41 JAIIO EST 2012 15° Concurso de Trabajos Estudiantiles

OVM2PN: Herramienta de análisis de configuraciones en Líneas de Productos de Software

Autor:

Nicolás Emilio Díaz Ferreyra

Institución:

UTN Regional Santa Fe

Carrera:

Ingeniería en Sistemas de Información

Cátedra:

Procesos de Desarrollo de Software

Docentes:

Dr. Horacio P. Leone

Dr. Silvio Gonnet

OVM2PN: Herramienta de análisis de configuraciones en Líneas de Productos de Software.

Abstract. Uno de los ejes centrales de una línea de producto de software (LPS) es la definición de su variabilidad. Esta prescribe las características a ser incluidas y las reglas de inclusión durante la derivación de productos individuales. Una forma de definir la variabilidad de una LPS es a través de un modelo de variabilidad ortogonal (OVM). Sin embargo, las familias de productos obtenidas pueden presentar ciertos problemas de inviabilidad, esto es, reglas de inclusión contradictorias que resultan en características imposibles de ser incorporadas en ningún producto. En este trabajo se propone una herramienta para representar, estudiar y detectar los problemas de inviabilidad en un OVM.

1 Introducción

Una línea de productos de software (LPS) es un conjunto de sistemas de software que comparten y gestionan un conjunto de características, que satisfacen las necesidades de algún mercado o misión en particular, y que son desarrollados a partir de un núcleo de software común [1]. Uno de sus principales beneficios proviene de la reducción de costos a partir del reuso de componentes y artefactos en varios productos.

La ingeniería de una LPS es un paradigma de desarrollo de aplicaciones de software que identifica dos procesos: la ingeniería de dominio y la ingeniería de aplicaciones [2]. En términos de la variabilidad de una LPS, el primer proceso la define y el segundo la explota seleccionando las características específicas durante la derivación de productos. En particular, la ingeniería de dominio es responsable de establecer el conjunto básico de elementos comunes y de definir la variabilidad de las aplicaciones resultantes de una LPS. El conjunto de elementos reusables incluye todos los tipos de artefactos asociados al software: requerimientos, diseño, realización, prueba, entre otros. Por otro lado, la ingeniería de aplicación se encarga de la derivación de aplicaciones a partir de los artefactos identificados en el proceso anterior. En ella se identifican dos actividades fundamentales: la selección de puntos de variación y la selección de variantes, ambas establecen el conjunto de características (configuración) del producto final.

Para dar soporte a tales actividades es necesario definir la variabilidad de la LPS. Existen dos enfoques para definirla: como parte integral de los artefactos de desarrollo o en un modelo separado. Pohl y colab. [3] plantean varias ventajas a favor de la segunda alternativa y proponen el modelo de variabilidad ortogonal (OVM). Los elementos básicos de OVM son los puntos de variación, las variantes y las dependencias de variabilidad y de restricciones.

Dado que la ingeniería de dominio no es solamente una etapa inicial sino que es un proceso constante, el OVM subyacente debe evolucionar para dar soporte a las nuevas

necesidades. El agregado, modificación o eliminación de elementos es el resultado de esta adaptación a las nuevas necesidades. Sin embargo, es necesario considerar que cualquier cambio puede derivar en un problema de inviabilidad. La inviabilidad puede darse en tres niveles: configuración, puntos de variación y variante. Una configuración inviable indica que ningún producto puede poseer el conjunto de características de tal configuración. Un punto de variación inviable nunca es considerado durante el proceso de derivación. Por último una variante inviable no puede ser incluida en ningún producto.

La presencia de alguno de estos niveles puede afectar no solo la derivación de futuros productos sino también invalidar los ya establecidos, como así también generar modelos confusos y contradictorios.

La simple inspección del modelo de variabilidad ortogonal para detectar la inviabilidad en cualquiera de sus tres niveles es una estrategia sumamente limitada. La complejidad de esta tarea es evidente, ya que basta con algunas decenas de puntos de variación y variantes para obtener OVM complejos y difíciles de examinar.

En este trabajo se propone el desarrollo de una herramienta de software, OVM2PN, que dé soporte a las actividades de Ingeniería de Dominio y de Aplicaciones en la validación de configuraciones de una LPS. Para detectar las inconsistencias vinculadas a la viabilidad de configuraciones de productos, puntos de variación y variantes se emplea el formalismo de redes de Petri [4] según la propuesta [5]. En [5] se define una topología de una red de Petri que permite la representación y estudio formal de los OVM. Las redes de Petri pueden ser utilizadas para estudiar la dinámica de eventos y sus precondiciones y post-condiciones. En términos de la variabilidad de una LPS, el OVM fija las condiciones o reglas de inclusión y la selección de puntos de variación y variantes corresponden a los eventos de una red de Petri. A su vez se observarán posteriormente, algunas propiedades de las redes de Petri que serán útiles para tratar los problemas de inviabilidad antes expuestos.

A continuación se realiza una síntesis de los elementos que constituyen un OVM. En la Sección 3 se introducen las topologías propuestas para construir una red de Petri a partir de los elementos básicos de OVM. En la Sección 4 se aborda la definición de una arquitectura de software y su implementación para el análisis de modelos OVM basados en redes de Petri. Luego, en la Sección 5 se incluye un caso de estudio. Finalmente, en la Sección 6, se presenta las conclusiones y trabajos futuros.

2 Modelo de Variabilidad Ortogonal

Un OVM especifica la variabilidad de una LPS. En la Fig. 1 se presenta el modelo de variabilidad ortogonal propuesto en [3] utilizando notación UML.

Los elementos básicos de un OVM son los *Puntos de variación* y las *Variantes* (Fig. 1). Un punto de variación es una representación de un ítem variable del mundo real o una propiedad variable de tal ítem. Una variante es una representación de una instancia particular del ítem. Como vemos en la Fig. 1, puntos de variación y variantes pueden asociarse entre sí de dos maneras, con *Dependencias de Variabilidad* y con *Dependencias de Restricción Punto de Variación-Variante*.

Una *Dependencia de Variabilidad* puede ser *Obligatoria* u *Opcional*. La dependencia obligatoria indica que una variante es siempre incluida en un producto si su punto de variación asociado es incluido. Una dependencia opcional establece que

la inclusión de un punto de variación puede (no necesariamente) incluir también la variante. Un conjunto de variantes opcionales (dos o más) junto a los valores máximo y mínimo definen una *Selección Alternativa*. Ambos valores, denominados cardinalidad máxima y mínima (*max* y *min* en Fig. 1), establecen el rango de cantidad de variantes que deberán ser incluidas junto al punto de variación.

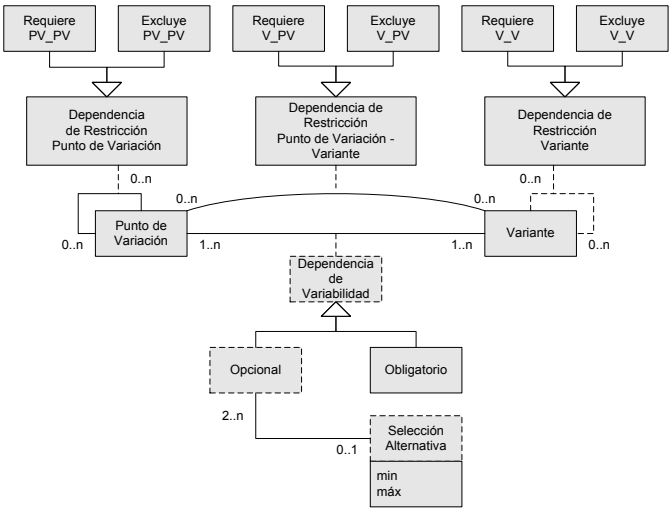


Fig. 1. *Meta modelo OVM propuesto en [3].*

Una *Dependencia de Restricción Punto de Variación - Variante* establece que la inclusión de una variante requiere o excluye un punto de variación. Este hecho es representado en la Fig. 1 por los conceptos *Requiere_V_PV* y *Excluye_V_PV*. En el primer caso la selección de la variante está condicionada a la selección del punto de variación. En el segundo caso, por el contrario, la relación establece que ambas son mutuamente excluyentes.

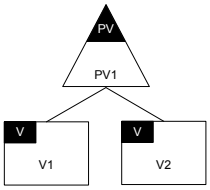
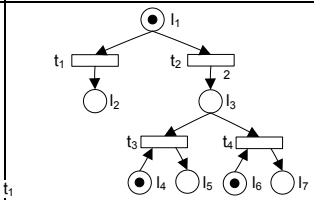
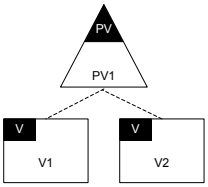
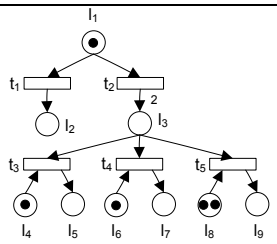
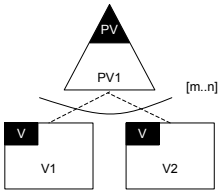
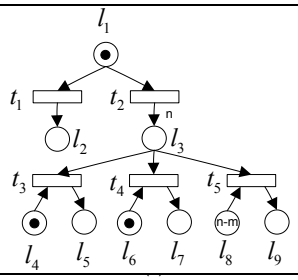
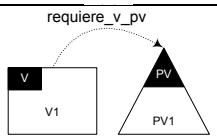
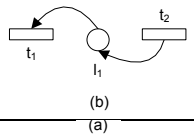
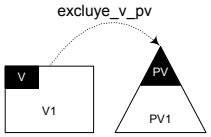
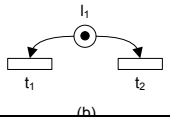
Finalmente, el OVM también define las dependencias de restricción únicamente entre puntos de variación (y entre variantes). Una relación *Requiere_PV_PV* (*Requiere_V_V*) establece que la inclusión de un *Punto de Variación* (*Variante*) está condicionada a la selección de otro *Punto de variación* (*Variante*). En el caso *Excluye_PV_PV* (*Excluye_V_V*) los *Puntos de variación* (*Variantes*) son excluyentes entre sí.

3 RP_{OVM}: Representación de OVM mediante Redes de Petri

A continuación se especifican las redes de Petri propuestas para representar los conceptos de OVM. Como se mencionó en la introducción, el presente trabajo implementa el modelo propuesto en [5]. En el mismo se definen las topologías para representar los elementos OVM a través de redes de Petri. Se abordan las topologías para el modelado de dependencias de variabilidad y de restricción. La selección de un punto de variación y la selección de una variante durante el proceso de derivación del producto son representados por los eventos (transiciones); los puntos de variación,

dependencias de variabilidad y dependencias de restricción son las pre-condiciones (lugares); y las variantes seleccionadas son las post-condiciones (lugares). La Tabla 1 resume la relación propuesta en [5] entre las dependencias que define el metamodelo OVM y los elementos de una Red de Petri. Aunque las siguientes redes de Petri corresponden a modelos OVM triviales, pueden ser combinados para dar soporte a modelos con mayor complejidad.

Tabla 1. *Correspondencias Dependencia OVM- red de Petri.*

DEPENDENCIA	OVM	PETRI
Obligatoria		
Opcional		
Alternativa		
Requiere		
Excluye		

El conjunto de redes de Petri para representar los elementos básicos del meta modelo OVM junto a las actividades del proceso de derivación se las denomina RP_{OVM}. Una de las fortalezas de las redes de Petri es el soporte que brindan para el

análisis de propiedades asociadas a sistemas dinámicos. Algunas propiedades están estrechamente vinculadas a los niveles de inviabilidad presentados en la Sección 1. Acotación y alcanzabilidad permitirán tratar con el problema de las configuraciones inviables y L1-viva (potencialmente disparable) con los puntos de variación y variantes inviables.

Las propiedades Acotación y Alcanzabilidad son útiles para tratar con el nivel 1 de inviabilidad (configuración). Siendo que una RP_{OVM} es acotada entonces su grafo de alcanzabilidad es finito. Los nodos del grafo representan todas las posibles configuraciones de la LPS, los de mayor interés son los nodos terminales. Estos nodos representan configuraciones que no tienen ninguna decisión pendiente sobre la inclusión o no de variantes o puntos de variación. Luego de un cambio en OVM (y su RP_{OVM}) el grafo de alcanzabilidad resultante puede compararse con el grafo de la red anterior para detectar configuraciones inviables en el nuevo modelo.

La propiedad L1-viva es útil para tratar los niveles 2 y 3 de inviabilidad. Si toda transición es potencialmente disparable, cualquier punto de variación y variante serán incluidos al menos en una configuración y en consecuencia el modelo OVM no tendrá ningún punto de variación ni variante inviable.

4 Diseño de OVM2PN

La herramienta VARMOD brinda soporte en la documentación, definición de la variabilidad y permite definir modelos OVM, es decir los puntos de variación, sus variantes y las relaciones entre los elementos variables (puntos de variación y variantes). A partir de esta aplicación se propone la definición e implementación de una herramienta que brinde soporte en el análisis y detección de inviabilidades en la LPS. Para esto la herramienta propuesta, OVM2PN (Fig 2), a partir de un modelo de variación ortogonal generado mediante la aplicación VARMOD (archivo .vedit en Fig. 2) genera la red de Petri RP_{OVM} (archivo .xml en Fig. 2) según la topología descripta en la sección anterior. Dicha red es luego analizada para detectar los problemas de inviabilidad empleando una aplicación de código abierto como PIPEv2 (Fig. 2). Se eligió en este caso "PIPEv2" (http://pipe2.sourceforge.net/) como aplicación que de soporte al análisis ya que permite construir entre otras cosas el grafo de alcanzabilidad a partir de una red dada.

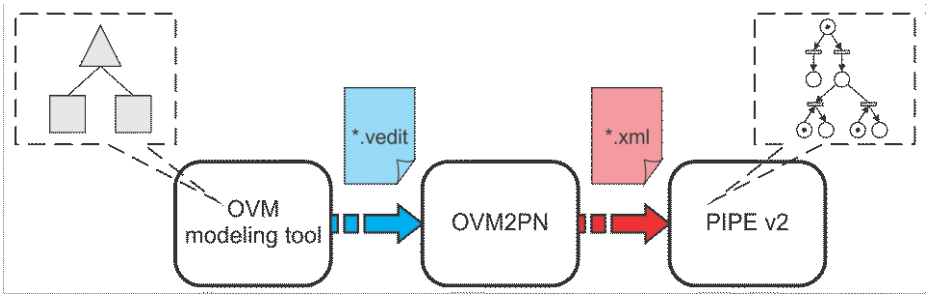


Fig. 2. *Propósito de OVM2PN*

Antes de comenzar el desarrollo de la aplicación OVM2PN, debieron hacerse algunas consideraciones. En una primera instancia, surgió la necesidad de construir un

"Parser" (Fig. 3) que permitiera la carga de los modelos OVM en formato XML (Fig. 4). El parser se encargará de leer secuencialmente el fichero de entrada que le proporciona el usuario y se valdrá de las clases correspondientes a un modelo de variación ortogonal para generar el modelo de objetos pertinente. Es en este punto donde además aparece la necesidad de un módulo que, a partir de los objetos correspondientes al modelo OVM generados por el parser, los utilice para generar la Red de Petri RP_{OVM} a través de la topología presentada en la sección anterior. Este módulo se valdrá de las clases vinculadas a una red de Petri para instanciar los objetos correspondientes.

Finalmente se debió evaluar de qué forma la red generada sería transferida al entorno de "PIPE 2" para su posterior análisis. Para dar solución a esta cuestión se pensó en el desarrollo de un nuevo módulo que a partir de los objetos instanciados de la Red de Petri, obtuviera un archivo XML que luego le permitiera al usuario cargarlo en la aplicación antes mencionada.

A continuación se detalla la arquitectura candidata sobre la cual se implementará la herramienta que dará soporte al análisis de los modelos OVM (Fig. 3).

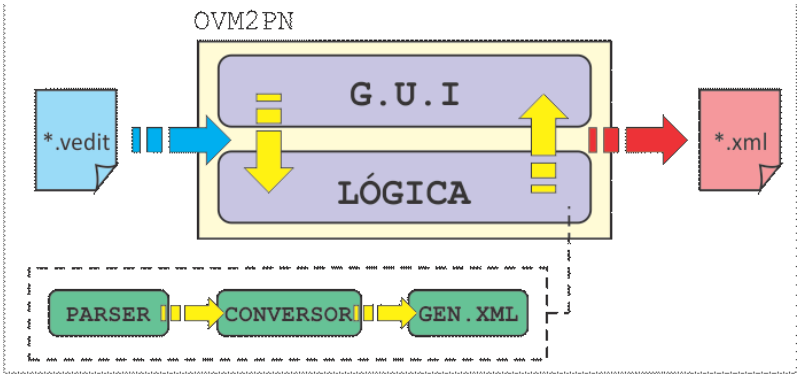


Fig. 3. *Arquitectura de OVM2PN*

La arquitectura resultante consta de dos capas que brindan servicios entre sí a través de un conjunto de interfaces. La primera corresponde a la GUI, o capa de presentación, la cual comprende a la interfaz gráfica de usuario y tiene por objetivo facilitar la interacción con el sistema. Ésta permite la carga de un archivo XML "*.vedit" en el cual se encuentra el modelo VMO a analizar. La capa Lógica es en donde se almacenan todos los componentes de procesamiento que dan lugar al archivo de salida en formato xml, para que pueda ser luego abierto y analizado por la plataforma PIPEv2 de acceso libre.

Esta capa fue definida mediante tres componentes implementando una arquitectura de tipo "pipes & filters". La GUI será la encargada de transferirle el modelo OVM en forma de un archivo XML al Parser, éste generará los objetos correspondientes al modelo de variación ortogonal y una vez completada la lectura pasará a transferírselos al Conversor. Éste conversor obtiene la red de Petri a partir de los objetos instancias de OVM según la topología presentada en la sección anterior. El módulo Gen. XML (Generador XML) es quien almacena en un nuevo archivo XML la red generada por

el Conversor; utiliza los objetos esta vez instanciados en dicho módulo y aplica un algoritmo de generación de código XML.

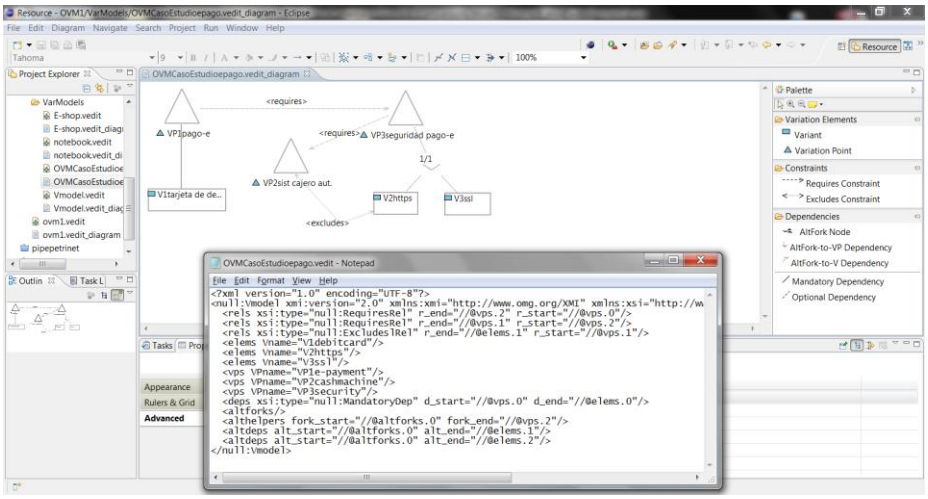


Fig. 4. *Modelo OVM en VARMOD y archivo .vedit*

Una vez procesado por completo el archivo "*.vedit", la capa Lógica presenta al usuario la ruta en donde se almacenó el fichero XML que es generado por el módulo Gen. XML y a su vez le da la opción de poder visualizarlo en el browser configurado por defecto en el ordenador.

5 Caso de estudio:

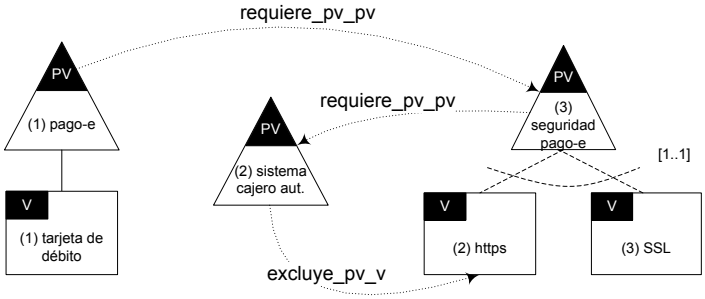
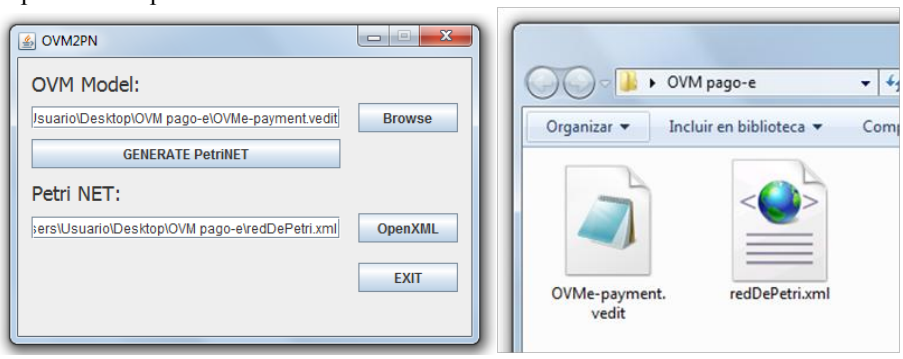


Fig. 5. *OVM de pago-e.*

En esta sección se aborda el caso de estudio introducido en la Fig. 5, donde se plantea un modelo OVM para la definición de la variabilidad de pago electrónico (pago-e) en una LPS. Se presentan los puntos de variación pago-e (*PV1*), sistema de cajero automático (*PV2*) y seguridad pago-e (*PV3*). La variante tarjeta de débito (*V1*) es obligatoria de *PV1*. Las variantes https (*V2*) y SSL (*V3*) son opcionales de *PV3* y componen una selección alternativa con cardinalidad [1..1]. Las variantes asociadas a *PV2* no afectan el resto de las dependencias y no fueron incorporadas en el MVO

expuesto. Por último las dependencias de restricción muestran la exclusión mutua entre *PV2* y *V2* (*Excluye_PV_V*) y las condiciones de inclusión de *PV1* y *PV3* (*Requiere_PV_PV*).

Se procederá entonces a la carga del archivo "OVMe-payment.vedit" correspondiente para luego generar la red de Petri en formato XML. A continuación (Fig. 6 (a)) se presenta la ventana principal de la aplicación. Como salida la aplicación, genera el archivo "redDePetri.xml" (Fig. 6 (b)) el cual es luego abierto empleando la aplicación "PIPEv2".



(a) (b)

Fig. 6. (a) *Pantalla principal y* **(b)** *archivo generado por OVM2PN.*

La Fig. 7 ilustra la red de petri generada por OVM2PN y abierta desde PIPEv2.

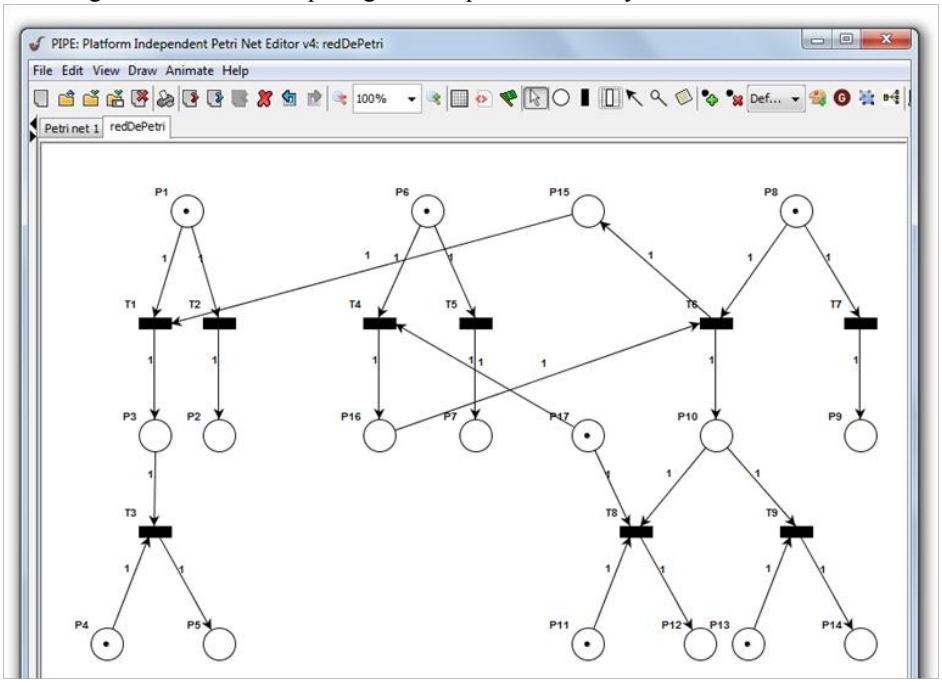


Fig. 7. *Red de Petri generada por OVM2PN abierta en el entorno PIPEv2.*

Como se mencionó en la Sección 3, al tratarse de una red acotada, podemos aplicar la propiedad de alcanzabilidad para tratar con problemas de inviabilidad vinculadas a la configuración. La herramienta empleada nos permite obtener el grafo de alcanzabilidad correspondiente a la red provista (Fig. 8). De éste nos interesan especialmente los nodos terminales, es decir, aquellos que corresponden a las posibles configuraciones de producto a partir de la definición de variabilidad del OVM. En cada uno de ellos se encuentra una configuración válida para el modelo propuesto, ergo, nos sirven para detectar inviabilidades en determinadas configuraciones. Dada una propuesta de personalización de un producto, bastará solo con observar los nodos hojas del grafo de alcanzabilidad correspondiente a la red de Petri generada para alertar sobre inconsistencias.

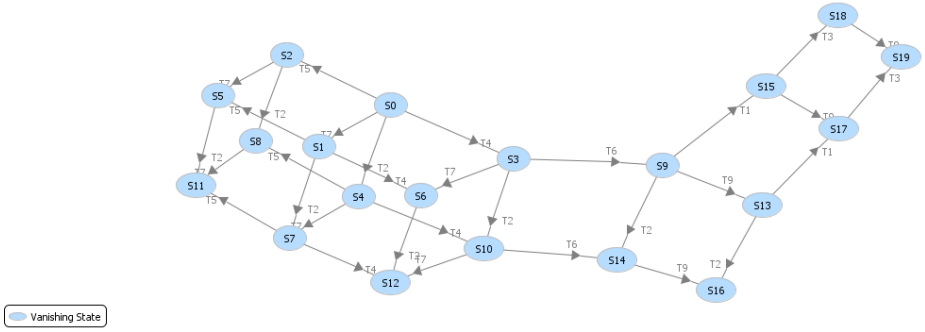


Fig. 8. *Grafo de alcanzabilidad.*

Además, el grafo incluye las transiciones que son necesarias para llegar desde el marcado inicial hacia el marcado final. Ciertas transiciones corresponden a la selección de un punto de variación (por ejemplo *T1* en Fig. 8, representa la selección del punto de variación pago-e (*PV1*) en Fig. 5), y otras corresponden a la selección de una variante (por ejemplo *T3* en Fig. 8, representa la selección de la variante tarjeta de débito (*V1*) en Fig. 5). Teniendo en cuenta que para que una configuración sea válida es condición necesaria que no queden transiciones sin ser ejecutadas, si por algún motivo quedara pendiente la selección de una variante, el mismo grafo nos alertará que el estado actual de la red no corresponde a un nodo hoja (por ejemplo estado S17 en Fig. 8), y no habilitará al usuario a almacenar la configuración.

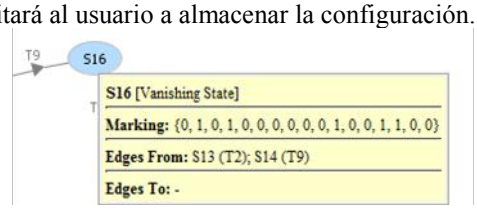


Fig. 9. *Nodo S16.*

Si observamos por ejemplo el nodo terminal *S16* (Fig. 8, Fig. 9), vemos cuál es el marcado correspondiente que genera dicha configuración (Fig. 9) y cuáles son las transiciones que se deben ejecutar para llegar hacia ella (desde S0 a S16 en Fig. 8, un camino posible es *T2*, *T4*, *T6*, y *T9*). Tomando ventaja de la trazabilidad existente, podemos concluir que *S16* corresponde a la configuración en donde el PV "pago-e"

no es seleccionado (*T2*), luego es seleccionado el PV "sistema cajero aut." (*T4*) como así también el PV "seguridad pago-e" (*T6*) y finalmente es seleccionada la variante "SSL".

6 Conclusiones y trabajos futuros

Es de vital importancia tanto para la ingeniería de procesos como para la ingeniería de productos el concepto de inviabilidad en una LPS. Esta puede manifestarse en tres diferentes niveles: configuración, puntos de variación y variantes. Una configuración inviable indica que ningún producto puede exhibir el conjunto de características incluidas en una determinada configuración. Un punto de variación inviable nunca es considerado durante el proceso de derivación. Por último una variante inviable no puede ser incluida en ningún producto.

OVM2PN genera una PN cuya propiedad de alcanzabilidad permite abordar problemas de inviabilidad vinculados a la configuración. Esto es posible gracias a que la propiedad L1-viva asociada al grafo de alcanzabilidad correspondiente a la PN subyacente al modelo OVM, permite tratar con los niveles 2 y 3 de inviabilidad. Garantizando la viabilidad en los niveles 2 y 3, estamos asegurando la inexistencia de problemas a nivel de configuración. Esto se debe a que si toda transición es potencialmente disparable, cualquier punto de variación y variante serán incluidos al menos en una configuración y en consecuencia el modelo OVM no tendrá ningún punto de variación ni variante inviable. Con lo mencionado anteriormente podemos afirmar que la herramienta presentada es de alta efectividad a la hora de obtener configuraciones válidas de productos pertenecientes a una misma LPS, tal como se ilustró en el caso de estudio (Fig. 8).

La herramienta presentada en este trabajo sirve de nexo entre un framework de diseño de modelos OVM y una herramienta de tipo *open source* con la cual se analizan las distintas propiedades de una red de Petri. Surge como nuevo desafío, la necesidad de desarrollar una herramienta que integre tanto modelado como análisis. De esta manera se buscará brindar un mayor soporte en los procesos de ingeniería de dominio e ingeniería de producto y alcanzar mayores niveles de reusabilidad de componentes y artefactos. Logrando esto, estaremos acercándonos aun más a las metas de reducción de costos que promete el empleo de LPS.

Referencias

1. Software Engineering Institute, Carnegie Mellon: A Framework for Product Line Practice, Version 5.0. http://www.sei.cmu.edu/productlines/frame_report/index.html (2009)

2. F. van der Linden, K. Schmid, E. Rommes: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer Heidelberg (2007)

3. K. Pohl, G. Böckle, F. van der Linden: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer Heidelberg (2005)

4. T. Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, Nro. 4 (1989)

5. O.C. Martinez, S. Gonnet, H. Leone: Análisis de Modelos de Variabilidad Ortogonal empleando Redes de Petri. Proceedings 39JAIIO - ASSE 2010, 562 – 567 (2010).