Ingeniería Industrial/ISSN 1815-5936/Vol. XLI/No. 2/mayo-agosto /2020/e4114

ARTÍCULO ORIGINAL
INFORMÁTICA EMPRESARIAL

Priorización de casos de prueba en entornos de desarrollo ágil

Prioritization of proof cases in agile development settings

José Miguel Loor-Intriago
https://orcid.org/0000-0002-4752-1765
Martha Dunia Delgado-Dapena
https://orcid.org/0000-0002-2601-3462
Perla Beatriz Fernández-Oliva
https://orcid.org/0000 0002 3360 4447

[I] Universidad Técnica de Manabí
E-mail: Jmloor@utm.edu.ec
[II] Universidad Tecnológica de La Habana "José Antonio Echeverría" (CUJAE)
E-mail: marta@ceis.cujae.edu.cu, perla@ceis.cujae.edu.cu

Recibido: 1 de febrero del 2019.
Aprobado: 26 de noviembre del 2019.

RESUMEN
Este trabajo presenta una propuesta de asignación de prioridades a casos de prueba en entornos de desarrollo ágiles. Mediante métodos matemáticos, se obtiene una función para el cálculo de la prioridad de los casos de prueba de los proyectos de software y su relación con las fases del ciclo de vida de desarrollo del proyecto. La función definida se basa en cuatro indicadores cuya información es posible obtener en cada una de las iteraciones del proyecto. Se presenta un estudio de casos que permite comparar los resultados de ordenamiento de los casos de prueba atendiendo a la prioridad con su efectividad para la detección de errores potenciales.
Palabras Clave: calidad de software, pruebas de software, desarrollo ágil.

ABSTRACT
This paper presents a proposal for assigning priorities to test cases in agile development environments. Through mathematical methods, a function is obtained to calculate the priority of the test cases of the software projects and their relationship with the phases of the project development life cycle. The defined function is based on four indicators whose information is possible to obtain in each of the project's iterations. A case study is also presented that allows comparing the results of ordering of the test cases, according to the priority with their effectiveness for the detection of potential errors and get expected result.

Keywords: *Software quality, software test, agile development*

I. INTRODUCCIÓN

Cuando se desarrolla Software, aun asumiendo la magnitud del proyecto, la cantidad de casos de pruebas posibles que se pueden obtener es muy alta, resultando imposible probar todas aquellas variantes o funcionalidades. Para esto se diseñan un conjunto de casos de pruebas siguiendo los métodos, técnicas, y metodologías que permitan obtener un orden o clasificación de prioridad entre las pruebas a diseñar. Con la finalidad de poder tomar la decisión de qué casos de pruebas ejecutar antes que otros, de acuerdo al tiempo y recursos con el que se dispone.

El control de calidad, se centra en un objetivo primordial, detectar errores, pues si una prueba no cumple este objetivo es una prueba que no está bien realizada, por lo tanto no es exitosa. En Pressman, R (2010)(32) se exponen los diferentes tipos, estrategias y técnicas de diseño de pruebas.

Varios investigadores han proporcionado técnicas de Selección de Pruebas de Regresión (RTS, por sus siglas en inglés), para contribuir a la solución de estos problemas (12), tales como:: técnica de función no central modificada, técnica de minimización enfocada en la modificación y técnica de minimización enfocada en la cobertura (14).

Ahora bien, el problema de la generación de suit de pruebas reducidas tiene naturaleza combinatoria y sigue siendo objeto de múltiples investigaciones y en particular la priorización de los casos de prueba (5, 17, 19, 21).

Algunas propuestas, se enfocan en la optimización de la priorización de las pruebas, de manera que las pruebas más importantes se ejecutan primero (36; 20). Muchas de estas incluyen algoritmos agregados, programación lineal. En (48) se procura optimizar las pruebas en grupos según sus objetivos comunes. La priorización de pruebas denominadas multiobjetivo o multifaceta son aquellas que utilizan varias estrategias a la vez para conseguir ordenar las pruebas de modo que se logren uno o varios objetivos de rendimiento; en (13) se propone una estrategia multiobjetivo.

Todos los proyectos de software son diferentes, ya que poseen características que varían de acuerdo al tipo de software a desarrollar, el ciclo de vida establecido y definido por la organización, así como las necesidades del cliente. De este modo las compañías dedicadas a desarrollar software o desarrolladores pueden optar por metodologías ágiles o tradicionales para desarrollar el software.

En las metodologías convencionales, el objetivo de descubrir errores se logra mediante una serie de casos de pruebas. Las pruebas de unidad e integración se concentran en la verificación funcional de un componente y en la incorporación de componentes en una arquitectura de software. Las pruebas de validación demuestran la conformidad con los requerimientos del software y las pruebas del sistema validan el software una vez que se incorporó en un sistema más grande. Cada paso de la prueba se logra a través de una serie de técnicas de prueba sistemáticas que apoyan en el diseño de casos de prueba.

En los entornos de desarrollo ágiles, la visión del cliente cobra especial importancia y con ella la necesidad de satisfacer sus demandas y requisitos. En estos contextos la prueba de software ocupa un papel protagónico e involucra a diversos miembros del equipo, desde los desarrolladores hasta los propios clientes. En *Scrum* y *XP,* unos de sus exponentes más utilizados por la comunidad de desarrolladores ágiles, el propietario del producto colabora con el equipo y

define los criterios de aceptación para las historias de usuario en relación con el producto que se debe entregar.

Se aprecia que existe una mayor propuesta para las pruebas de regresión (14, 16; 41; 25; 46), para pruebas de sistema, (21; 45) pruebas funcionales.

Es imperativo entonces definir indicadores con el objetivo de utilizarlos para priorizar casos de pruebas en metodologías ágiles considerando una serie de indicadores (11). Dichos indicadores estarán relacionados con los intereses de los clientes, usuarios y personas interesadas en el producto, las limitaciones de tiempo, recursos humanos, presupuestos, las necesidades del negocio, las imposiciones del mercado, dependencias entre requisitos, costos de implementación, entre otros. La asignación de la prioridad podrá ser realizada por los clientes mediante encuestas, reuniones o cuestionarios; debiendo priorizar las historias de usuario. El arquitecto será el responsable de identificar y priorizar las historias de usuarios y escenarios inmersos en los requerimientos del sistema (6).

Para solucionar la problemática existente se plantea el objetivo de la investigación: Definir un método para priorizar los casos de pruebas en enfoques ágiles.

TRABAJOS RELACIONADOS

Con prácticas ágiles, utilizadas por la industria (6, 11), como el desarrollo guiado por pruebas (Test Driven Development o TDD); cada segmento del código se construye a partir de una prueba que define su comportamiento correcto. Se diseñan casos de prueba de caja blanca automatizados antes de que se produzca el código (1, 2). Antes de que un programador integre su código al código base debe haber pasado el 100% de sus casos de prueba. Para las Pruebas de Aceptación (*Acceptance Test Driven Development o ATDD*) algunas propuestas como XP promueven el uso de casos de prueba de aceptación escritas por el cliente para controlar la completitud del proyecto. Cuando un caso de prueba de aceptación ha pasado exitosamente, se considera que la funcionalidad especificada ha sido implementada apropiadamente (3, 4, 14).

En los enfoques ágiles las propuestas de priorización de las pruebas cobran especial interés, pues los períodos en que un producto es liberado a producción pueden estar entre uno y dos meses. En estos enfoques las pruebas deben ser las suficientes, más no exhaustivas ya que elevan el presupuesto y modifica la planificación del proyecto retrasando la versión entregable al cliente. Las pruebas no garantizan la ausencia de defectos, pero si debe garantizar que el producto que es liberado a producción no tenga defectos de alto impacto en sus funcionalidades. El objetivo de las pruebas no es cero defectos, sino cumplir y demostrar las expectativas del usuario final del software, siempre y cuando la ética de la persona que realiza las pruebas esté orientada al cliente y comprometida con la calidad.

II. En la actualidad existen una gran cantidad de propuesta presentadas por investigadores como (22, 23, 27, 28), dedicado a la tarea de investigar sobre la priorización de casos de pruebas ya sean propuestas con indicadores como:

el tiempo disponible a probar, importancia de requisito para el cliente
el riesgo a la ejecución de pruebas, requisito y código de cobertura, históricos de fallas
la severidad de defecto
pruebas de regresión y multiobjetivo con sus diferentes ventajas y desventajas todas dedicadas a la metodologías convencionales o tradicionales.

Se proponen diferentes técnicas, métodos o modelos para priorizar Casos de pruebas, algunas de estas propuestas se muestran en (1, 3, 14, 15).

El diseño y ejecución de las pruebas de software, y en particular la priorización de los casos de prueba dentro de la suit, cobran especial importancia en los entornos de desarrollo ágil (18, 20, 26, 27).

A partir del análisis de la bibliografía consultada, referente al tema objeto de estudio en esta tesis, se arriban a las siguientes conclusiones parciales:

Las propuestas encaminadas a la priorización de casos de prueba en entornos ágiles están encaminadas fundamentalmente a pruebas de regresión. Utilizan indicadores con datos históricos no siempre disponibles para cada caso de prueba en cada iteración considerando su carácter incremental.

Se necesita definir propuestas que prioricen los casos de prueba dando mayor importancia a la detección de errores potenciales integrando las técnicas de diseño de casos de prueba para cada tipo de prueba en particular.

II. MÉTODOS

Se ha definido un sistema de evaluación que considera los casos de prueba a ejecutar en cada iteración y su priorización dentro de esta. Para ello se han definido cuatro indicadores: $I_1, I_2, I_3 e I_4$, las cuatro funciones de evaluación de cada indicador en un caso de prueba y la función de evaluación de la prioridad de cada caso de prueba dentro de la suit correspondiente a cada iteración.

Los dos primeros indicadores I_1 y I_2 están relacionados directamente con la iteración del proceso de desarrollo y mientras que I_3 e I_4 están enfocados a las características del caso de prueba generado dentro de la suit de pruebas de la iteración.

I_1 es la prioridad del requisito en la iteración, esta puede obtenerse a partir de la información que se ofrece en los artefactos utilizados en el propio desarrollo ágil.

I_2 es la relevancia de los cambios realizados en el requisito en la iteración, es una función de la importancia y magnitud de los cambios realizados.

I_3 es la relevancia de los valores de entrada del caso de prueba dentro de la suit.

I_4 es la relevancia del escenario cubierto por el caso de prueba dentro de la suit.

Sea $f \rightarrow (0,1)$ la función de evaluación de la prioridad del caso de prueba CP_j con $1 \leq j \leq n$ y n es la cantidad de casos de prueba de la suit generada en la iteración. Teniendo la prioridad más alta 1 y la prioridad más baja 0, prioridad es el orden relativo de la ejecución de cada caso de prueba en la suit de pruebas que se puede considerar en la reducción de k suit en casos de restricciones de acuerdo con la disponibilidad de recursos para ejecutar la prueba.

$$f\left(w_i, \delta_i\left(CP_j\right)\right)_{i=1}^{4} = \sum_{i=1}^{2} w_i \frac{\delta_i\left(CP_j\right)}{Max(\delta)} + \sum_{i=3}^{4} w_i \delta_i\left(CP_j\right) \quad (1)$$

Donde:

$\delta_i(\mathrm{CP}_j)$, es la función de evaluación de los indicadores I_1, I_2, I_3, I_4 para CP_j, que en el caso de los dos primeros se corresponde con el valor del indicador dado por el equipo de proyecto en la iteración, mientras que en los dos últimos se corresponde con las funciones siguientes:

$$\delta_3\left(CP_j\right) = \frac{\sum_{l=1}^{m} \phi\left(v_l\right)}{m} \quad (2)$$

$$\delta_4\left(CP_j\right) = \gamma\left(E_{ej}\right) \quad (3)$$

$$\phi\left(v_l\right) = \frac{c\,v_l}{ct} \quad (4)$$

Donde:

m, es la cantidad de variables de entrada al caso de prueba CP_j .

$\phi(v_l)$, es la significación del valor v_l de la variable l en el caso de prueba CP_j.

$\gamma(E_{ej})$, es la significación del camino o escenario e cubierto por el caso de prueba CP_j, tal que $\sum_{e=1}^{et} Sig_e = 1$; siendo Sig_e la significación del escenario e y et la cantidad total de escenarios o caminos para la unidad bajo prueba.

m, es la cantidad de variables de entrada al CP_j.

cv_l, es la cantidad de clases de equivalencia de la variable l a las que pertenece el valor v_l del CP_j. Estas clases de equivalencia se obtienen aplicado las técnicas de diseño que corresponda en cada caso.

ct, es la cantidad total de clases de equivalencia de la variable l.

III. RESULTADOS

Como parte de la validación de la propuesta se ha diseñado un estudio de casos compuesto por dos casos y que persigue el objetivo de evaluar la similitud entre el orden de priorización de los casos de prueba utilizando el modelo propuesto y la efectividad de estos casos de prueba. La pregunta que guía este estudio es: ¿Existe similitud entre el orden relativo de los casos de pruebas y su efectividad?

El Contexto será la prueba de un componente específico.

Caso A: Generación de una suit de pruebas de caja blanca utilizando técnicas de camino básico y condiciones,

Unidad de análisis A1: mortalidad de mutantes de cada caso de prueba

Unidad de análisis A2: valor de priorización de cada caso de prueba

Caso B: Generación de una suit de pruebas de caja negra utilizando técnicas de clases de equivalencia y valores límites.

Unidad de análisis B1: mortalidad de mutantes de cada caso de prueba

Unidad de análisis B2: valor de priorización de cada caso de prueba

Para el diseño de estos casos se ha utilizado un componente que clasifica un triángulo según la longitud de sus tres lados: lado1, lado2, lado3. Se ha diseñado una suit de pruebas compuesta por 19 casos de prueba que se presentan en la Tabla 1.

Tabla 1. Diseño de la suit de pruebas

CP	LADO1	LADO2	LADO3	RESULTADO ESPERADO
1	-1	0	0	No es un triángulo. Longitudes de los lados menores o iguales a cero.
2	17	0	0	No es un triángulo. Longitudes de los lados menores o iguales a cero.
3	10	9	-1	No es un triángulo. Longitudes de los lados menores o iguales a cero.
4	36	17	10	No es un triángulo. Longitudes de los lados incorrectas.
5	17	35	18	No es un triángulo. Longitudes de los lados incorrectas.
6	10	16	26	No es un triángulo. Longitudes de los lados incorrectas.
7	12	12	12	Triángulo Equilátero.
8	16	16	10	Triángulo Isósceles.
9	10	16	10	Triángulo Isósceles.
10	17	18	18	Triángulo Isósceles.
11	17	16	6	Triángulo Escaleno.
12	5	12	20	No es un triángulo. Longitudes de los lados incorrectas.
13	0	0	0	No es un triángulo. Longitudes de los lados menores o iguales a cero.
14	25	25	25	Triángulo Equilátero.

15	20	20	14	Triángulo Isósceles.
16	18	17	7	Triángulo Escaleno.
17	1	1	1.9	Triángulo Isósceles.
18	0.1	0.1	0.1	Triángulo Equilátero.
19	0.1	0.2	0.25	Triángulo Escaleno.

Para poder estimar la efectividad de cada caso de prueba se aplicó la técnica de mutación, que permite estimar la efectividad de cada caso de prueba en función de la cantidad de mutantes muertos durante la ejecución del caso de prueba con cada uno de los mutantes (29). Mientras más mutantes mata un caso de prueba, más efectivo es.

Se diseñaron 64 mutantes, resultantes de aplicar los operadores de mutación siguientes: con AOR 28 mutantes, con LCR 4 mutantes, con UOI 22 mutantes y con ROR 10 mutantes. Adicionalmente se decidió diseñar un mutante en el que se unifican dos de los caminos del algoritmo, ambos referentes a la detección de valores incorrectos de las longitudes de los lados suministradas por el usuario, pero uno se corresponde con valores menores o iguales a cero y el otro con valores mayores que cero, pero que no cumplen la condición de que cada lado debe cumplir la condición de que su longitud sea menor que (longitudLado1 + longitudLado2 + longitudLado3) /2.

Para los 19 casos de prueba diseñados se aplicó la función de priorización. Los valores de prioridad de cada caso de prueba dependen del tipo de prueba que se está considerando, según la actividad del proceso ágil en que esta se efectúa. Como se está trabajando con un único requisito se asigna un único valor a los indicadores $I_1=2$ e $I_2=5$.

Los valores de prioridad que se obtienen con la aplicación de la propuesta se muestran en la Tabla 2. La efectividad de los casos de prueba se mide a través de la cantidad de mutantes que mató cada caso de prueba mediante la prueba de caja blanca ejecutada. En la Tabla 2 se muestra la mortalidad y efectividad de los casos de prueba.

Tabla 2. Valores de prioridad para la prueba de aceptación

CP	f(S) Caja blanca	f(S) Caja negra	Mutantes muertos	Efectividad
1	0,602	0,607	65	1
2	0,602	0,607	65	1
3	0,570	0,575	65	1
4	0,570	0,575	17	0.262
5	0,570	0,575	15	0.230
6	0,570	0,575	13	0.2
7	0,575	0,625	36	0.554
8	0,575	0,625	32	0.492
9	0,575	0,625	33	0.508
10	0,575	0,625	34	0.523
11	0,575	0,625	30	0.462
12	0,570	0,575	56	0.862
13	0,620	0,625	65	1
14	0,575	0,625	36	0.554
15	0,575	0,625	34	0.523
16	0,575	0,625	30	0.462

17	0,642	0,692	35	0.538
18	0,675	0,725	40	0.615
19	0,675	0,725	33	0.508

Caso A: Generación de una suit de pruebas caja blanca utilizando técnicas de camino básico y clases de equivalencia.

Para analizar los resultados en el caso de la prueba de caja blanca es necesario agrupar los casos de prueba según el camino que cubre cada uno, pues la priorización será útil para seleccionar una suit en la que se garantice cobertura de los caminos.

Haciendo el análisis de los valores de prioridad para alcanzar una cobertura de las ramas o caminos del 100% se debe seleccionar un caso de prueba por cada camino, en este caso el que mayor nivel de prioridad obtuvo. Los conjuntos de los casos de prueba que corresponden a cada uno de los 11 caminos son:

$\{Cp\_1, Cp\_12, Cp\_13\}\{Cp\_2\}$
$\{Cp\_3\}\{Cp\_4\}\{Cp\_5\}\{Cp\_6\}$
$\{Cp\_7, Cp\_14, Cp\_18\}\{Cp\_8, Cp\_15, Cp\_17\}$
$\{Cp\_9\}\{Cp\_10\}\{Cp\_11, Cp\_16, Cp\_19\}$

A partir de los valores de prioridad se selecciona el caso de prueba con mayor valor de prioridad en cada uno d ellos conjuntos anteriores, Por tanto, la Suit de prueba queda conformada con los casos de prueba 13, 2, 3, 4, 5, 6, 18, 17, 9, 10, y 19 que satisfacen respectivamente cada uno de los 11 caminos.

Caso B: Generación de una suit de pruebas caja negra utilizando técnicas de satisfacción de escenarios y clases de equivalencia.

Para analizar los resultados en el caso de la prueba de caja negra es necesario agrupar los casos de prueba según el escenario que cubre cada uno, pues la priorización será útil para seleccionar una suit en la que se garantice cobertura de los escenarios.

Haciendo el análisis de los valores de prioridad para alcanzar una cobertura de los escenarios del 100% se debe seleccionar un caso de prueba por cada escenario, en este caso el que mayor nivel de prioridad obtuvo. Los conjuntos de los casos de prueba que corresponden a cada uno de los 4 escenarios son:

$\{Cp\_1, Cp\_2, Cp\_3, Cp\_4, Cp\_5, Cp\_6, Cp\_13\}$
$\{Cp\_7, Cp\_14, Cp\_18\}$
$\{Cp\_8, Cp\_9, Cp\_10, Cp\_15, Cp\_17\}$
$\{Cp\_11, Cp\_16, Cp\_19\}$

A partir de los valores de prioridad se selecciona el caso de prueba con mayor valor de prioridad en cada uno d ellos conjuntos anteriores, Por tanto, la Suit de prueba queda conformada con los casos de prueba 13, 4, 18, 17 y 19 que satisfacen respectivamente cada uno de los 11 caminos.

Como se puede apreciar los casos de prueba seleccionados en el Caso A y el Caso B son los que mayor valor de prioridad tienen entre los casos de prueba de cada camino y escenario respectivamente; ellos coinciden con los de mayor efectividad en cada camino o escenario.

En las figuras 1, 2, 3 y 4 se muestra el orden relativo de los casos de pruebas que satisfacen los caminos 1, 7, 8 y 11 respectivamente, tanto por método de priorización como por la efectividad. El resto de los caminos no se analizan porque solo hay un caso de prueba que los satisface.

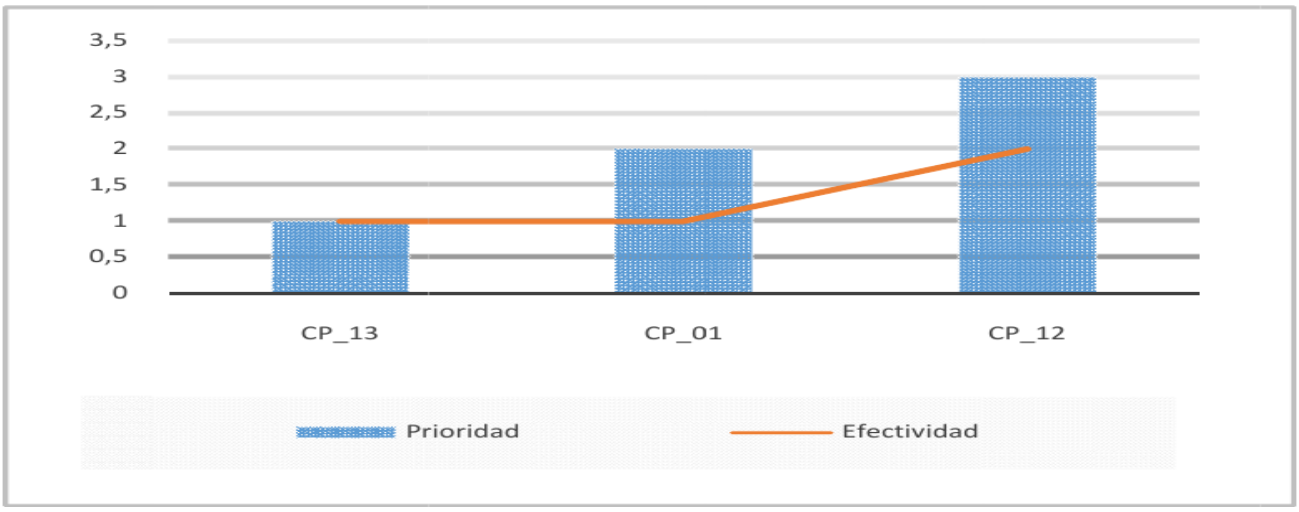

Fig.1. Resultados para las pruebas de caja blanca camino 1

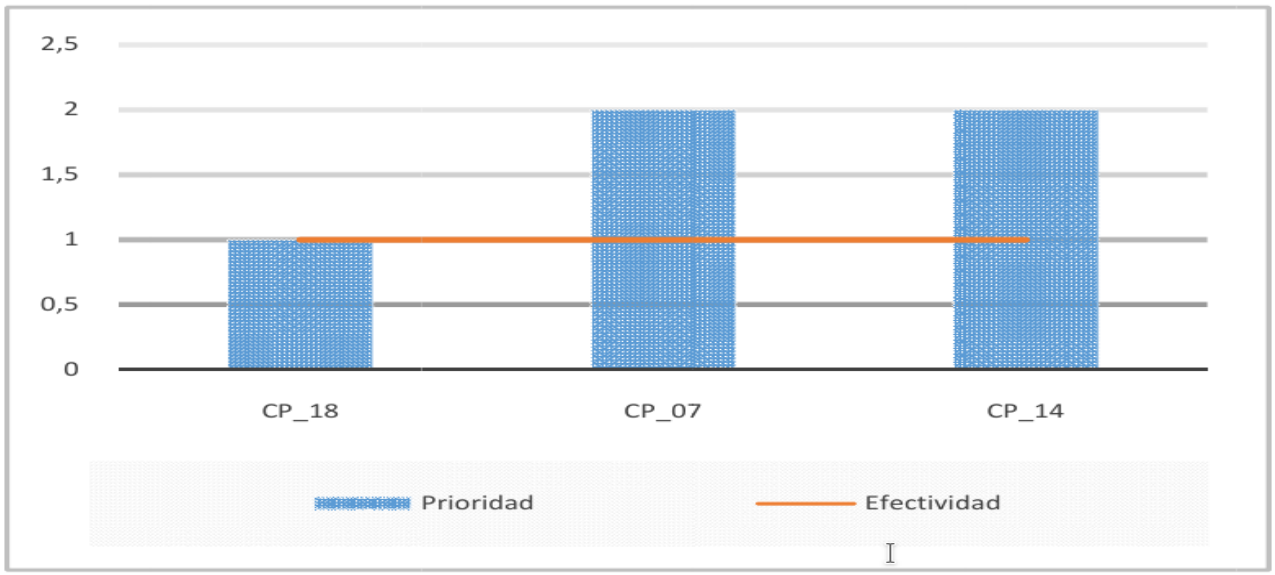

Fig 2. Resultados para las pruebas de caja blanca camino 2

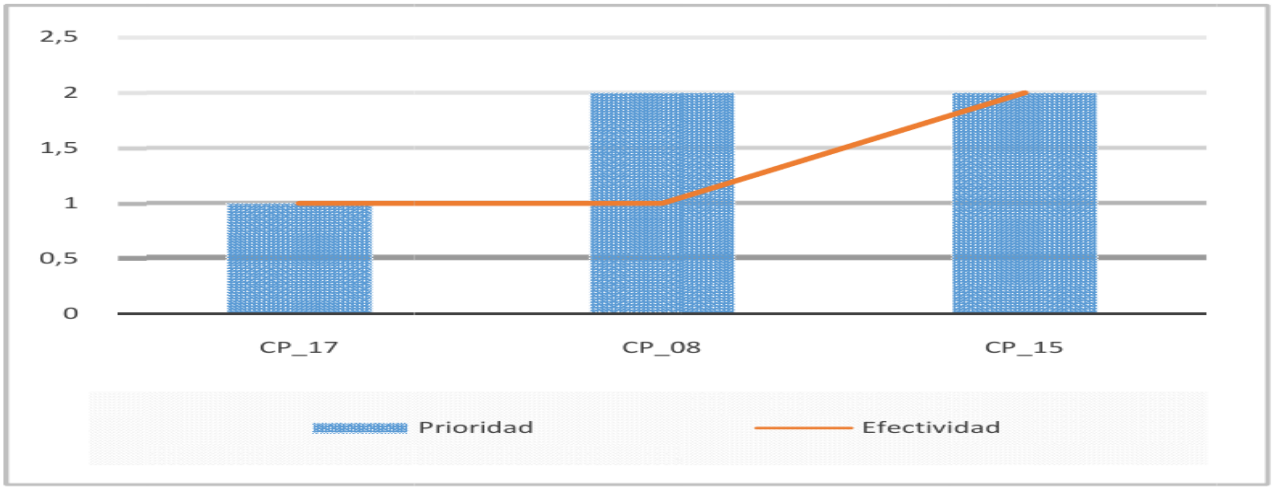

Fig. 3. Resultados para las pruebas de caja blanca camino 8

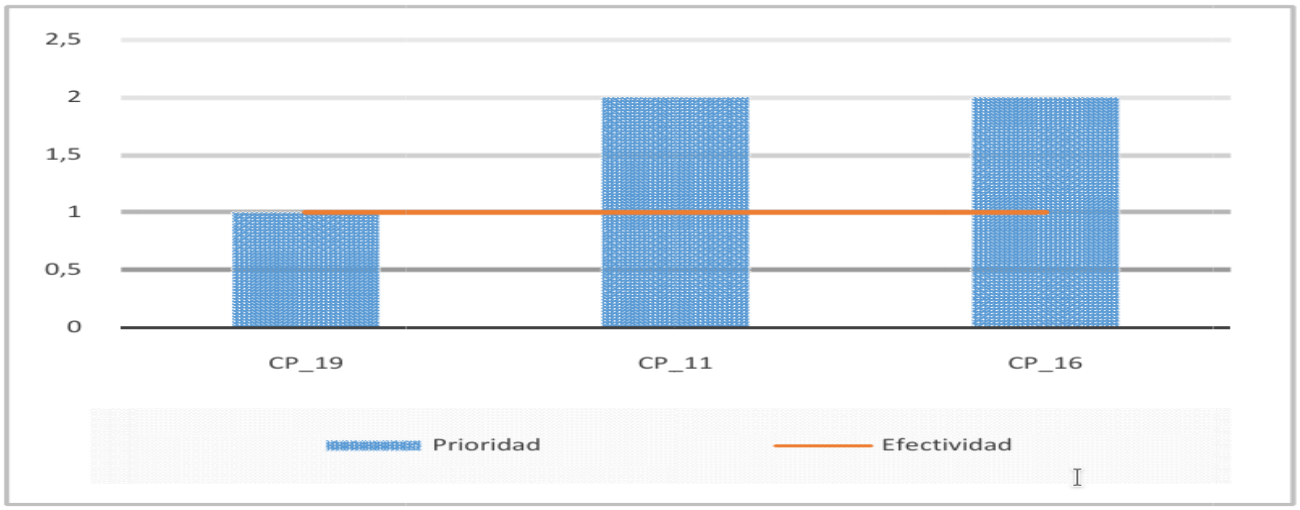

Fig. 4. Resultados para las pruebas de caja blanca camino 11

En las figuras 5, 6, 7 y 8 se muestra el orden relativo de los casos de pruebas que satisfacen los escenarios 1, 2, 3 y 4 respectivamente, tanto por el método de priorización como por la efectividad.

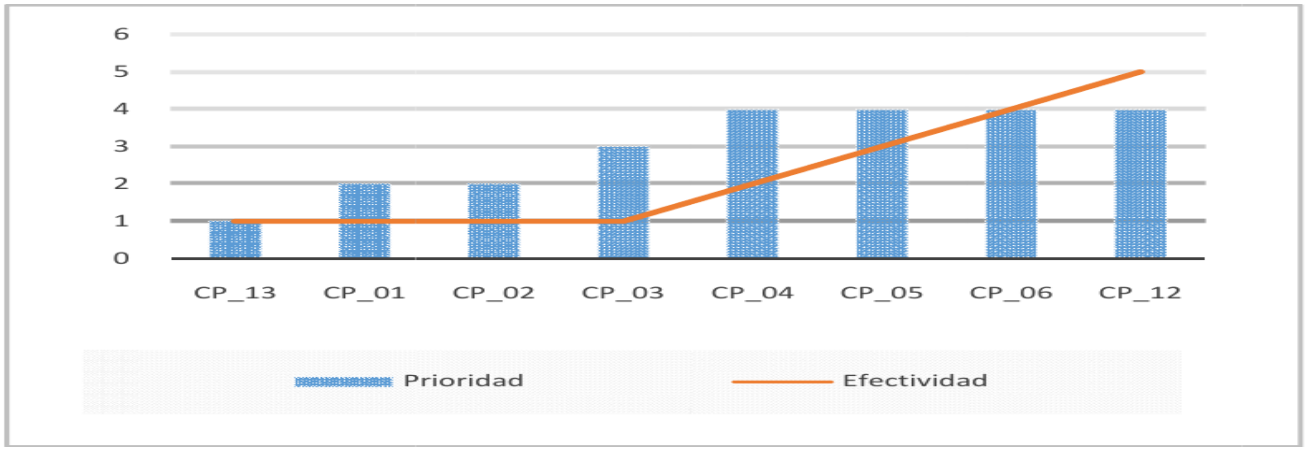

Fig. 5. Resultados para las pruebas de la caja negra escenario 1

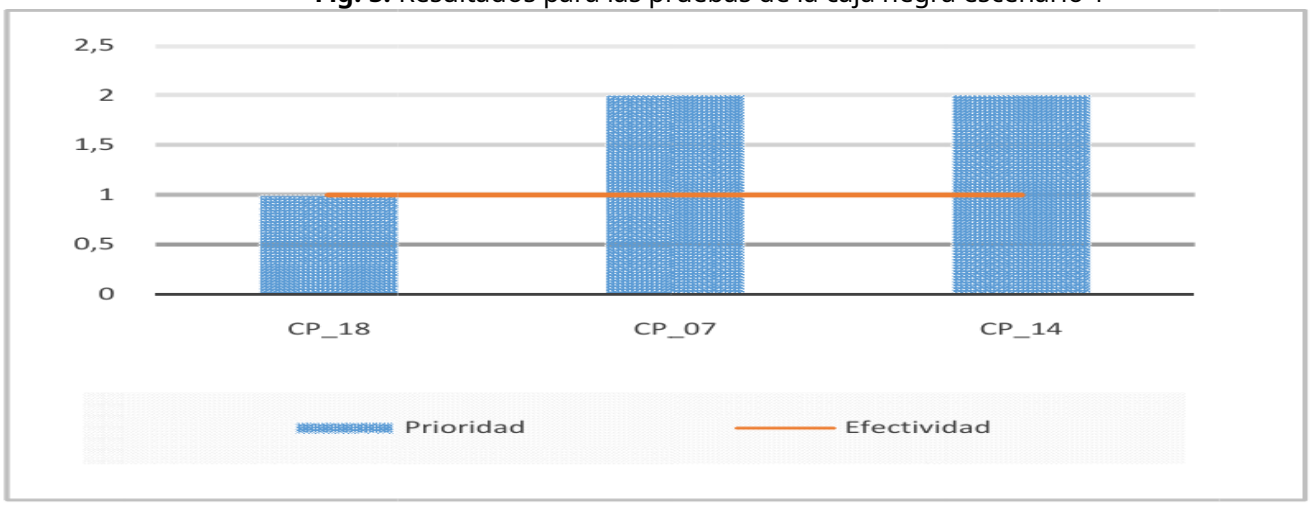

Fig. 6. Resultados para las pruebas de la caja negra escenario 2

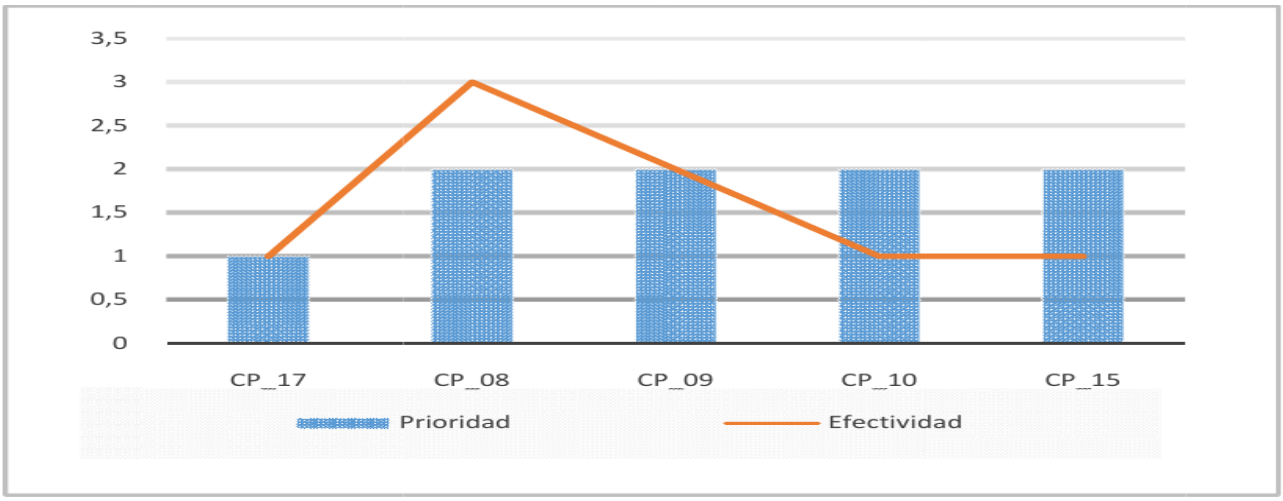

Fig. 7. Resultados para las pruebas de la caja negra escenario 3

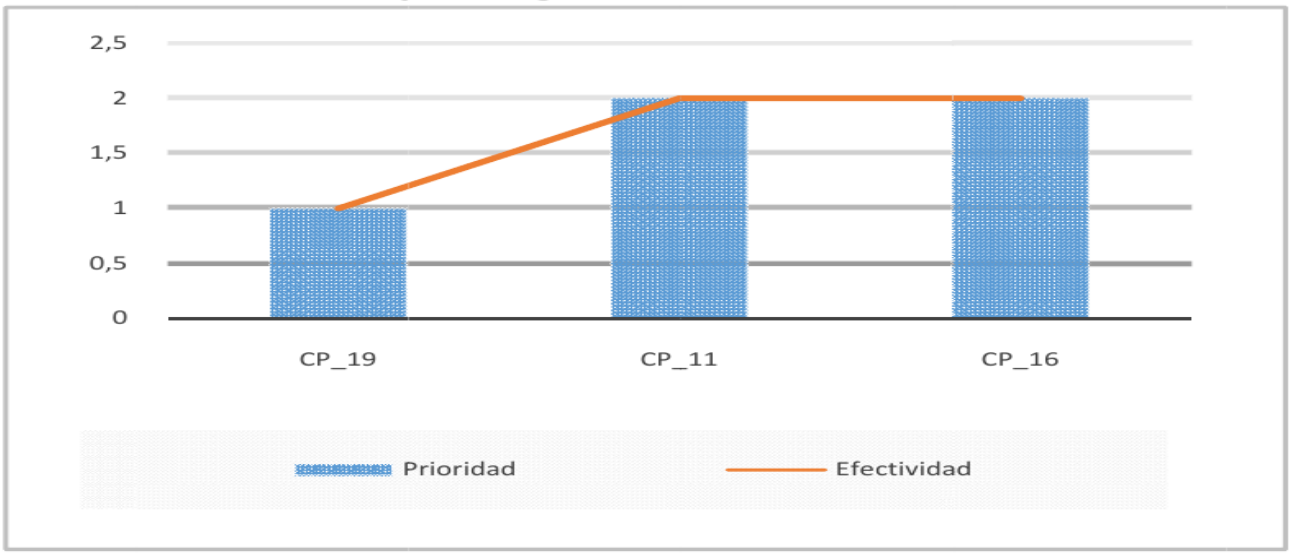

Fig. 8. Resultados para las pruebas de la caja negra escenario 4

IV. DISCUSIÓN

A partir A partir de el análisis de los resultados obtenidos con la aplicación del método y contrastando los casos de prueba que son seleccionados, para cada camino y escenario en cada uno de los casos de estudio; se puede constatar:

La efectividad del método de priorización propuesto es mayor que la de las soluciones anteriores. Esto se puede observar analizando que da mayor prioridad a los casos de prueba cuyas combinaciones de valores de entrada pueden ayudar a encontrar mayor cantidad de errores potenciales.

La solución propuesta de la da mayor prioridad a los casos que mayor cantidad de errores detecte y se lo está dando por la calidad de valores de entrada.

Un buen valor de entrada es priorizado en la medid da que ese valor permite encontrar más errores. Para que un u valor de e entrada detecte d mayor cantidad de errores tiene que ser conformado aplicando las diferentes técnicas de ingeniería de software.

Con la misma cantidad de esto os valores se detecta mayor cantidad de e errores por lo tanto en este estudio se demuestra a que se puede reducir la suit de prueba si se escogen estos valores significativos.

V. CONCLUSIONES

1. Se han definido indicadores de priorización que han sido integrados en una función para priorizar los casos de prueba en una suit dentro de una iteración.
2. Los resultados del estudio de casos permiten constatar que las prioridades asignadas por la función propuesta coinciden con los casos de prueba que más potencialidades tienen de detectar errores.
3. Para trabajos futuros debe realizarse el ajuste de los pesos de la función definida y hacer una experimentación con suit de pruebas de diferentes tamaños.

VI. REFERENCIAS

1.Anand, S., E. K. Burke, et al. "An orchestrated survey of methodologies for automated software test case generation." The Journal of Systems and Software 86(8) pp: 1978– 2001, 2013.

2.Chen, T., X.-s. Zhang. "State of the art: Dynamic symbolic execution for automated test generation." Future Generation Computer Systems 29(7): 1758-1773." 2013.

3.Delgado, M. D.; Macias, A.; Larrosa, D. "Model for Automatic Generation of Search-Based Early Test." Computación y Sistemas 21(3): 503-513, 2017.

4.Fernández, P. "Modelo para la generación automática de combinaciones de valores de pruebas unitarias." Tesis de Maestría, Instituto Superior Politécnico José Antonio Echeverría, La Habana, Cuba, 2016.

5.Gao, D.; Zhao, L. "Test Case Prioritization for Regression Testing Based on Ant Colony Optimization". 6thm Internetional Conference on Software engineering and Service Science ICSESS, Beijing, China, IEEE, pp. 275-279, 2015.

6.García, J. M. "Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos de software Universidad de Oviedo, Trabajo de fin de Master, 2015.

7.Goyal, A. et al. "Systematic Review on Quality Assessment of Test Case Prioritization Techniques in Software Testing.", 2014.

8.Gupta, S. et al. "A novel approach for test case prioritization" International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012.

9.Jatain, A. et al. "A Systematic Review of Techniques for Test Case Prioritization". International Journal of Computer Applications, 2013, (68)2:38-42. ISSN: 0975-8887.

10.Kaur, A. et al. "A New Technique for Test Case Prioritization" International Journal of Computer Science and Mobile Computing, 2013.

11.Kaur, K. "Applying agile methodologies in industry projects: Benefits and challenges", 1st International Conference on Computing, Communication, Control and Automation, ICCUBEA 2015 ISSN: 832-836, 2015.

12.Kim, S. and Baik, J. "An effective fault aware test case prioritization by incorporating a fault localization technique." ESEM '10 Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.", 2010.

13.Kumar, H. et al. "A Hierarchical System Test Case Prioritization Technique based on Requirements." 13 th Annual International Software Testing Conference in India, 2013 04 – 05, Bangalore, India, December 2013.

14.Larrosa, D. et al. "Diseño y ejecución de pruebas unitarias en diferentes lenguajes. VIII Taller Internacional de Calidad en las Tecnologías de la Información y las Comunicaciones. La Habana, Cuba, 2018.

15.Macías, A. et al. "Generador de valores de casos de pruebas funcionales." Lámpsakos(15): 51-58, 2016.

16.Manika, T. and Malhotra, S. "An Approach for Test Case Prioritization Based on Three Factors I.J. Information Technology and Computer Science, 2015, 04, 79-86 Published Online March 2015 in MECS, 2015.

17.Muthusamy, T. et al. "Effectiveness Of Test Case Prioritization Techniques Based On Regression Testing". International Journal of Software Engineering & Applications (IJSEA), 2014, 5(6), 113-123. ISSN: 0975-9018, 2014.

18. Polo, M. and Ruiz, F. "Test Case Generation with Regular Expressions and Combinatorial Techniques". 10th IEEE International Conference on Software Testing, Verification and Validation Workshops, Tokyo, Japón, IEEE, 2017.ISBN: 978-1-5090-6676-6

19.Prakash, N. and Rangaswamy, R. "Potentially Weighted Method for Test Case Prioritization."Journal of Computational Information Systems 9(18): ISSN 7147-7156, 2013.
20. Rajendrani, M. "A survey on different approaches for software test case prioritization Journal of King Saud University Computer and Information Sciences, 2018.
21. Roongruangsuwan, S. and Daengdej, J. "Test Case Prioritization Techniques". Journal of Theoretical and Applied Information Technology,2010;(18)2:45-60. ISSN 1992-8645.
22.Rothermel, G. et al. "Prioritizing Test Cases For Regression Testing." IEEE Transactions on Software Engineering.2001; (27)10.ISSN: 0098-5589
23.Ruchika, M. et al. "A Regression Test Selection and Prioritization Technique" Journal of Information Processing Systems 6 (2): 2010. ISSN 1976-913X
24.Sakti, A. and Pesant, G. "JTeXpert at the SBST 2017 tool competition. SBST '17 Proceedings of the 10th International Workshop on Search-Based Software Testing, Buenos Aires, Argentina, ACM, 2017.
25.Singhal, H. and Tyagi, K. "An Evolution of Test Case Prioritization Techniques". International Journal of Computer Applications, 2015, (130)1: 33-37. ISSN: 0975 – 8887, 2015.
26.Sultan, Z. and Nazir, S. "Analytical Review on Test Cases Prioritization Techniques: An Empirical Study". International Journal of Advanced Computer Science and Applications (IJACSA), (8) 2: 293-302. ISSN 1005-8885, 2017.
27.Vani, B. et al. "Developing Prioritization Criteria and Determining FDD of t-way Combinatorial Interaction Test for Web Based Application", The International Daily journal, 30(123), 132-138 ISSN 2278 – 5469 EISSN 2278 – 5450, Discovery Publication, 2015.
28.VijayaKumar, C. et al. "Prioritization of Functional Test Suites Using Closed Dependency Structures International Journal of Advanced Research in Computer and Communication Engineering, (4)1, January 2015.
29.Zhang, Z. and Yan, J. "Generating combinatorial test suite using combinatorial optimization." The Journal of Systems and Software, 98: 191-207, 2014.

Los autores declaran que no hay conflictos de intereses de ningún tipo

Contribución de cada autor

José Miguel Loor-Intriago: Participó en la revisión del artículo y conformación del estado del artey la propuesta del método para la priorización de los casos de pruebas. Así como el diseño, la implementación y la ejecución de los casos de estudio experimentales.

Martha Dunia Delgado-Dapena: Realizó el diseño de la investigación y colaboró en la formulación del modelo matemático para la propuesta de priorización de los casos de pruebas. Apoyó a la revisión del estado del arte, la redacción y revisión de de la versión final del artículo.

Perla Beatriz Fernández-Oliva: Participó en el d de la investigación y colaboró en la formulación del modelo matemático para la propuesta de priorización de los casos de pruebas. Apoyo a la revisión del estado del arte, la redacción y revisión de la versión final del artículo.