Representación de la Variabilidad en Líneas de Productos de Software empleando Redes de Petri

Cristian Martinez, Nicolás Díaz, Silvio Gonnet, Horacio Leone

INGAR (UTN - Conicet)
Facultad Regional Santa Fe, Universidad Tecnológica Nacional
{ocmartinez, sgonnet, hleone}@santafe-conicet.gov.ar,
nicoediaz@gmail.com

Abstract. Se define variabilidad como la posibilidad que posee un sistema de ser extendido, cambiado, localizado, o configurado para su uso en un contexto particular. Su especificación en una línea de productos de software (LPS) es una actividad central, donde se establecen las familias de productos con sus variantes, y dependencias. Una forma de definir la variabilidad de una LPS es a través de un modelo de características ("features", FM). 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. Tales inconvenientes pueden provenir tanto de errores durante la elaboración inicial del FM, como por cambios introducidos para alcanzar nuevas necesidades. En este trabajo se propone una herramienta basada en redes de Petri para representar y analizar los FMs y detectar los problemas antes mencionados.

Keywords: Features, variabilidad, Redes de Petri

1 Introducción

Una línea de productos de software (LPS) es un conjunto de sistemas de software que comparten y gestionan un conjunto de características 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 posee dos procesos fundamentales: la ingeniería de dominio y la ingeniería de aplicaciones [2]. En términos de la variabilidad de una línea de productos, 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. 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 este proceso se 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 desarro-

llo o en un modelo separado. Pohl y colab. [3] plantean varias ventajas a favor de la segunda alternativa. Dentro del grupo de las segundas alternativas, el método más empleado es el análisis de dominio orientado a características (FODA) [4]. FODA se centra en la representación de las características del dominio (features), y de relaciones estructurales entre ellas. FODA propone un lenguaje de modelado de features (FM), el cual es luego extendido por otros autores [5].

Dado que la ingeniería de dominio no es solamente una etapa inicial sino que es un proceso constante, el FM 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, se debe considerar que cualquier cambio puede resultar en un problema de inviabilidad. Una configuración inviable indica que ningún producto puede poseer el conjunto de características de tal configuración. Esto puede afectar la derivación de futuros productos, invalidar los ya establecidos y generar modelos confusos y contradictorios.

La simple inspección del FM [4, 5] para detectar la inviabilidad es una estrategia sumamente limitada. La complejidad de esta tarea es evidente, ya que basta con algunas decenas de features para obtener FMs complejos y difíciles de examinar. Benavides y colab. [6] resaltan la tendencia ascendente en los últimos años en la cantidad de features disponibles en las LPSs. De 15 features empleadas en el 2004, en el 2010 fue común encontrar modelos con 300 features. Este escenario requiere de herramientas de análisis que brinden soporte en el análisis de las LPSs.

Kang y colab. [4] y Benavides [6] han identificado un conjunto completo de operaciones que deben ser realizadas durante el análisis de FMs. Diversas propuestas han abordado el estudio parcial de estas operaciones [7, 8, 9] mediante métodos formales. Estas contribuciones emplean distintos formalismos, como lógica proposicional y programación por restricciones, y permiten la verificación de la LPS. Sin embargo, ninguna de estas contribuciones aporta información sobre el proceso de derivación del producto: selección de features para establecer una configuración final. En consecuencia, el principal objetivo de este trabajo es la introducción de un enfoque basado en redes de Petri (RP) [10] para la representación y evaluación de FM como un sistema dinámico. En esta propuesta consideramos tres funciones relevantes vinculadas al problema de inviabilidad antes expuesto en una LPS: (1) detección de nodos muertos, (2) obtención de un producto, y (3) obtención de todos los productos.

- Detección de nodos muertos: un nodo muerto representa una feature que nunca aparece en una configuración de la LPS. La inconsistencia introducida al modelo por estos nodos inviables introducen complejidad a la LPS y dificultan su mantenimiento.
- Obtención de un producto: esta función permite obtener un producto cuya configuración es factible.
- Obtención de todos los productos: esta función permite conocer todos los productos posibles. Es de vital importancia en la evolución de una LPS, ya que todos los productos previamente generados deben ser válidos luego de los cambios introducidos.

La primera función aborda un problema de consistencia de la LPS, mientras que las otras dos funciones son problemas de satisfacibilidad. A partir de lo enunciado, en esta contribución representamos los FMs empleando RP. A partir de la red obtenida es posible analizar sus propiedades brindando soporte a las tres funciones recientemente introducidas. A diferencia de las propuestas [7, 8, 9] y las analizadas en [6], nuestro trabajo se enfoca en las actividades y su precedencia. Para cada configuración válida es posible encontrar la secuencia de decisiones relacionadas con la inclusión y exclusión de features. Además, el modelo propuesto posee la capacidad de simulación y análisis que provén las RPs.

A continuación se realiza una síntesis de los elementos que constituyen un FM y el formalismo RP. Luego, en la Sección III, se describen las topologías propuestas para construir una RP a partir de los elementos básicos de un FM. En la Sección IV se presenta la definición formal de las topologías antes mencionadas. La Sección V desarrolla un caso de estudio. Finalmente, la Sección VI discute conclusiones y líneas de investigación a seguir.

2 Modelo de Características y Redes de Petri

2.1 Modelo de Características

Un FM representa la información de todos los productos posibles de una LPS modelando las características del dominio (*features*) y las relaciones estructurales entre ellas [4]. En un FM, las *features* están conectadas jerárquicamente en una estructura similar a un árbol, donde una *feature padre* (o compuesta) está conectada con sus *features hijos* (o sub-features) mediante las siguientes relaciones de *dependencia de variabilidad*:

- *Obligatoria*. Relación entre una feature padre y una feature hijo que indica que la feature hijo aparece en todos los productos donde es incluido el padre.
- *Opcional*. Relación entre una feature padre y una feature hijo que indica que la feature hijo puede ser opcionalmente incluida en los productos donde se incluye a la feature padre.
- *Alternativa* o *cardinalidad de grupo*. Relación entre una feature padre y un conjunto de features hijos. La cardinalidad del grupo se representa mediante un intervalo [*min*..*max*] que limita el número de features hijos que pueden incluirse en un producto cuando se incluye a la feature padre, siendo los valores *min* y *max* la cantidad mínima y máxima de features hijos a incluir. Las relaciones *OR* y *XOR* son casos especiales de cardinalidad [*1*..*max*] y [*1*..*1*], respectivamente.

En un FM, es posible definir ciertas *restricciones* entre las *features* que atraviesan el árbol:

- *Requiere*. Relación que condiciona la inclusión de una feature a la selección de otra feature.
- *Excluye*. Relación que indica que dos features son excluyentes entre sí.

2.2 Formalismo Red de Petri

Las RPs son una herramienta de modelado matemático que permite estudiar la dinámica de eventos junto a sus precondiciones y post-condiciones [10]. Una RP es un grafo dirigido con dos tipos de nodos: transiciones y lugares. Una transición (evento) está vinculado con un conjunto de lugares de entradas y salidas mediante arcos, representando las precondiciones y post-condiciones del evento, respectivamente. En la Tabla 1 se presenta una definición formal de una RP.

Table 1. Definición formal de una red de Petri.

Una red de Petri es definido como una 5-tupla (L, T, A, P, M_0), donde:
$L = \{p_1, p_2, \ldots, p_m\}$ es un conjunto finito de lugares,
$T = \{t_1, t_2, \ldots, t_n\}$ es un conjunto finito de transiciones,
$A \subseteq (L \times T) \cup (T \times L)$ es un conjunto de arcos,
$P: A \rightarrow \mathbb{N}$ es una función de peso,
$M_0: L \rightarrow \mathbb{N}_0$ es el marcado inicial,
$L \cap T = \varnothing$ y $L \cup T \neq \varnothing$.

3 Representación de la Variabilidad empleando Redes de Petri

En esta sección se definen las topologías propuestas para representar los elementos de FM a través de RPs. Los elementos de un FM junto a las actividades fundamentales de la ingeniería de aplicación pueden ser considerados según el enfoque evento/condición del modelado de RPs. Las *selecciones de features* realizadas durante la derivación de un producto corresponden a *eventos* del proceso de la ingeniería de aplicación y se representan mediante transiciones de una RP. Las *features*, las *dependencias de variabilidad* y las *dependencias de restricciones* son las precondiciones que rigen tales eventos y se representan mediante *lugares* de una RP. Finalmente las *features* incluidas son el resultado de la ocurrencia de los eventos, esto es, las post-condiciones, también representadas por *lugares*. En la Tabla 2 se resumen las relaciones propuestas entre los elementos de FM, las actividades del proceso de ingeniería de aplicación, y los elementos de una RP. Entre paréntesis se incluyen los nombres genéricos que se le asignan a cada elemento.

A la RP resultante la denominamos RP_{FM}. A partir de la red de Petri RP_{FM} es posible estudiar su dinámica y mostrar la relación entre sus marcados (M) y las posibles configuraciones del FM subyacente. Los marcados de interés son aquellos que no tienen ninguna transición (t) habilitada, es decir, que todas las decisiones de selección de features fueron resueltas. De tales marcados, los lugares que corresponden a features reflejan una configuración probable de la LPS.

En primer lugar, introducimos brevemente la notación empleada, luego, en las siguientes secciones se presenta las topologías propuestas para representar los elementos de FM a través de RPs. Si consideramos a l_1 como un lugar que representa a una feature f_1, M el marcado de la red en un instante dado, y $M(l_1)$ la cantidad de marcas en l_1 en el marcado M, $M(l_1)$=1 representa la selección de la feature f_1, y $M(l_1)$=0 indi-

ca la no inclusión de la feature f_1. La secuencia de disparo σ es la cadena de eventos (selecciones) para alcanzar el marcado M. La siguiente notación se emplea para denotar ciertos pre- y post-conjuntos:

- $\bullet t = \{l \mid (l, t) \in A\}$, conjunto de lugares de entrada de t,
- $t\bullet = \{l \mid (t, l) \in A\}$, conjunto de lugares de salida de t,
- $\bullet l = \{t \mid (t, l) \in A\}$, conjunto de transiciones de entrada de t,
- $l\bullet = \{t \mid (l, t) \in A\}$, conjunto de transiciones de salida de t.

Table 2. Relaciones entre los elementos de FM y Red de Petri.

Elementos de FM	**Elementos de Red de Petri**
Feature (f_i)	1 Lugar (*PfiSeleccionada*) y 1 Arco
Selección de una Feature (f_i)	1 Transición (*TSeleccionfi*)
Feature raíz (f_0)	2 Lugares (*P0, Pf0NoSeleccionada*), 3 Arcos y 1 Transición (*TNoSeleccionf0*)
Obligatoria (f_1 —● f_2)	1 Lugar (*Pf1obligatoriaf2*) y 2 Arcos
Opcional (f_1 —o f_2)	2 Lugares (*Pf1opcionalf2*, *Pf2NoSeleccionada*), 4 Arcos y 1 Transición (*TNoSeleccionf2*)
Alternativa (f_p [min,max] — f_1 .. f_n)	3 Lugares (*Pfpalternativaf1fn*, *PNoSeleccionfpAlternativaf1fn*, *Pf1fnNoSeleccionada*), 4 Arcos y 1 Transición (*TNoSeleccionfpAlternativaf1fn*) Por cada alternativa: 1 lugar (*PCardMaxfi*) y 1 arco
Requiere (f_1 requiere f_2)	1 Lugar (*Pf1requieref2*) y 2 Arcos
Excluye (f_1 excluye f_2)	1 Lugar (*Pf1excluyef2*) y 2 Arcos

3.1 Característica Raíz del FM

La feature raíz, f_0, del FM se representa mediante la RP presentada en la Fig. 1. La RP incluye al lugar *P0* el cual está inicialmente marcado, es decir, $M_0(P0)$= 1. La marca en *P0* (Fig. 1) habilita las transiciones *TNoSeleccionf0* y *TSeleccionf0*, transiciones que modelan los eventos de no selección y selección de la feature f_0, respectivamente. A partir de este modelo existen dos posibles configuraciones $\{\varnothing, \{f_0\}\}$. La primera ($\varnothing$) no considera la feature raíz f_0 y es representado por $M_i(Pf0NoSeleccionada)$= 1, i >0. En cambio, la segunda configuración ($\{f_0\}$) considera la selección de f_0, resultando $M_i(Pf0Seleccionada)$= 1, para i >0.

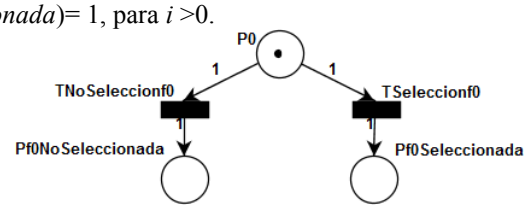

Fig. 1. Red de Petri propuesta para la feature raíz, f_0, de un FM.

3.2 Dependencia de variabilidad obligatoria

Tal como se indicó en la Sección 2.1, una dependencia de variabilidad obligatoria (Fig. 2 (a)) entre una feature f_1 y una feature f_2 establece que la consideración de la

feature f_1 es condición suficiente para la inclusión de la feature f_2. La Fig. 2 (a) muestra un FM con la feature f_1 vinculado a la feature f_2 por este tipo de dependencia de variabilidad. A partir de este modelo existen dos posibles configuraciones {∅, {f_1, f_2}}. La primera (∅) no considera la feature f_1, mientras que la segunda ({f_1, f_2}) al considerar f_1 incluye obligatoriamente a *f2*.

La RP propuesta, presentada en Fig. 2 (b), representa la dependencia de variabilidad obligatoria mediante el lugar *Pf1obligatoriaf2* y los arcos que vinculan las transiciones *TSeleccionf1* y *TSeleccionf2*. Estas transiciones se corresponden con las selecciones de las features f_1 y f_2, respectivamente. El lugar *Pf2Seleccionada* representa el hecho que la feature f_2 es parte de la configuración del producto final cuando M_i(*Pf1Seleccionada*)= 1, para algún i >0.

Fig. 2. Dependencia de variabilidad obligatoria. (a) modelo FM incluyendo una feature obligatoria (f_2). (b) red de Petri propuesta.

3.3 Dependencia de variabilidad opcional

En una relación opcional (Fig. 3 (a)) entre una feature f_1 y una feature f_2, la selección de la feature f_1 no implica necesariamente la inclusión de la feature f_2. A partir de este modelo existen tres posibles configuraciones {∅, {f_1}, {f_1,f_2}}.

La RP propuesta (Fig. 3 (b)) representa la dependencia de variabilidad opcional mediante el lugar *Pf1opcionalf2* y los arcos que vinculan las transiciones *TSeleccionf1*, *TSeleccionf2*, y *TNoSeleccionf2*. Estas transiciones se corresponden con las selecciones de las features f_1 y f_2, y la no selección de f_2, respectivamente. El lugar *Pf2Seleccionada* representa que la feature f_2 es parte de la configuración del producto final cuando M_i(*Pf2Seleccionada*)= 1, para algún i >0. Por lo contrario, una marca en *Pf2NoSeleccionada* representa la decisión de no incluir la feature opcional *f2*.

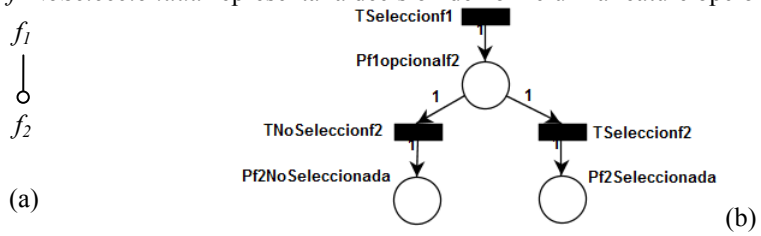

Fig. 3. Dependencia de variabilidad opcional. (a) modelo FM incluyendo una feature opcional (f_2). (b) red de Petri propuesta.

3.4 Dependencia de variabilidad alternativa o de cardinalidad de grupo

Una selección alternativa establece que la selección de la feature padre f_p (Fig. 4 (a)) es condición suficiente para la inclusión de al menos *min* y a lo sumo *max* features

alternativas f_i, i=*1..n* (Fig. 4 (a)). Si consideramos n=2, *min*=0, y *max*=1 el conjunto de configuraciones posibles es $\{\varnothing, \{f_p\}, \{f_p, f_1\}, \{f_p, f_2\}\}$.

La red de Petri propuesta, presentada en Fig. 4 (b), representa la dependencia de variabilidad alternativa mediante el lugar *Pfpalternativaf1fn* y los arcos que lo vinculan con las transiciones *TSeleccionfp*, *TSeleccionf1*, *TSeleccionf2*, …, *TSeleccionfn*, y *TNoSeleccionfpAlternativa1fn*. El arco que vincula *TSeleccionfp* con *Pfpalternativaf1fn* posee un peso igual a *max*. Ese arco establece la cantidad máxima de selecciones de features alternativas f_i, es decir, disparos de las transiciones *TSeleccionf1*, *TSeleccionf2*, …, *TSeleccionfn*.

El lugar *PfjSeleccionada* (j= *1..n*) representa que la feature f_j es parte de la configuración del producto final cuando M_i(*PfjSeleccionada*)= 1, para algún i >0. La cantidad de veces que puede ser seleccionada cada feature alternativa está limitada por los lugares *PCardMaxf1*, *PCardMaxf2*, …, *PCardMaxfn*, los cuales están marcados en M_0. Por lo contrario, cada marca en *Pf1fnNoSeleccionada* representa la decisión de no incluir una feature alternativa y el lugar *PNoSeleccionfpAlternativaf1fn*, con un marcado inicial igual a *max* – *min*, condiciona la cantidad máxima de no selecciones que se pueden realizar si f_p es seleccionada. Si consideramos nuevamente n=2, *min*=0, y *max*=1 las configuraciones posibles serían:

- σ = *TSeleccionfp TSeleccionf1*. El marcado resultante es: *M*(*Pf1Seleccionada*)=1, *M*(*Pf2Seleccionada*)=0, representa la configuración $\{f_p, f_1\}$.
- σ = *TSeleccionfp TSeleccionf2*. El marcado resultante es: *M*(*Pf1Seleccionada*)=0, *M*(*Pf2Seleccionada*)=1, representa la configuración $\{f_p, f_2\}$.
- σ = *TSeleccionfp TNoSeleccionfpAlternativaf1fn*. El marcado resultante es: *M*(*Pf1Seleccionada*)=0, *M*(*Pf2Seleccionada*)=0, representa la configuración $\{f_p\}$.

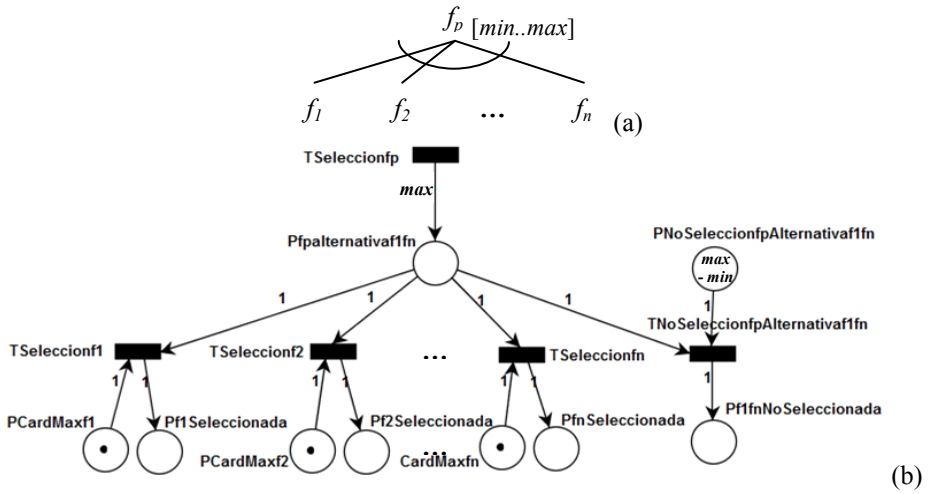

Fig. 4. Dependencia de variabilidad alternaiva. (a) modelo FM incluyendo n features alternativas. (b) red de Petri propuesta.

3.5 Dependencia de restricción requiere

En una dependencia requiere la consideración de una feature está sujeta a la inclusión de la feature vinculada. La Fig. 5 (a) muestra que la feature f_1 *requiere* a la feature f_2. La topología propuesta para este caso se presenta en la Fig. 5 (b). Las transiciones *TSeleccionf1* y *TSeleccionf2* corresponden a los eventos de selección de las features f_1 y f_2, respectivamente. El lugar *Pf1requieref2* representa la restricción. El disparo de *TSeleccionf2* (evento condicionante) deposita una marca en *Pf1requieref2* habilitando la transición *TSeleccionf1* (evento condicionado).

Fig. 5. Dependencia de restricción requiere. (a) modelo FM donde la feature f_1 requiere a la feature f_2. (b) red de Petri propuesta.

3.6 Dependencia de restricción excluye

Una dependencia excluye establece que dos features son mutuamente excluyentes. La Fig. 6 (a) ilustra una relación *excluye* entre las features f_1 y f_2. La Fig. 6 (b) especifica la topología propuesta para este caso. Las transiciones *TSeleccionf1* y *TSeleccionf2* corresponden a los eventos de selección de las features f_1 y f_2, respectivamente. El lugar *Pf1excluyef2* representa la restricción. La marca en *Pf1excluyef2* habilita las transiciones *TSeleccionf1* y *TSeleccionf2*, pero sólo una de ellas puede ser disparada.

Fig. 6. Dependencia de restricción excluye. (a) modelo FM de la relación excluye entre dos features. (b) red de Petri propuesta.

4 Definición formal de RP_{FM}, conceptos y propiedades

En esta sección especificamos formalmente la red RP_{FM} y la interpretación de algunos conceptos y propiedades de las RPs en el contexto de FMs.

4.1 Definición formal de RP_{FM}

En la Tabla 3 se define una RP_{FM} a partir de la definición de un *FM*. La notación empleada representa el conjunto de lugares con L, el conjunto de transiciones con T y el conjunto de arcos con A. Dividimos los conjuntos de lugares L y transiciones T en subconjuntos acordes a su interpretación. Algunos de estos subconjuntos representan explícitamente elementos de FM, tales como L^F, L^M, L^O, L^G, L^E, y L^Q que representan features, dependencias obligatoras, opcionales, alternativas, restricciones requiere y excluyen, respectivamente. Los otros subconjuntos de lugares están vinculados a la

representación de la cardinalidad, la posibilidad de no selección de una feature y a la raíz del FM. Cada lugar de L^C es definido para indicar el número máximo de veces que una cierta feature puede ser incluida, en este caso el valor siempre es 1. L^A especifica la cantidad mínima de features que deben ser seleccionadas (o el número máximo de no selecciones). Las transiciones T^F representan la selección de features. En cambio, las transiciones T^N especifican la no selección de features; y son empleados para deshabilitar transiciones de la RP_{FM}.

Table 3. Definición formal de una RP_{FM}.

Sea un modelo de características *FM* definido como (F, r, M, O, G, C, E, Q), donde: $F= \{f_1, f_2, \ldots, f_n\}$ es un conjunto finito de features, y $r \in F$, es la feature raíz de *FM*. $M \subseteq F \times F$ es un conjunto de relaciones de dependencia obligatorias, $O \subseteq F \times F$ es un conjunto de relaciones de dependencia opcionales, $G \subseteq F \times \mathbb{P}(F)$ es un conjunto de relaciones de dependencia alternativas, C: $G \rightarrow \mathbb{N} \times \mathbb{N}$ es una función de cardinalidad, $E \subseteq F \times F$ es un conjunto de relaciones de restricciones excluyentes, $Q \subseteq F \times F$ es un conjunto de relaciones de restricciones requiere.
Una RP_{FM} es definida como una 5-tupla (L, T, A, P, M_0), donde: $L= L^F \cup L^R \cup L^M \cup L^O \cup L^G \cup L^N \cup L^C \cup L^A \cup L^E \cup L^Q$ $T= T^F \cup T^N$ $A \subseteq (L \times T) \cup (T \times L)$ es un conjunto finito de arcos, P: $A \rightarrow \mathbb{N}$ es una función de peso, M_0: $L \rightarrow \mathbb{N}_0$ es el marcado inicial, $L \cap T = \varnothing$ y $L \cup T \neq \varnothing$.
donde $L^F = \{PfiSeleccionada \mid i= 1..n\}$ features seleccionadas $L^R = \{P0\}$ raíz $L^M = \{Pfiobligatoriafj \mid (f_i, f_j) \in M\}$ dependencias obligatorias $L^O = \{Pfiopcionalfj \mid (f_i, f_j) \in O\}$ dependencias opcionales $L^G = \{Pfpalternativaf1fk \mid (f_p, A) \in G, A=\{f_1, \ldots, f_k\} \subset F\}$ dependencias alternativas $L^N = \{PfjNoSeleccionada \mid f_j = r \vee (\exists f_i \mid (f_i, f_j) \in O)\} \cup \{Pf1fkNoSeleccionada \mid (\exists f_p \mid (f_p, A) \in G, A=\{f_1, \ldots, f_k\} \subset F)\}$ features no seleccionadas (raíz, opcional o alternativa) $L^C = \{PCardMaxfj \mid (\exists f_p \mid (f_p, A) \in G, f_j \in A)\}$ cardinalidad máxima de cada feature en un grupo $L^A = \{PnoSeleccionfpAlternativaf1fk \mid (f_p, A) \in G, A=\{f_1, \ldots, f_k\} \subset F)\}$ cardinalidad máxima de features no seleccionables en un grupo $L^E = \{Pfiexcluyefj \mid (f_i, f_j) \in E\}$ restricciones excluye $L^Q = \{Pfirequierefj \mid (f_i, f_j) \in Q\}$ restricciones requiere $T^F = \{TSeleccionfi \mid i= 1..n\}$ evento selección de feature $T^N = \{TNoSeleccionfj \mid f_j = r \vee (\exists f_i \mid (f_i, f_j) \in O)\} \} \cup \{TNoSeleccionfpAlternativaf1fk \mid (\exists f_p \mid (f_p, A) \in G, A=\{f_1, \ldots, f_k\} \subset F)\}$ evento no selección de feature (raíz, opcional o alternativa)

Las condiciones descriptas en la Tabla 4 permiten representar reglas FM, y están vinculadas al marcado inicial y al número de arcos entrantes y salientes desde y hacia lugares y transiciones. Por ejemplo, el lugar P0 ($\in$ LR) no posee transiciones entrantes ($\bullet$P0 = $\varnothing$) y el número de marcas en el marcado inicial es 1 (M0(P0) = 1). Las transiciones salientes debe ser P0$\bullet$ = {TSeleccionf0, TNoSeleccionf0} y corresponden a la selección y no selección de la feature raíz del FM.

Table 4. Condiciones impuestas en Lugares y Transiciones de una RP_{FM}.

Condiciones en lugares:
$\bullet P0 = \varnothing$, $P0\bullet = \{TSeleccionf0, TNoSeleccionf0\}$, $M_0(P0) = 1$, $P0 \in L^R$
$\bullet l \subseteq T^F$, $\lvert \bullet l \rvert = 1$, $l\bullet = \varnothing$, $M_0(l) = 0$, $\forall\ l \in L^F$
$\bullet l \subseteq T^F$, $\lvert \bullet l \rvert = 1$, $l\bullet = \subseteq T^F$, $\lvert l\bullet \rvert = 1$, $M_0(l) = 0$, $\forall\ l \in L^M$
$\bullet l \subseteq T^F$, $\lvert \bullet l \rvert = 1$, $l\bullet = \subseteq T^F \cup T^N$, $\lvert l\bullet \rvert = 2$, $M_0(l) = 0$, $\forall\ l \in L^O$
$\bullet l \subseteq T^F$, $\lvert \bullet l \rvert = 1$, $l\bullet = \subseteq T^F \cup T^N$, $\lvert l\bullet \rvert > 1$, $M_0(l) = 0$, $\forall\ l \in L^G$
$\bullet l \subseteq T^N$, $\lvert \bullet l \rvert = 1$, $l\bullet = \varnothing$, $M_0(l) = 0$, $\forall\ l \in L^N$
$\bullet l = \varnothing$, $l\bullet \subseteq T^F$, $\lvert l\bullet \rvert = 1$, $M_0(l) = 1$, $\forall\ l \in L^C$
$\bullet l = \varnothing$, $l\bullet \subseteq T^N$, $\lvert l\bullet \rvert = 1$, $M_0(l) > 0$, $\forall\ l \in L^A$
$\bullet l = \varnothing$, $l\bullet \subseteq T^F$, $\lvert l\bullet \rvert = 2$, $M_0(l) = 1$, $\forall\ l \in L^E$
$\bullet l \subseteq T^F$, $\lvert \bullet l \rvert = 1$, $l\bullet \subseteq T^F$, $\lvert l\bullet \rvert = 1$, $M_0(l) = 0$, $\forall\ l \in L^Q$
Condiciones en transiciones:
$\bullet t \subseteq L^R \cup L^M \cup L^O \cup L^G \cup L^Q \cup L^E$, $\lvert \bullet l \rvert \geq 1$, $t\bullet = L^F \cup L^M \cup L^O \cup L^G \cup L^Q$, $\lvert l\bullet \rvert \geq 1$, $\forall\ t \in T^F$
$\bullet t \subseteq L^R \cup L^O \cup L^G \cup L^A$, $\lvert \bullet l \rvert \geq 1$, $t\bullet = L^N$, $\lvert l\bullet \rvert = 1$, $\forall\ t \in T^N$

4.2 Conceptos de RP aplicados a FM

El significado de algunos conceptos de las RPs nos permiten comprender la relación entre la dinámica de una RP y las configuraciones permitidas por un FM. En los siguíentes párrafos analizamos los conceptos de marcado, marca, y secuencia de disparo.

- Marcado. Un marcado es un *n*-vector donde *n* es el número total de lugares de la RP. La i-ésima componente de *M* (*M*(*i*)) representa el número de marcas en el lugar *i*. Cada *M* describe un producto específico (o configuración) de una LPS y las marcas indican qué features son incluidas. Los marcados de interés son aquellos sin transiciones habilitadas, es decir, aquellos donde se han realizado todas las decisiones en el proceso de configuración.
- Marca. La presencia de una marca en un lugar posee distintos significados según el concepto de FM asociado. Para el caso de feature, una marca en un lugar L^F indica que la feature representada por el lugar es incluido en la configuración. En una restricción de dependencia, la marca asegura que las restricciones mutuamente exclusivas (inclusivas) deshabiliten (habiliten) una transición luego del disparo de otra. En el caso de cardinalidad, el número de marcas en un lugar restringe el máximo o el mínimo de transiciones que pueden ser disparadas, cuando el lugar está vacío, no es posible seleccionar ninguna feature mas.
- Secuencia de disparo. La secuencia de disparo desde el marcado inicial M_0 a un marcado donde no exista transición habilitada representa la secuencia de eventos para alcanzar una configuración dada. Esta secuencia nos brinda información sobre la selección de features para conformar una posible configuración de una LPS.

4.3 Propiedades de la red RP_{FM}

En este trabajo nos centramos en aquellas propiedades de la red RP_{FM} que están relacionadas con las operaciones identificadas en la Sección 1. A continuación explica-

mos cómo las propiedades de acotación ("boundedness"), alcanzabilidad ("reachability"), y L1-viva o potencialmente disparable ("L1-live") permiten abordar los problemas de consistencia y satisfacibilidad de un FM de una LPS.

- Acotación. Una RP es *k*-acotada si, a partir de un marcado inicial M_0, para cualquier estado alcanzable el número de marcas en cualquier lugar no excede el valor *k*. En una red de Petri no acotada, sin importar la secuencia de disparo, no puede precisarse el límite superior de marcas de ninguno de sus lugares. Una RP_{FM} es *k*-acotada, siendo *k* el valor máximo de las cardinalidades máximas de las dependencias alternativas. En caso de no poseer dependencias alternativas la cota *k* es igual a 1. En consecuencia, y junto a la propiedad de alcanzabilidad, analizado a continuación, una RP_{FM} permite conocer todos los productos posibles (configuraciones posibles) derivables a partir de un *FM*.
- Alcanzabilidad. Dada una RP con un marcado inicial M_0, M_n es alcanzable si existe una secuencia de disparo σ que posibilite llegar de M_0 a M_n. Una forma de analizar esta propiedad es por medio de un grafo de alcanzabilidad. Un grafo de alcanzabilidad es un grafo dirigido donde los nodos representan los marcados y los arcos las transiciones que transforman un marcado en su sucesor. El nodo inicial corresponde al marcado inicial (M_0), los nodos intermedios a aquellos con al menos una transición habilitada y los nodos terminales a los que no tienen ninguna transición habilitada. En las redes de Petri *k*-acotadas se verifica que su grafo de alcanzabilidad es finito [10].
- L1-viva o potencialmente disparable. La propiedad "liveness" está vinculada con la ausencia de estancamientos o nodos muertos. Una RP "live" garantiza que es posible encontrar una secuencia para disparar cualquier transición. Esta propiedad asegura la completa ausencia de nodos muertos. Existen varios niveles de "liveness" [10]. El nivel L1-viva, también llamado potencialmente disparable, establece que una transición puede ser disparada en alguna secuencia al menos una vez. Si todas las transiciones cumplen con el nivel L1-viva, la RP es L1-viva.

A partir de las dos primeras propiedades, siendo que una RP_{FM} es acotada, podemos afirmar que 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 features. Esto es de utilidad para identificar todas las posibles configuraciones de una LPS y poder brindar soporte a las funciones (2) encontrar un producto y (3) obtener todos los productos. Además, en caso de ocurrir un cambio en el FM (y su RP_{FM}), el grafo resultante puede compararse con el grafo de la red anterior para detectar configuraciones viables en el modelo previo e inviables en el nuevo modelo.

La propiedad L1-viva es útil para abordar la función (1) detección de nodos muertos. Si toda transición es potencialmente disparable, cualquier feature será incluida al menos en una configuración y en consecuencia el FM no tendrá ninguna feature inviable.

5 Caso de Estudio

El siguiente caso de estudio describe parcialmente la variabilidad de una LPS del software vinculado a teléfonos móviles. En primer lugar, introducimos el FM, luego generamos la red RP_{FM} para representar el FM y analizamos sus propiedades.

En la Fig. 7 se presenta el modelo parcial FM de una familia de software vinculado a teléfonos móviles (por razones de espacio sólo incluye 7 features). El modelo fue realizado con la herramienta SPLOT [11]. La feature raíz del modelo es *MPhone*. Si esta feature es seleccionada, las features *Calls* y *Screen* también deben ser consideradas, ya que están vinculadas con *MPhone* por medio de una dependencia obligatora. En cambio, *GPS* y *Camera* son features opcionales. Desde la feature *Screen* se definen como features alternativas *Basic* y *HRes*, de las cuales se debe seleccionar una y solo una (cardinalidad [1,1] en Fig. 7). Las features *GPS* y *Basic* son mutuamente excluyentes y selección de la *Camera* requiere la inclusión de *HRes* (parte inferior de Fig. 7).

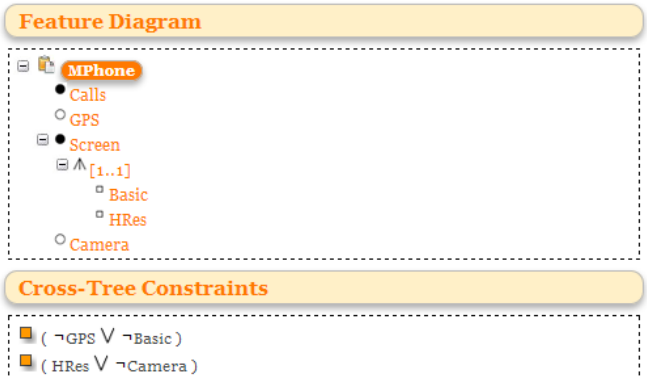

Fig. 7. FM parcial de teléfono móvil.

La Fig. 8 presenta la red RP_{FM} obtenida a partir del FM presentado en la Fig. 7. Para la asistencia en la generación de la red RP_{FM} desarrollamos una herramienta, FM2PN, que a partir de un FM generado mediante la aplicación SPLOT genera la red de Petri RP_{FM} según la topología descripta en la Sección 3 de este trabajo. FM2PN fue construida empleando el generador de analizador léxico JLex [12] y el generador de parser CUP [13], ambos basados en Java. Para el caso de estudio se obtuvo una red con 22 lugares, 11 transiciones, y 34 arcos. La feature *MPhone* es representada por el lugar *PMPhoneSeleccionada* (Fig. 8) y la no selección de esta feature es representada por *PMPhoneNoSeleccionada*. Las features obligatorias *Calls* y *Screen* son modeladas por los lugares *PCallsSeleccionada* y *PScreenSeleccionada*, respectivamente. Por las features opcionales *GPS* y *Camera* se incluyeron los lugares *PGPSSeleccionada* y *PCameraSeleccionada* representando su selección y los lugares *PGPSNoSeleccionada* y *PCameraNoSeleccionada* representando la no selección. La restricción excluye ($\neg GPS \vee \neg Basic$ en Fig. 7) es representada por el lugar *PGPSexcluyeBasic* y la restricción requiere ($HRes \vee \neg Camera$ en Fig. 7) por el lugar *PCamerarequiereHres*. Las transiciones *TSeleccionMPhone*, *TSeleccionCalls*, *TSeleccionGPS*, *TSeleccionS-*

creen, *TSeleccionBasic*, *TSeleccionHRes*, y *TSeleccionCamera* (*TNoSeleccionMPhone*, *TNoSeleccionGPS*, *TNoSeleccionScreenAlternativaBasicHRes*, y *TNoSeleccionCamera*) representan la selección (no-selección) de las features.

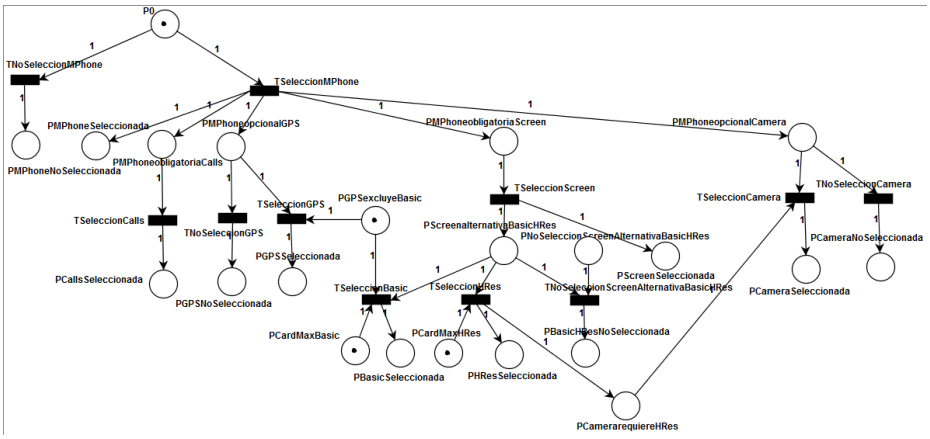

Fig. 8. RP_{FM} del FM de teléfono móvil.

FM2PN genera la red RP_{FM} en un archivo en formato xml. Dicha red es luego analizada para detectar los problemas de inviabilidad empleando la aplicación PIPEv2 (Fig. 8, Fig. 9, Fig. 10). PIPEv2 [14] permite construir entre otras cosas el grafo de alcanzabilidad (Fig. 9) de una RP dada, así como simular la red (Fig. 10).

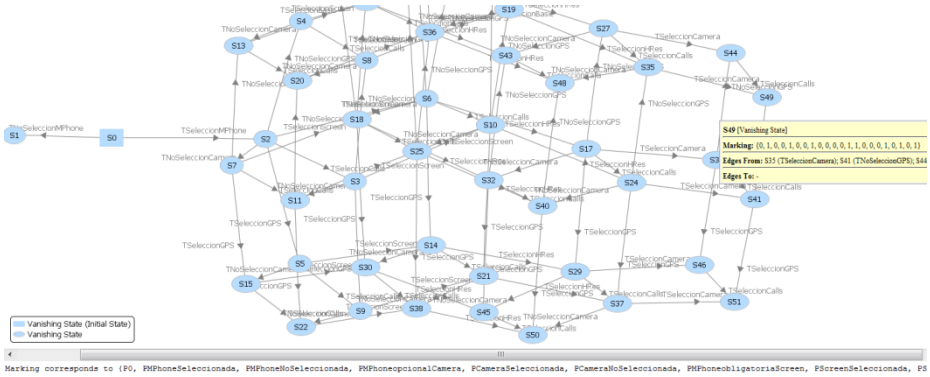

Fig. 9. Vista parcial del grafo de alcanzabilidad.

El nodo *S0* (Fig. 9, ilustrado como un rectángulo) representa el marcado inicial (M_0) y los nodos terminales representan las distintas configuraciones en la LPS, por ejemplo el nodo *S49* de la Fig. 9, posee el marcado {0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1} donde los lugares marcados son: *PMPhoneSeleccionada, PCallsSeleccionada, PGPSNoSeleccionada, PGPSExcluyeBasic, PCardMaxBasic, PHResSeleccionada, PScreenSeleccionada,* y *PCameraSeleccionada*. Tal marcado representa la

selección de las features *MPhone*, *Calls*, *HRes*, *Screen*, y *Camera*, es decir, la configuración {*MPhone*, *Calls*, *HRes*, *Screen*, *Camera*}. Además, el grafo de alcanzabilidad provee información de la secuencia de evento que permitió alcanzar tal configuración. En particular, una de las secuencias posibles para alcanzar la configuración representada por el nodo *S49* está dada por los eventos σ = *TSeleccionMPhone TNoSeleccionGPS TSeleccionCalls TSeleccionScreen TSeleccionHRes TSeleccionCamera*, secuencia de disparo que es ilustrada en la Fig. 10, mostrando el marcado final.

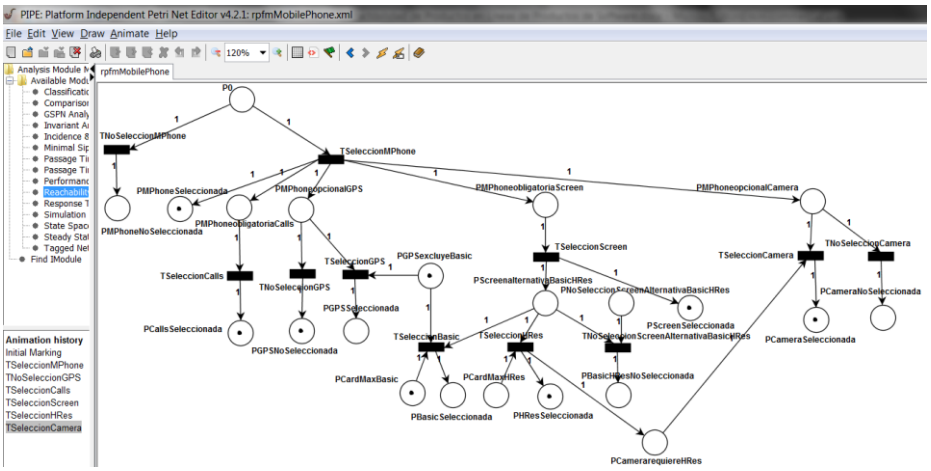

Fig. 10. Marcado resultante de aplicar σ = *TSeleccionMPhone TNoSeleccionGPS TSeleccionCalls TSeleccionScreen TSeleccionHRes TSeleccionCamera* en M_0.

Con respecto a satisfacibilidad, el conjunto de configuraciones permitidas es {∅, {*MPhone*, *Calls*, *Screen*, *Basic*}, {*MPhone*, *Calls*, *Screen*, *HRes*}, {*MPhone*, *Calls*, *Screen*, *HRes*, *Camera*}, {*MPhone*, *Calls*, *Screen*, *GPS*, *HRes*}, {*MPhone*, *Calls*, *Screen*, *GPS*, *HRes*, *Camera*}}.

6 Conclusiones y Trabajos Futuros

En este trabajo se especificó RP_{FM}, la definición de la topología de una subclase de RP que permite la representación y el estudio formal de los FMs durante el desarrollo y evolución de una LPS. El análisis se centra en las propiedades de las RPs, en particular acotación, alcanzabilidad, y potencialmente disparable proveen una base sólida para analizar las funciones de satisfacibilidad y consistencia planteadas en [4] y [6]. El modelo propuesto fue implementado en una herramienta, FM2PN, la cual permite generar a partir de un FM la red de Petri RP_{FM} y realizar el análisis con herramientas de análisis de RPs.

Un desafío interesante es el tamaño de los FM. La tendencia ascendente en el número de features [6] requiere de técnicas de transformación para reducir y simplificar las RPFM y los grafos de aclanzabilidad, manteniendo sus propiedades. También se plantea extender el estudio de otras propiedades de RP en el marco de una

RPFM. Entre ellas se puede mencionar la reversibilidad y la distancia sincrónica. La primera permite recuperar el marcado inicial desde cualquier otro marcado, en cierto sentido, posibilita la reconstrucción de una RPFM a partir de sus configuraciones. La distancia sincrónica es una métrica del grado de relación entre transiciones y puede emplearse para obtener información cualitativa sobre las dependencias del FM subyacente.

Agradecimientos. Se agradece el apoyo financiero brindado por CONICET, la Universidad Tecnológica Nacional y Agencia Nacional de Promoción Científica y Tecnológica (PAE-PICT-2007-02315).

Referencias

1. Northrop, L., Clements, P., Bachmann, F., Bergey, J., Chastek, G., Cohen, S. Donohoe, P., Jones, L., Krut, R., Little, R., McGregor, J., O'Brien, L.: A framework for product line practice, version 5.0, http://www.sei.cmu.edu/productlines/frame_report/index.html (2009)
2. van der Linden, F., Schmid, K., Rommes, E.: Software product lines in action: the best industrial practice in product line engineering. Springer Heidelberg (2007)
3. Pohl, K., Böckle, G., van der Linden, F.: Software product line engineering: foundations, principles, and techniques, Springer: Heidelberg (2005)
4. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA). Feasibility study, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990)
5. Schobbens, P., Trigaux, J., Heymans, P., Bontemps, Y.: Generic semantics of feature diagrams. Computer Networks 51, 456-479 (2007)
6. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: A literature review, Journal of Information Systems 35, 615-636 (2010)
7. Batory, D.: Feature models, grammars, and propositional formulas. In: Software Product Lines Conference. LNCS, vol. 3714, pp. 7–20 (2005)
8. Sun, J., Zhang, H., Li, Y. Wang, H.: Formal semantics and verification for feature modeling. In: Proceedings of the ICECSS05, pp. 303-312 (2005)
9. Benavides, D., Ruiz-Cortés, A., Trinidad, P.: Using constraint programming to reason on feature models. In: 17th International Conference on Software Engineering and Knowledge Engineering, pp. 677-682 (2005)
10. Murata, T.: Petri Nets: properties, analysis and applications. In: proceedings of the IEEE, Vol. 77:4, pp.541-580 (1989)
11. M. Mendonca, M. Branco, D. Cowan, “S.P.L.O.T. - Software Product Lines Online Tools”, In 24th ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, Orlando, Florida, USA, 2009.
12. E. Berk, C. Scott Ananian, “JLex: A Lexical Analyzer Generator for Java”, disponible en: http://www.cs.princeton.edu/~appel/modern/java/JLex/
13. S. Hudson, F. Flannery, C. Scott Ananian, “CUP Parser Generator for Java”, disponible en: http://www.cs.princeton.edu/~appel/modern/java/CUP/
14. N. Dingle, W. Knottenbelt, T. Suto, “PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets”, ACM SIGMETRICS Performance Evaluation Review, Vol. 36(4), 34-39, 2009.