Automatic Application of the Category-Partition Method with Loops Support over Functional Requirements

Aplicación Automática del Método de Categoría-Partición a Requisitos Funcionales con Soporte Para Bucles

Javier J. Gutiérrez, María J. Escalona, Manuel Mejías, Isabel M. Ramos

Dpto. de Lenguajes y Sistemas de Información
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla
Avd. Reina Mercedes s/n. 41012. España
{javierj, escalona, risoto, ramos}@lsi.us.es

Claudia P. Gómez

Instituto de Informática
Facultad de Ciencias Exactas, Físicas y Naturales
Universidad Nacional de San Juan
Av. Ignacio de la Roza 590 (o). San Juan Argentina
cpgomez@iinfo.unsj.edu.ar

ABSTRACT

This paper extends previous works about the automatic generation of test cases. The start point id the Category-Partition method and the extended use case pattern. This paper describes the automatism of both techniques and extends them to manage loops in use cases.

Keywords

Test case, Activity diagram, Category-Partition Method.

RESUMEN

Con este artículo se amplían trabajos anteriores, para la generación de pruebas a partir de casos de usos mediante un proceso automático generando así los objetivos de prueba. Se aplica para tal fin el método de Categoría – Partición, y el Patrón Extendido de Casos de Usos, sin embargo esta vez se apunta a los casos de usos que presentan bucles.

1. INTRODUCCIÓN

Actualmente, una de las técnicas más empleadas para la especificación funcional de sistemas software son los casos de uso. De manera muy resumida, un caso de uso es la descripción de una interacción entre el sistema y uno o varios actores para la consecución de un resultado. A pesar de que UML ofrece una notación para la definición de casos de uso (compuesta principalmente por diagramas de casos de uso asociados a diagramas de actividades y diagramas de estados), en la actualidad se utilizan con gran frecuencia técnicas de definición de casos de uso basadas en plantillas y lenguaje no formal (como por ejemplo [6] y [9]). Esto supone un reto adicional para los procesos automáticos de generación de pruebas a partir de casos de uso.

En este trabajo, el concepto de prueba de caso de uso hace referencia a pruebas de caja negra para verificar la satisfacción de los requisitos funcionales del sistema a prueba [5]. Actualmente, existen varias propuestas que describen cómo generar casos de prueba. Dos estudios comparativos de dichas propuestas pueden consultarse en [8] y en [13]. En total, en ambos estudios, hay un total de 22 propuestas diferentes. Sin embargo, la gran mayoría de dichas propuestas pueden agruparse en dos grandes técnicas: el análisis de escenarios (en dónde se identifican casos de prueba a partir de los escenarios de un caso de uso) y el método de categoría-partición (en dónde se identifican casos de prueba definiendo valores concretos para todo elemento que puede tener varios valores en un caso de uso).

En dos trabajos anteriores, se presentó la automatización de ambas técnicas. En concreto, en [15], presentamos un proceso sistemático y una herramienta de soporte para la construcción de un diagrama de actividades a partir de un requisito en lenguaje no formal y la identificación de caminos de dicho diagrama como posibles casos de prueba. En [12] presentamos otro proceso sistemático y otra herramienta (ambos complementarios al primer trabajo) para la generación de variables operacionales, dominios, restricciones y el cálculo de combinaciones de valores

válidas (esto es, combinaciones que satisfacen las restricciones). Dicho trabajo es una sistematización e implementación del método de categoría-partición [1], [17] y variables operacionales [3].

Sin embargo, este último trabajo tenía una importante limitación que era la imposibilidad de trabajar con bucles. En el contexto de casos de uso, un bucle es una repetición de una parte de un caso de uso, por ejemplo, cuando un actor introduce una información incorrecta y el sistema la solicita de nuevo.

La principal aportación de este trabajo es la mejora del proceso descrito en [12] para trabajar con bucles. Para ello, ha sido necesario definir conceptos adicionales. Como instancias de variables operacionales y restricciones de bucle, que se explicarán en detalle en las siguientes secciones. Así mismo, se presenta una modificación de la herramienta de soporte de este proceso con las nuevas incorporaciones.

La organización de este trabajo se describe a continuación. En la sección 1.1 se definen términos utilizados a lo largo de este trabajo. En la sección 2 se resumen brevemente nuestros trabajos anteriores, los cuáles son el punto de partida para la propuesta que presentamos en la sección 3. En esa sección, se describe la técnica para generar secuencias de manera automática, haciendo especial hincapié en las modificaciones desarrolladas para el soporte de los bucles. Seguidamente, en la sección 4 se comentan brevemente otros trabajos relacionados. Finalmente, en la sección 5, se exponen las conclusiones.

1.1 Definición de Términos

Categoría: Ver variable operacional.
Combinación de valores: Una combinación de valores es una secuencia concreta de instancias de variables operacionales junto con el dominio en el que ha tomado el valor en cada una de esas instancias.
Partición: Subconjunto del dominio de una variable operacional, con la característica de que, para todos los valores de dicho conjunto, el sistema exhibe el mismo comportamiento.
Restricción de bucle: Una restricción de bucle es una estructura de información que indica la variable y partición de la cuál debe tomar su valor para que ocurra un bucle y, además, indica todas las variables operacionales que se evaluarán de nuevo como consecuencia del bucle.
Restricción de no valor: Una restricción de no valor es una expresión similar a una estructura condicional en lenguajes de programación. Esta estructura indica que si una variable (o un conjunto) toma valores en una de sus particiones, entonces no es necesario que otras variables tomen ningún valor. Este tipo de restricciones permiten reducir el número de combinaciones obtenidas eliminando aquellas que no puede realizarse en la práctica.
Variable operacional: Tanto el término variable operacional como el término categoría hacen referencia al mismo concepto por lo que pueden utilizarse indistintamente. En este trabajo nos decantamos por el término variable operacional ya que se acopla mejor al concepto y al término instancia de variable definido más abajo. Una variable operacional es cualquier fragmento de un caso de uso que puede variar entre dos ejecuciones del mismo caso de uso. En un trabajo previo [16], se presentó una clasificación para las variables operacionales en tres tipos: variables de actor, de sistema y de información. Una variable de actor define un conjunto de operaciones que el actor tiene disponible, y entre las que elegirá, durante la realización de un caso de uso. Una variable del sistema define un estado interno del sistema bajo prueba cuyo valor condiciona la realización de un caso de uso. Finalmente, una variable de información, define un flujo de información desde un actor externo al sistema (variable de información de entrada) o desde el sistema a un actor externo (variable de información de salida). Otro aporte interesante es la definición de un conjunto de heurísticas para la clasificación automática de las variables operacionales encontradas.
Instancia de variable operacional. Una instancia de una variable operacional es una evaluación de dicha variable operacional. Como se verá más adelante, una variable operacional puede ser evaluada distintas veces como consecuencia de un bucle, y en cada evaluación puede presentar un valor distinto.

2. UN RESUMEN DE LA GENERACION DE CASOS DE PRUEBA A PARTIR DE ESCENARIOS DE CASOS DE USOS

En las referencias [13], [14] y [15] se explica en detalle el proceso, modelos y algoritmos para generar un diagrama de actividades a partir de un caso de uso y obtener objetivos de prueba definidos como recorridos sobre el diagrama de actividades.

Para la implementación automática del método de categoría-partición presentada en este trabajo, es necesario contar con el diagrama de actividades correspondiente al caso de uso para obtener las variables y particiones y también son necesarios los escenarios del caso de uso (expresados como caminos del diagrama de actividades) para la identificación de las restricciones. Por este motivo, en los siguientes párrafos se resumen las ideas fundamentales.

A lo largo del proceso, se mostrará un ejemplo tomado a partir de un caso de uso de un sistema de gestión de eventos on-line desarrollado por la empresa CodeCharge (www.codecharge.com) y utilizado con su permiso. Como modelo de plantilla de caso de uso hemos utilizado el modelo propuesto en la metodología Navigational Development Technique (NDT) [9], la cual ofrece un modelo de requisitos completo, formal y flexible. NDT propone una plantilla tabular para definir requisitos funcionales mediante casos de uso. Hemos elegido el modelo de requisitos y las plantillas de NDT por dos motivos principales. En primer lugar, este modelo está basado en un metamodelo formal definido mediante UML [9]. En segundo lugar, el modelo de requisitos de NDT ha sido aplicado en proyectos reales y complejos, con resultados muy exitosos [10] y [11]. Para su procesamiento automático, se ha almacenado la información relevante del caso de uso en un documento XML de la figura 1 (el DTD correspondiente puede descargarse de la página de la herramienta)

```
<useCase id="01-AddLink">
    <description>This use case allows to introduce a new link.</description>
    <precondition>No.</precondition>
    <postcondition>A new link is stored.</postcondition>
    <mainSequence>
        <step id="1">The visitor selects the option for introduce a new link.</step>
        <step id="2">The system recovers all the stored categories and it asks for the information of a link.</step
        <step id="3">The visitor introduces the information of the new link.</step>
        <step id="4">The system stores the new link.</step>
    </mainSequence>
    <alternativeSteps>
        <astep id="3.1">At any time, the visitor may cancel this operation then the use case ends.</astep>
    </alternativeSteps>
    <errorSteps>
        <estep id="2.1">If there was an error recovering the categories, then the system shows an error message
            and this use case ends.</estep>
        <estep id="2.2">If there were not categories found, then the system shows and error message and this
            use case ends.</estep>
        <estep id="3.2">If the link name, category or URL are empty, then the system shows an error message with
            the result of repeat step 2.</estep>
        <estep id="4.1">If there is an error storing the link, then the system shows an error message and this
            use case ends.</estep>
    </errorSteps>
</useCase>
```

Figura 1: Caso de Uso de ejemplo para la búsqueda de enlaces.

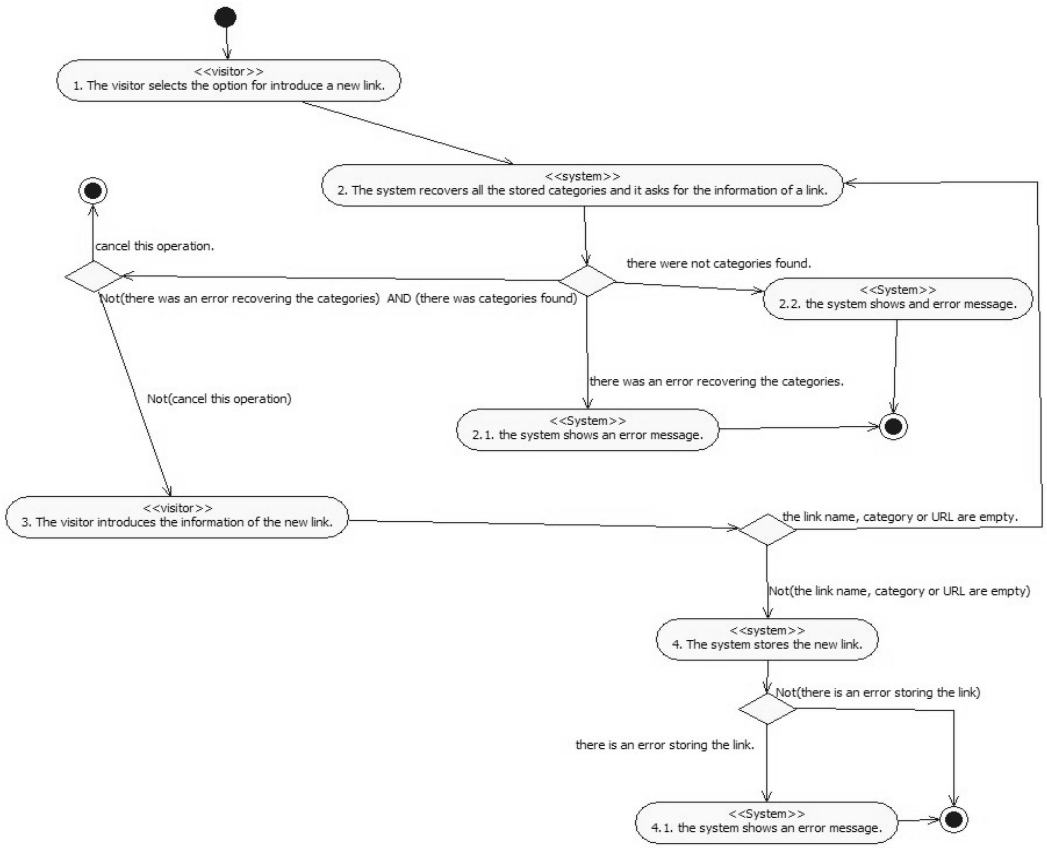

Figura 2. Diagrama de actividades obtenido a partir del caso de uso

```
<variables>
 <variable id="V_D01">
  <partition id="P_1">
    <condition>there was an error recovering the categories.</condition>
  </partition>
  <partition id="P_2">
    <condition>Not(there was an error recovering the categories)</condition>
  </partition>
 </variable>
 <variable id="V_D02">
  <partition id="P_1">
    <condition>there were not categories found.</condition>
  </partition>
  <partition id="P_2">
    <condition>Not(there were not categories found)</condition>
  </partition>
 </variable>
 <variable id="V_D03">
  <partition id="P_1">
    <condition>cancel this operation.</condition>
  </partition>
  <partition id="P_2">
    <condition>Not(cancel this operation)</condition>
  </partition>
 </variable>
 <variable id="V_D04">
  <partition id="P_1">
    <condition>the link name, category or URL are empty.</condition>
  </partition>
  <partition id="P_2">
    <condition>Not(the link name, category or URL are empty)</condition>
  </partition>
 </variable>
 <variable id="V_D05">
  <partition id="P_1">
    <condition>there is an error storing the link.</condition>
  </partition>
  <partition id="P_2">
    <condition>Not(there is an error storing the link)</condition>
  </partition>
 </variable>
</variables>
```

Figura 3. Variables obtenidas automáticamente a partir del diagrama de actividades.

A continuación, se construye de manera automática un diagrama de actividades para representar el comportamiento del caso de uso. En la figura 1 se muestra un caso de uso para realizar una búsqueda sobre el catálogo de enlaces definido como un documento XML que sigue el formato de la herramienta de soporte. El diagrama de actividades generado automáticamente se muestra en la figura 2 (como se puede ver, existe un bucle). Actualmente, nuestras herramientas de soporte sólo pueden trabajar con casos de uso definidos en lengua inglesa, por lo que todos los ejemplos mostrados a continuación están en dicho idioma.

Después, los escenarios del caso de uso se obtienen de manera automática recorriendo el diagrama de actividades desde su nodo de inicio hasta un nodo de fin. Distintos criterios de cobertura pueden aplicarse, como el criterio de recorrer todos los nodos o el criterio de recorrer todas las transiciones.

3. APLICACIÓN DEL MÉTODO CATEGORÍA-PARTICIÓN A LOS CASOS DE USOS

El proceso, basado en el método categoría-partición, presentado en este artículo consta de cuatro actividades: construcción del diagrama de actividades y generación de escenarios a partir de un caso de uso (como se ha visto en la sección anterior), identificación de las variables operaciones y sus particiones, identificación de restricciones de no valor e identificaciones de bucle y, por último, generación de combinaciones de valores. La primera actividad ya ha sido resumida en la sección anterior. Las tres actividades restantes se definen en las siguientes secciones.

3.1 Identificación de variables operacionales

El punto de partida para la identificación de variables y particiones son los diagramas de actividades de cada caso de uso. Las variables operacionales se identifican, o bien en aquellas acciones que representen una entrada o salida de información del sistema, o bien en los nodos de decisión del diagrama de actividades.

Los dominios de dichas variables operacionales se identifican a partir de los nodos de decisión del diagrama de actividades. Cada una de las transiciones salientes del nodo decisión será una partición distinta del dominio de cada variable operacional evaluada en dicha transición.

En [12] se describe el algoritmo para realizar estas operaciones.

El resultado de esta actividad es un listado de variables con sus correspondientes particiones. En la figura 3 se muestra el resultado mediante un documento XML para el diagrama de actividades de la figura 2 generado con nuestra herramienta de soporte (el DTD correspondiente puede descargarse de la página de la herramienta).

Aunque no se recoge en el archivo XML obtenido como resultado, en esta actividad también se calcula el máximo número de veces que una variable será evaluada. Toda variable es evaluada, al menos, una vez. Esto significa que en el conjunto de escenarios obtenidos de un diagrama de actividades (es decir, los caminos del diagrama de actividades) existirá al menos uno donde una variable operacional aparezca. Sin embargo, debido a la presencia de bucles, pueden existir escenarios dónde una misma variable sea evaluada más de una vez. En este caso, se almacena el máximo número de evaluaciones que una variable tiene en un escenario.

En el diagrama de actividades de ejemplo (figura 2), se aprecia claramente que todas las variables menos la última (es decir: V_D01, V_D02, V_D03 y V_D04) pueden ser evaluadas más de una vez debido al bucle existente. Como no se indica el número máximo de veces que dicho bucle puede aparecer, consideramos que el bucle puede aparecer ninguna o una vez, por ello, el máximo número de veces que las cuatro variables anteriores serán evaluadas es de 2. En cambio, el máximo

número de veces que la ultima variable será evaluada (variable V_D05) será 1.

Este valor será relevante en el cálculo de combinaciones, tal y como se verá en la sección 3.3.

3.2 Identificación de restricciones

Muy a menudo, los valores que toma una variable operacional establecen restricciones sobre los valores que pueden tomar otras variables operacionales. Por ejemplo, en el conjunto de variables y particiones de la figura 3, si la variable V_D01 toma el valor que indica que no se encuentra ninguna categoría (partición P_1), el caso de uso finaliza y ninguna de las variables restantes necesita tomar ningún valor.

Además, si la variable V_D04, toma el valor P_1 (algunos fragmentos de información se han dejado vacíos), es necesario evaluar de nuevo las variables V_D01, V_D02, V_D03 y V_D04 (las cuáles pueden tomar el mismo valor que en la vez anterior o un valor distinto).

Para poder tener estas características en cuenta y poder generar combinaciones de valores que se ajusten a ellas es necesario identificar dos tipos de restricciones: restricciones de no valor y restricciones de bucle. En los siguientes párrafos se abordan las restricciones de no valor y, al final de esta sección, las restricciones de bucle.

Para la identificación de restricciones de no valor será necesario, en primer lugar, construir un mapa de dependencias y, en segundo lugar, realizar la construcción de restricciones.

Un mapa de dependencias es una matriz cuadrada donde cada fila y columna corresponde a una variable operacional y donde cada celda sólo puede contener un 0 o un 1. La intersección de la variable operacional en una fila X con la variable operacional en una fila Y, indica si existe algún posible valor para X que permita a un camino alcanzar la variable Y (un 1), o si no existe ninguna manera de llegar a la variable Y (un 0). Por definición, no existe dependencia de una variable consigo misma. Un modelo para la definición de matrices de dependencias se muestra en la figura 4.

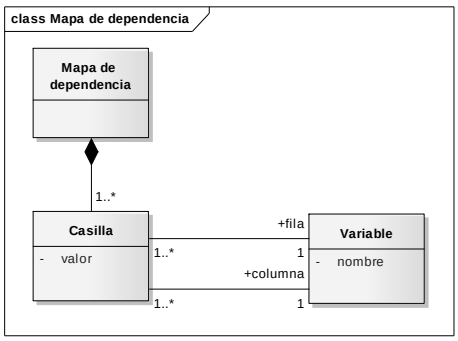

Figura 4. Modelo para matrices de dependencia.

El mapa de dependencias del diagrama de actividades de la figura 2 generado mediante la herramienta de soporte se muestra en la tabla 1.

Tabla 1. Mapa de dependencias del ejemplo

	V_D01	V_D02	V_D03	V_D04	V_D05
V_D01	0	1	1	1	1
V_D02	0	0	1	1	1
V_D03	0	0	0	1	1
V_D04	0	0	0	0	1
V_D05	0	0	0	0	0

A partir del conjunto de variables operacionales y particiones, del mapa de dependencias de las variables operacionales y del conjunto de caminos se puede obtener de manera automática un conjunto de restricciones de no valor. Para ello, se aplica la siguiente regla: si una variable operacional A permite alcanzar un conjunto de variables operacionales CV y encontramos un camino P dónde A toma un valor perteneciente a la partición PD de la variable A y dicho camino no contiene al subconjunto SCV de CV, entonces, para cualquier valor de la partición PD de la variable A, las variables del subconjunto SCV no toman ningún valor. A continuación se muestra un ejemplo de esta regla.

Aplicando esta regla al diagrama de actividades de la figura 2 y al conjunto de valores y particiones obtenidos en las secciones anteriores, se ve que, por ejemplo, la variable V_D01 permite alcanzar el conjunto de variables {V_D02, V_D03, V_D04, V_D05}. Sin embargo, existe un camino (ver diagrama de actividades) donde V_D01 toma un valor de la partición P_1 y dicho camino no contiene el subconjunto {V_D02, V_D03, V_D04, V_D05} (porque el camino termina). Por este motivo, se identifica la siguiente restricción: Si V_D01 toma valor en P_01 entonces V_D02, V_D03, V_D04 y V_D05 no toman ningún valor, y no será necesario seguir calculando combinaciones.

Durante el cálculo de escenarios de la actividad 1, también se identifican y se anotan los bucles. Durante el cálculo de los escenarios, se detecta un bucle cuando se recorre algún nodo del diagrama de actividades que ya se ha recorrido con anterioridad. Debido a las reglas para la definición de casos de uso y construcción de diagramas de actividades, sólo es posible recorrer de nuevo nodos del diagrama de actividades debido a alguna rama alternativa de un nodo decisión.

Con esta información es posible obtener de manera directa la información necesaria para las restricciones de bucle. La variable que da origen al bucle es la variable asociada al nodo decisión mencionado en el paso anterior. El bucle aparecerá cuando dicha variable tome sus valores en el dominio correspondiente a la rama de salida del nodo decisión. Las variables afectadas por el bucle se obtienen de todos los nodos decisión que se visitan por segunda vez.

3.3 Generación de combinaciones de valores

Con las variables y particiones obtenidas (5 variables con dos particiones cada una) se obtienen 32 combinaciones posibles. Debido al bucle existente, las cinco variables dan lugar a 9 instancias distintas (ya que 4 de ellas pueden evaluarse una segunda vez debido al bucle). Como una de las instancias sólo puede tomar un único valor (para evitar que el bucle se repita más de una vez), el número de combinaciones máximas aumenta

de 32 a 256. Sin embargo, gracias a las restricciones de no valor y a las restricciones de bucle, dicho número se va a ver reducido a 10, tal y como se explica en los siguientes párrafos.

Con toda la información obtenida, y con las características mencionadas en los párrafos anteriores, se va a generar un script en lenguaje BeanShell (figura 6) para calcular todas las posibles combinaciones del caso práctico que cumplan las restricciones generadas en la sección anterior (restricciones de no valor y de bucle). Se ha omitido la definición de variables y funciones auxiliares del script así como las restricciones de no valor asociadas a todas las instancias de las variables V02 y V03 (el código es muy similar al código para la variable V01). El código de este script funciona de manera similar al lenguaje TSL definido para el método CPM [1]. A continuación se describe cómo se ha generado el código para los elementos calculados en las secciones anteriores.

Para las restricciones de bucle se utilizarán los conceptos de instancia de variable y de contexto. El concepto de instancia de variables ya se ha definido con anterioridad y hace referencia a cada vez que una variable se evalúa (y, por tanto, puede tener un nuevo valor). El concepto de contexto hace referencia al bucle concreto cuyas instancias se están evaluando. Por ejemplo, durante un contexto concreto, deben tenerse en cuenta las restricciones que afecten sólo al conjunto de instancias de dicho contexto, sin modificar el valor de las instancias de otro contexto.

Al menos tiene que existir un contexto, ya que, todas las variables se evalúan al menos una vez. El número máximo de contextos que pueden existir es el mayor número de instancias de las variables operacionales, ya calculado en las secciones anteriores (en el ejemplo utilizado en este artículo, el número de contexto es dos).

Para cada contexto, se debe generar una versión de cada una de las restricciones de no valor que afecte únicamente a las variables de dicho contexto.

Las instancias que no participan en un contexto han sido marcadas con un valor especial. Gracias a este valor, es posible omitir dichas instancias de la secuencia y simplificar el resultado obtenido.

El resultado de la ejecución de este script se muestra en la captura de pantalla de la figura 5. Como se puede apreciar, se han eliminado el 96% de combinaciones.

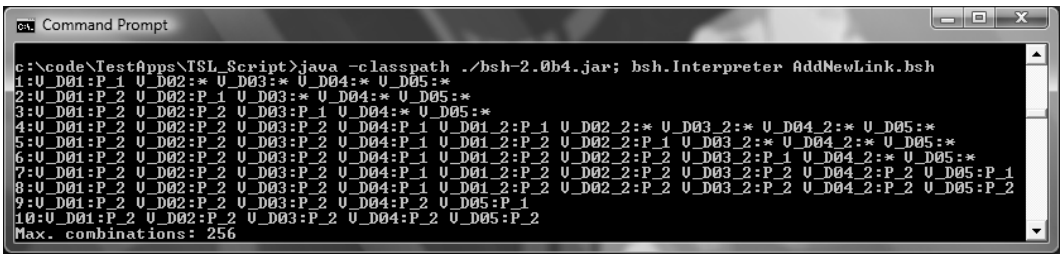

Figura 5. Cálculo de las combinaciones.

```
2  boolean P_V_D01_1 = false;
3  boolean P_V_D01_2 = false;
4
5  // Properties
6  if (V_D01_1 == "P_1")
7      P_V_D01_1 = true;
8
9  // Values
10 if (P_V_D01_1)
11     V_D02_1 = V_D03_1 = V_D04_1 = V_D05_1 = V_D01_2 = "*";
12
13 // Loops
14 if (V_D04_1 != "P_1") {
15     V_D01_2 = V_D02_2 = V_D03_2 = V_D04_2 = "--";
16     P_V_D01_2 = P_V_D02_2 = P_V_D03_2 = P_V_D04_2 = false;
17 }
18
19 // Properties
20 if (V_D01_2 == "P_1")
21     P_V_D01_2 = true;
22
23 // Values
24 if (P_V_D01_2)
25     V_D02_2 = V_D03_2 = V_D04_2 = V_D05_1 = "*";
```

Figura 6. Script para la generación de combinaciones de prueba para el caso práctico (fragmento).

Cada una de las combinaciones de la figura 5 indica el estado en que debe estar el sistema y la información que debe introducir el usuario para poder probar un escenario del caso de uso. Por ejemplo, para probar el escenario principal del caso de uso, que corresponde con la combinación 10, no debe aparecer ningún error recuperando las categorías (V_D01 toma valor en P_2), debe encontrarse alguna categoría (V_D02 toma valor en P_2), no se debe cancelar la operación (V_D03 toma valor en P_2), la información debe ser correcta (V_D04 toma valor en P_2) y no debe haber un error almacenando el nuevo enlace (V_D05 toma valor en P_2).

4. TRABAJOS RELACIONADOS

La mayoría de trabajos sobre prueba de casos de uso [13] [8] [7] no se centran en variables operacionales sino en el análisis de caminos de ejecución o de escenarios de caso de uso.

Además de los trabajos citados en la introducción, la propuesta PLUTO [2] aplica el método CPM para generar pruebas a partir de los puntos de variabilidad de casos de uso para familias de productos.

No se ha encontrado ningún trabajo que proponga el uso de las dos técnicas de manera complementaria. Aunque existen trabajos que proponen procesos para la generación automática de objetivos de prueba a partir de casos de uso definidos como prosa, por ejemplo [4], no hemos encontrado ningún trabajo que automatice la aplicación de CPM o del patrón de Binder.

5. CONCLUSIONES

En este trabajo se ha presentado un procesos sistemático y automático (mediante una herramienta de soporte de libre descarga y uso) que permite aplicar el método de categoría-partición y el patrón Extended Use Cases para obtener objetivos de prueba a partir de casos de uso escritos en lenguaje no formal.

Como se ha citado en la sección 4, la mayoría de propuestas para la obtención de objetivos de prueba a partir de casos de uso se centran en el análisis de escenarios de casos de uso. Sin embargo, en este trabajo hemos mostrado como el uso de la técnica CPM y de variables operacionales puede ser complementaria al análisis de escenarios, ya que permite definir futuros valores de prueba y resultados del sistema. Como hemos mencionado, no hemos tenido conocimiento de ningún trabajo hasta la fecha que proponga dicha relación.

Las dos herramientas de soporte utilizadas en este artículo son: ObjectGen para la generación de diagramas de actividades y objetivos de prueba, y ValueGen para la identificación de variables, identificación de restricciones y generación de combinaciones. Ambas pueden descargarse libremente en www.lsi.us.es/~javierj.

6. AGRADECIMIENTOS

Este trabajo ha sido desarrollado en el marco del Ministerio de Ciencia y Educación de España, bajo el programa de Investigación, Desarrollo e Innovación, proyecto OSimTec(TIN207- 67843-C06-03) y REPRIS (TIN2005-24792-E)

7. REFERENCIAS

[1] Balcer M. Hasling W. Ostrand T. 1989. Automatic Generation of Test Scripts from Formal Test Specifications. ACM SIGSOFT'89 - Third Symposium on Software Testing, Verification, and Analysis (TAVS-3), ACM Press, pp. 257-71.

[2] Bertolino, A., Gnesi, S. 2004. PLUTO: A Test Methodology for Product Families. Lecture Notes in Computer Science. Springer-Verlag Heidelberg. 3014 / 2004. pp 181-197.

[3] Binder, R.V. 1999. Testing Object-Oriented Systems. Addison Wesley.

[4] Boddu R., Guo L., Mukhopadhyay S. 2004. RETNA: From Requirements to Testing in Natural Way. 12th IEEE International Requirements Engineering RE'04.

[5] Burnstein, I. 2003. Practical software Testing. Springer Professional Computing. USA.

[6] Cockburn, A. 2000. Writing Effective Use Cases. Addison-Wesley 1st edition. USA.

[7] Roubtsov S. Heck P. 2006. Use Case-Based Acceptance Testing of a Large Industrial System: Approach and Experience Report. TAIC-PART. Windsor, UK.

[8] Denger, C. Medina M. 2003. Test Case Derived from Requirement Specifications. Fraunhofer IESE Report. Germany.

[9] Escalona M.J. 2004. Models and Techniques for the Specification and Analysis of Navigation in Software Systems. Ph. European Thesis. University of Seville. Seville, Spain.

[10] Escalona M.J. Martín-Pradas A., De Juan L.F, Villadiego D., Gutiérrez J.J. 2005 El Sistema de Información de Autoridades del Patrimonio Histórico Andaluz. V Jornadas de Bibliotecas Digitales. Granada, Spain.

[11] Escalona M.J. Gutiérrez J.J. Villadiego D. León A. Torres A.H. 2006. Practical Experiences in Web Engineering. 15th International Conference On Information Systems Development. Budapest, Hungary, 31 August – 2 September.

[12] Gutiérrez J.J. Escalona M.J. Mejías M. Torres J. Torres A. "Generación automática de objetivos de prueba a partir de casos de uso mediante partición de categorías y variables operacionales". XIV Jornadas sobre Ingeniería del Software y Bases de Datos JISBD. Zaragoza. Spain. 2007.

[13] Gutiérrez, J.J., Escalona M.J., Mejías M., Torres, J. 2006. Generation of test cases from functional requirements. A survey. 4º Workshop on System Testing and Validation. Potsdam. Germany.

[14] Gutiérrez J.J. Escalona M.J. Mejías M. Torres J. 2006. Modelos Y Algoritmos Para La Generación De Objetivos De Prueba. Jornadas sobre Ingeniería del Software y Bases de Datos JISBD. Sitges. Spain.

[15] Gutiérrez J.J. Escalona M.J. Mejías M. Reina A.M. 2006. Modelos de pruebas para pruebas del sistema. Taller de Desarrollo de Software Dirigido por Modelos. Jornadas sobre Ingeniería del Software y Bases de Datos JISBD. Sitges. Spain.

[16] Gutiérrez J.J. Escalona M.J. Mejías M. Torres A.H. Torres J. 2007. Generación e implementación de pruebas del sistema a partir de casos de uso. Revista Española de Innovación, Calidad e Ingeniería del Software. Volumen 3. Número 3. España. http://www.ati.es/article.php3?id_article=823x

[17] Ostrand T. J., Balcer M. J. 1988. Category-Partition Method. Communications of the ACM. 676-686.