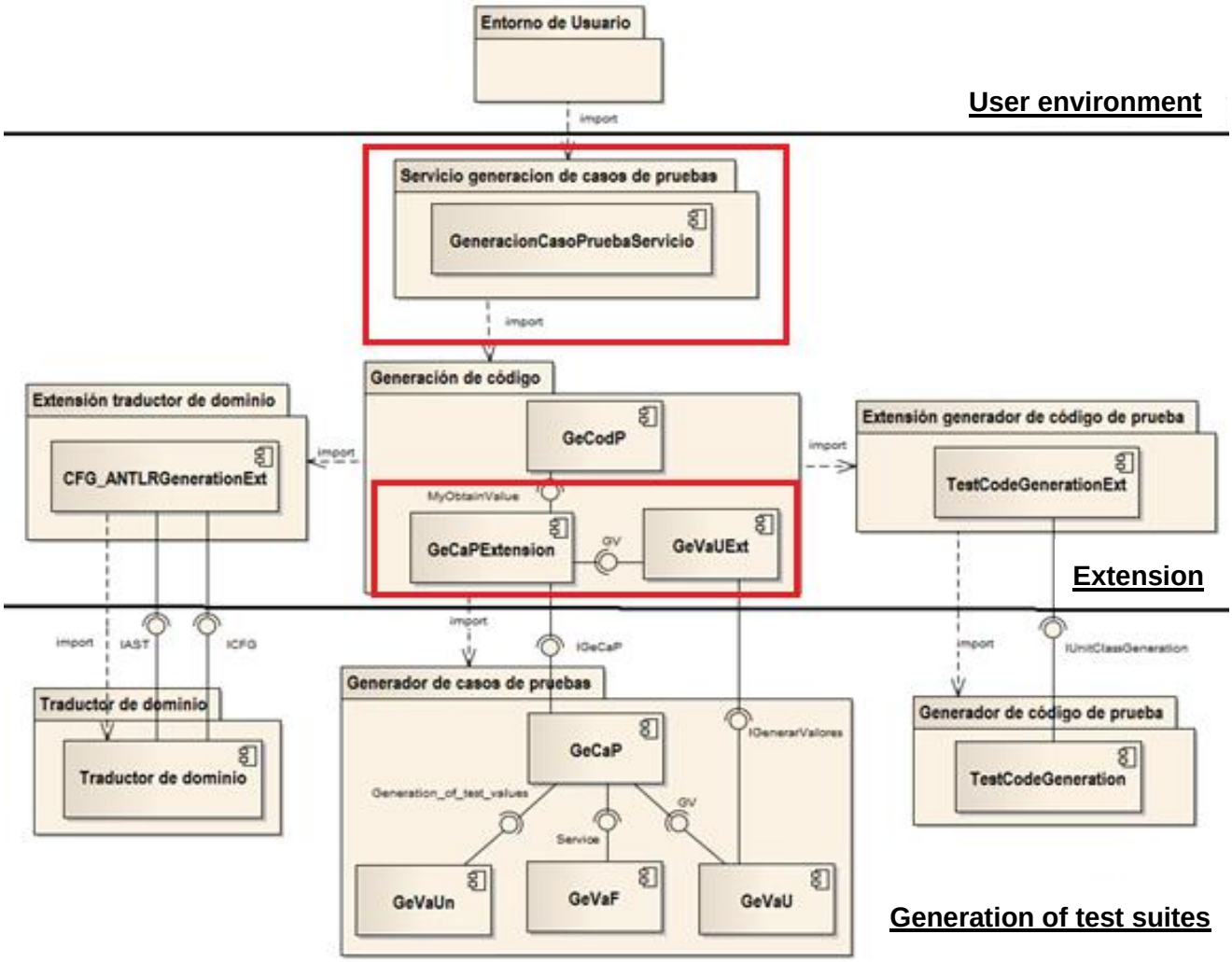

Fig. 1- Architecture of MTest.search support tools.

Pattern 2: Extension of the GeCaP component to incorporate meaning of variables and paths/scenarios in the exchange files.

Problem: An advanced user needs to incorporate new specifications of the variables and paths/scenarios for the generation of reduced test suites.

Solution: To add new specifications of the variables and/or paths/scenarios, the following steps must be taken into account:

1. Create a new component that imports the GeCaP component, the GeVaU component extension, and their dependencies.
2. Implement a class that inherits from ObtainValue and implements the IGeCaP interface. To do this, the methods in charge of:
    a. Generate the initial values of each variable
    b. Get the list of generated initial values
    c. Create the exchange file VariablesValores.txt
    d. Create the exchange file Caminos.txt

    (Incorporate the new criteria in the redefined methods)
2. Once the GeVaU and GeCaP extensions have been created, the advanced user must import the GeCodP component, in the extension layer, and redefine the method that generates the test suites/test code, taking into account the optimization model to be used for that generation.

Consequences: The application of this pattern allows generating the VariablesValores.txt and Caminos.txt exchange files incorporating the new specifications to be taken into account for the generation of the test suites.

Pattern 3: Customization of MTest.search to the user's specific production environments.

Problem: A user needs to capture the requirements or source code description and view the test cases or test methods, respectively, generated by using the MTest.search support tools.

Solution: To customize MTest.search to the user's specific production environments, the following steps must be taken into account:

1. Create a new client application that imports the GeCodP component from the *Generación de código* package and its dependencies.
2. Capture the overview of the requirement or source code under test.
3. Implement a class that uses the necessary methods to generate the test suites or test methods.

Consequences: The application of this pattern allows the creation of client applications for various productive environments. The GeCodP component guarantees the communication between the user

environment and the MTest.search components, the generation of test cases taking into account different criteria in the optimization model configurations.

Results and discussion

In order to evaluate the effectiveness of the defined patterns to generate reduced test suites, taking into account new criteria incorporated in the search-based reduction model, two case studies were designed that were driven by the following questions:

1. Is it possible to extend the search-based reduction model to incorporate new criteria in the configuration of the optimization problem using the defined patterns?
2. Is it possible to generate reduced test suites from the proposed patterns?

For this, the defined patterns were applied and the corresponding extensions were implemented. In addition, the model has been customized for the Eclipse 4.5.2 or lower development environment.

The generation of suites and test methods was carried out on a PC with the following characteristics: Intel Core i3-4160 3.60 GHz processor and 4 GB RAM. The XUnit 4 format was used, in Eclipse 4.5.2.

Question 1. *Is it possible to extend the search-based reduction model to incorporate new criteria in the configuration of the optimization problem using the defined patterns?*
For this, the necessary components that implement the interfaces of the *Traductor de dominio* subsystem were created, taking into account Pattern 1 described in the previous section to extend it to new input domains (HENRY, 2021).

For the creation of the component that generates the reduced test suites based on new criteria incorporated into the optimization model, Pattern 1 was applied (See Figure 2). Then, to extend GeCaP and incorporate new criteria in the VariablesValores.txt and Caminos.txt exchange files, it was taken into account Pattern 2 described in the previous section (See Figure 3).

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

The GeVaUExtension_3102021 component receives three files: VariablesValores.txt, Condiciones.txt and Caminos.txt, which allows the generation of the combinations of the values that cover a given path or scenario according to the conditions it presents. The VariablesValores.txt file contains the values of the input variables. This extension component incorporates new criteria to be processed by the search-based reduction model for the generation of reduced test suites.

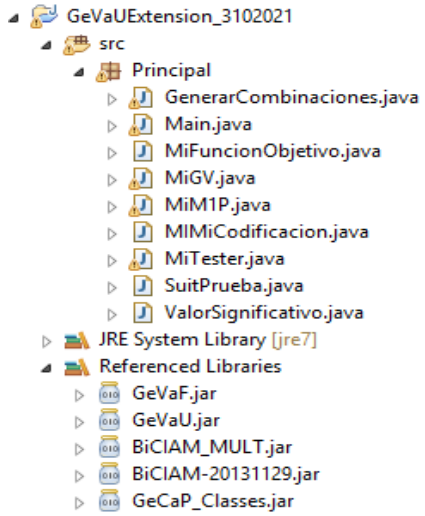

Fig. 2- Implementation view of GeVaUEExtension_3102021 component.

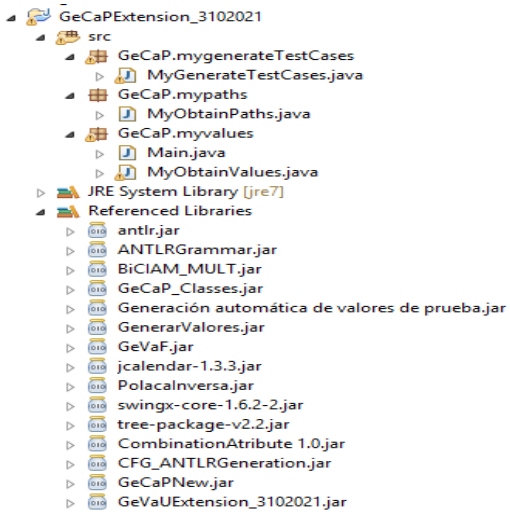

Fig. 3- Implementation view of the GeCaPExtension_3102021 component.

These criteria are: numerical significance value of each test value generated for each input variable (See Figure 4) and significance value of each test path/scenario (See Figure 5). In this extension it was defined that the significance of each path (numerical value in the interval [0, 1]) will depend on the number of conditions contained, it is the result of dividing the number of conditions in the path by the total number of conditions. Figure 5 shows the structure of the Caminos.txt file that contains the information on the independent paths.

Question 2. *Is it possible to generate reduced test suites from the proposed patterns?*

To validate the generation of reduced test suites, the description of the Reserve Product functional requirement was used, belonging to an electronic commerce project, in which 5 variables of interest

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

intervene (three of numerical domain: *cantidadMinima, cantidadMaxima, cantidadReal, cantidadAReservar;* and one of domain string of characters: idProducto). This functionality is responsible for verifying the availability of the product to be reserved. Figure 6 A shows a fragment of the VariablesValores.txt file with the values generated for the *cantidadMinima* variable and the significance assigned to each value.

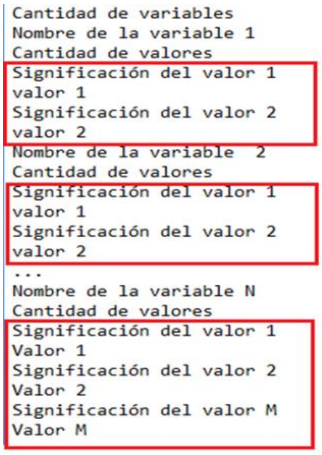

Fig. 4- Structure of the VariablesValores.txt file.

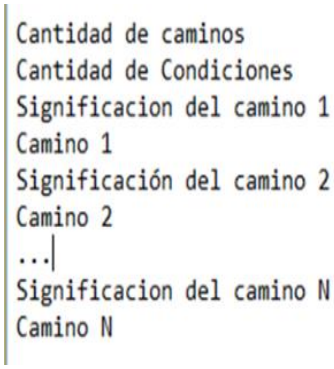

Fig. 5- Structure of the Caminos.txt file.

Based on the strategy defined to assign the significance value for each path, Figure 6 B shows the Caminos.txt file that collects the structure and significance of each path/scenario, and Figure 6 C shows the conditions involved in the paths/scenarios.

From the analysis of the significance of the values and the paths, the test cases are generated, giving priority to the paths with more conditions and using more significant values to detect a greater number of errors. Figure 7 shows a fragment of the test cases generated taking into account the criteria incorporated by the implemented extensions.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 68-84

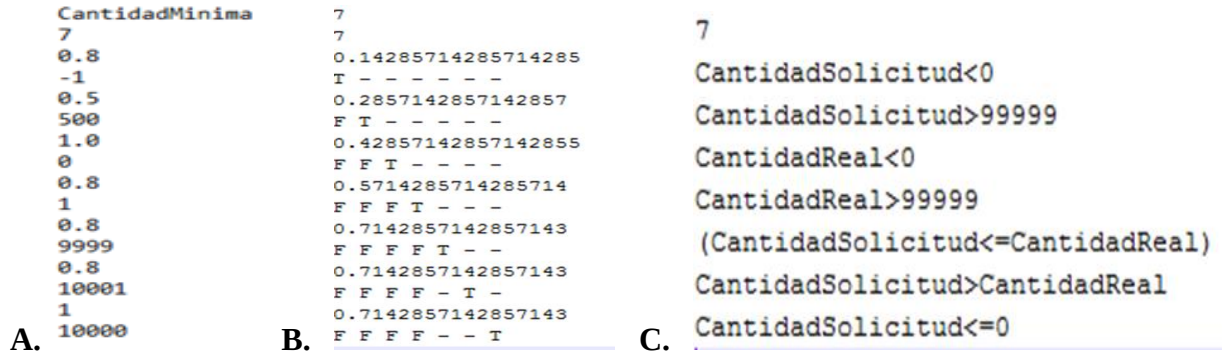

Fig. 6- Fragments of the exchange files A: VariablesValores.txt, B: Caminos.txt, C: Condiciones.txt of the Reserve Product requirement.

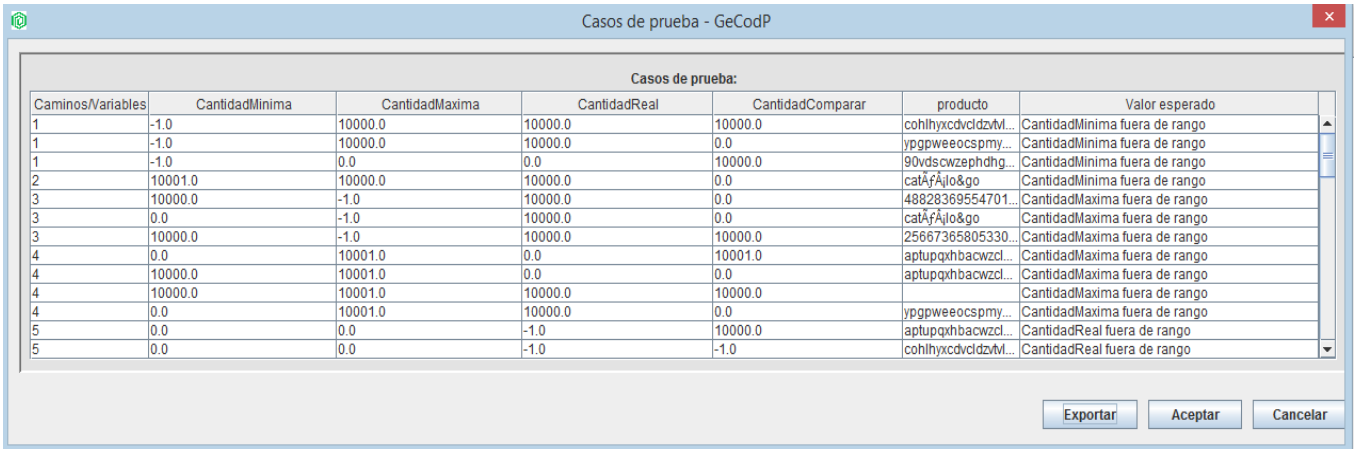

Fig. 7- Fragment of the test cases generated taking into account the significance criteria of values and paths/scenarios incorporated in the MTest.search support tools.

Conclusions

As a result of the definition of the implementation patterns that allow the search-based reduction model of the MTest.search model to be extended, new elements to be taken into account during the generation of the test cases can be incorporated, without having to modify the components support for domain, test, and execution models. In this way, the testing process is facilitated during the development of software products, since it allows automating the design of test cases and offers the possibility of extending the support components to MTest.search to define new configurations of the optimization problem, contributing

to obtain test cases that allow detecting a greater number of errors. Furthermore, these components can be integrated into different software development environments.

References

Al-Sammarraie, Hn, And Jawawi, Dn: "Multiple Black Hole Inspired Meta-Heuristic Searching Optimization For Combinatorial Testing", Ieee Access, Vol. 8, Pp. 33406-33418, 2020.

Ashritha, S. And Padmashree, T.: "Machine Learning For Automation Software Testing Challenges, Use Cases Advantages & Disadvantages", International Journal Of Science And Research Technology, Vol. 5, No. 9, Pp. 1301-1307, 2020.

Balera, Jm And Junior Va: "A Systematic Mapping Addressing Hyper-Heuristics Within Search-Based Software Testing", Information And Software Technology, Vol. 114, Pgs. 176-189, 2019.

Blé, C.: "Agile Design With Tdd", Creative Commons, 2020.

Delgado, Md, Macías, A., Larrosa, D., Verona, S. And Fernández, Pb: "Model For Automatic Generation Of Search-Based Early Tests", Computación Y Sistemas, Vol. 21, No. 3, Pp. 503-513, Issn: 2007-9737, 2017 .

Felbinger, H., Wotawa, F. And Nica, M.: "Adapting Unit Test By Generating Combinatorial Test Data", Ieee International Conference On Software Testing, Verification And Validation Workshops (Icstw), Pp. 352-355, 2018.

Fernández, Pb, Cantillo, W., Delgado, Md, Rosete A., And Yáñez C.: "Generation Of Test Value Combinations Using Metaheuristics," Revista De Ingeniería Industrial, Vol. 37, No. 2, Pgs. 200-207, 2016.

Güemes, Am, Delgado, Md, And Larrosa, D.: "Implementation Of Grammatical Structures In The Antlr Tool (Another Tool For Language Recognition) With A View To Obtaining The Flow Control Graph", 19th Scientific Convention On Engineering And Architecture , Havana, 2018.

Güemes, Am: "Components For The Genera-Tion Of Junit Test Code From Java Source Code", Degree Thesis, Technological University Of Havana "José Antonio Echeverría", Cujae, Havana, 2020.

Güemes, Am; Delgado, Md And Larrosa, D.: “Implementation Patterns To Extend Test Code Generation To New Languages In Gecodp”, Revista Cubana De Ciencias Informáticas, Vol. 15, Special No. Uciencia Ii, Pp. 93-110, Issn: 2227-1899, 2021.

Harikarthik, Sk, Palanisamy, V. And Ra-Manathan, P.: “Optimal Test Suite Selection In Regression Testing With Test Case Prioritization Using Modified And Whale Optimization Algorithm”, Cluster Computing, Vol. 22, No. 5, Pp. 11425 - 11434, 2019.

Henry H.: "Extension Component Of The Search-Based Reduction Model Of Mtest.Search", Degree Thesis, Technological University Of Havana "José Antonio Echeverría", Cujae, Havana, 2021.

Junit: Official Junit Site. Available: Www.Junit.Org , 2021.

Khari, M., Sinha, A., Herrera, E. And Cre-Spo, Rg: “On The Use Of Meta-Heuristic Algorithms For Automated Test Suite Generation In Software Testing”, Toward Humanoid Robots: The Role Of Fuzzy Sets: A Handbook On Theory And Applications, Pp. 149-197, 2021.

Larrosa D., Delgado Md, Güemes Am: “Gecodp: Tool For Generating Test Code Integrated Into Production Environments”, 22nd Ibero-American Conference On Software Engineering (Cibse 2019), P. 649-656, Isbn: 9781510887954 , Havana, 2019-B.

Larrosa, D., Fernández, Pb And Delgado, Md: “Gecap: Unit Testing Case Generation From Java Source Code”, Polibits, Vol. 57, Pp. 67-73, 2018.

Larrosa, D.: "Component For The Automatic Generation Of Unit Test Cases", Degree Thesis, Tech-Nological University Of Havana "José Antonio Eche-Verría", Cujae, Havana, 2016.

Larrosa, D.: "Extensions To Mtest.Search For The Automatic Generation Of Unit Tests In Different Languages", Master's Thesis, Technological University Of Havana "José Antonio Echeverría", Cujae, Havana, 2019.

Loor, Jm: "Prioritization Of Test Cases In Agile Development Environments", Master's Thesis, Technological University Of Havana "José Antonio Echeverría", Cujae, Havana, 2019.

Macías, A., Delgado Md, Fajardo J. And Larrosa, D.: “Functional Test Case Value Generator”, Lámpsakos, No. 15, Pp. 51-58, 2016.

Mishra, Db, Mishra, R., Das, Kn And Acharya, Aa: “Test Case Generation And Optimization For Critical Path Testing Using Genetic Algorithm”, Soft Computing For Problem Solving, Pp. 67-80, Singapore, 2019.

Musa, Ja, Romli, R. And Yusoff, N.: "An Analysis On The Applicability Of Meta-Heuristic Searching Techniques For Automated Test Data Gen-Eration In Automatic Programming Assessment", Baghdad Science Journal , Vol. 16, No. 2, Pp . 515-533, 2019.

Myers, Gj, Badgett, T., Thomas, Tm And Sandler, C.: "The Art Of Software Testing", Ed. John Wiley & Sons, United States Of America, 2012.

Nunit: Official Site Of Nunit. Available: Http://Www.Nunit.Org/ , 2021.

Oliinyk, B. And Oleksiuk, V.: "Automation In Software Testing, Can We Automate Anything We Want", Proceedings Of The 2nd Student Workshop On Computer Science & Software Engineering (Cs&Se@ Sw 2019), Pp. 224-234, Ukraine, 2019.

Pandey, A. And Banerjee, S.: "Test Suite Optimization Using Chaotic Firefly Algorithm In Software Testing", Research Anthology Recent Trends, Tools, And Implications Of Computer Programming, Pp. 722-739, Igi Global, 2021.

Polo, M., Ruiz. F., Rodríguez R. And García I.: "Test Case Generation With Regular Expressions And Combinatorial Techniques", 10th Ieee International Conference On Software Testing, Verification And Validation Workshops, Pp. 189-198, Tokyo, Japan, 2017.

Pressman R. And Maxim, B.: "Software Engineering. A Practitioner's Approach", 8th. Ed. Mcgrawhill Education, New York, 2015.

Rojas, Dm And Pérez, Z.: "Component For The Automatic Generation Of Interesting Unit Test Values Using Loop And Condition Techniques", Diploma Thesis, "José Antonio Echeverría" Technological University Of Havana, Cujae, Havana, 2016.

Serna, E., Martínez, R. And Tamayo Pa: "A Model To Determine The Maturity Of Software Test Automation As An Area Of Research And Development", C Omputation And Systems, Vol. 21, No. 2, Pp . 337-352, 2017.

Serna, E., Martínez, R., Tamayo, P. And De Envigado, Iu: "A Review Of The Reality Of Software Test Automation", Computación Y Sistemas, Vol. 23, No. 1, Pp. 169-183, 2019.

Sharma, R. And Saha, A.: "Optimal Test Sequence Generation In State Based Testing Using Month Flame Optimization Algorithm", Journal Of Intelligent & Fuzzy Systems, Vol. 35, No. 5, Pp. 5203-5215, 2018.

Verona, S.: "Automatic Generation Of Functional Test Cases From User Stories", Master's Thesis, Technological University Of Havana "José Antonio Echeverría", Cujae, Havana, 2018.

Zhang, Jm, Harman, M., Ma, L., & Liu, Y.: "Machine Learning Testing: Survey, Landscapes And Horizons", Ieee Transactions On Software Engi-Neering , 2020.

Zhu, Z. And Jiao, L.: "Improving Search-Based Software Testing By Constraint-Based Genetic Operators", Genetic And Evolutionary Computation Conference (Gecco'19), Czech Republic, 2019.

Conflicto de interés

Los autores autorizan la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Alejandro Miguel Güemes Esperón y Martha Dunia Delgado Dapena.
2. Curación de datos: Alejandro Miguel Güemes Esperón y Mailyn Moreno Espino.
3. Análisis formal: Alejandro Miguel Güemes Esperón.
4. Adquisición de fondos: Martha Dunia Delgado Dapena.
5. Investigación: Alejandro Miguel Güemes Esperón y José Miguel Loor Intriago.
6. Metodología: Alejandro Miguel Güemes Esperón.
7. Administración del proyecto: Martha Dunia Delgado Dapena.
8. Recursos: Martha Dunia Delgado Dapena.
9. Software: Alejandro Miguel Güemes Esperón.
10. Supervisión: Martha Dunia Delgado Dapena y Mailyn Moreno Espino.
11. Validación: Alejandro Miguel Güemes Esperón.
12. Visualización: Alejandro Miguel Güemes Esperón y José Miguel Loor Intriago.
13. Redacción – borrador original: Alejandro Miguel Güemes Esperón.
14. Redacción – revisión y edición: Martha Dunia Delgado Dapena y Mailyn Moreno Espino

Financiación

Universidad Tecnológica de La Habana "José Antonio Echeverría", Cujae.