UML y REDES DE PETRI EN LA EVALUACION DE PERFORMANCE DE SISTEMAS

Santiago Pérez, Mario Distefano, Antonio Paseo, Atilio Ranzuglia
{santiagocp, mdistefa, apasero, aranzuglia}@frm.utn.edu.ar

MIP
(Grupo de Modelos Industriales Paralelos)
Facultad Regional Mendoza,
Universidad Tecnológica Nacional
Mendoza Argentina
0261-4239119 (int. 176)

I) Resumen

En el estudio de la Ingeniería de Software se ha destinado un gran esfuerzo por parte de distintos grupos de investigación, para incorporar a las especificaciones de software, ciertos atributos no funcionales relacionados con la performance, tiempo, fiabilidad, planificación, entre otros. Específicamente, debido al vacío importante entre el diseño de software y el análisis de performance, la Ingeniería de Performance de Software (SPE-Software Performance Engineering [1]) considera el análisis cuantitativo de la conducta de los sistemas de software, desde las fases iniciales de desarrollo. Varios lenguajes han sido propuestos para tal fin, y existen trabajos proponiendo especificar performance a partir del lenguaje de modelado UML [2]. Estos esfuerzos han llevado a la adopción del perfil de UML para la planificación, performance y comportamiento temporal del software UML-SPT [3], teniendo en cuenta también que el lenguaje UML progresivamente se ha convertido en un estandar universal para la modelación de software. Es un lenguaje semiformal y esta respaldado por el Object Management Group (OMG)[4]. La especificación de performance de software con un lenguaje semiformal como UML, requiere su integración con un formalismo de modelación de performance, como las Redes de Petri Estocásticas Generalizadas (GSPN-Generalize Stochastic Petri Nets [5][6]). Sobre esta base, se puede construir una herramienta CASE que permita el traslado de los diagramas UML a GSPN, y a partir de ello, la evaluación de performance de sistemas. Se toma como antecedente, el plug-in ArgoSPE [7] sobre ArgoUML [8] de la Universidad de Zaragoza (España), que genera archivos de GSPN en el formato de la herramienta de simulación y análisis de performance GreatSPN [9] de la Universidad de Turín (Italia).
Se describen estas herramientas y su necesidad para la evaluación de performance de sistemas.

I1) Introducción

El diseño e implementación de sistemas complejos, en general, es una difícil tarea de ingeniería. En los últimos años, la modelación, validación, evaluación de performance e implementación de tales sistemas ha sido fortalecida con la ayuda de modelos formales.
Las GSPN son un adecuado paradigma formal para soportar el ciclo de vida completo de un sistema de evento discreto complejo. Han sido usadas para la modelación y la evaluación de sistemas de fabricación flexibles, arquitecturas multiprocesador, sistemas de comunicación, y también para la escritura de programas concurrentes eficientes y confiables.
En las aplicaciones de sistemas de software, los requerimientos funcionales son obviamente importantes, pero no son los únicos. Los objetivos de performance son también importantes. Es decir,: el grado en que un sistema de software satisface sus objetivos de tiempo, aspecto que se vuelve crítico en algunas aplicaciones de tiempo real. Dado que la Ingeniería de Software es una disciplina relativamente joven, se han asumido y reconocido la importancia del uso de metodologías, métodos formales, lenguajes

y herramientas de desarrollo bien establecidas en el tiempo. Sin embargo, los objetivos de performance no están usualmente incluidos en las primeras etapas del ciclo de vida del software. Siendo los requerimientos de performance críticos para el éxito de los sistemas de software de hoy en día, varios productos de software finales fallan para cumplir aquellos requerimientos.. Por ello, varios investigadores defienden el principio que la performance debería ser incluida en el proceso de diseño de software desde muy temprano. El campo de investigación que trata con el objetivo de construir software con performance predecible, especificando y analizando la conducta cuantitativa desde las fases de desarrollo iniciales de un sistema a través de su ciclo de vida entera ha sido reconocida con el termino Software Performance Engineering (SPE). El lenguaje de modelación unificado (UML) combinado con una metodología orientada a objetos, es hoy en día la aproximación más ampliamente usada en la comunidad de Ingeniería de Software. Así, la mayoría de herramientas CASE soportan la metodología orientada a objetos, y usan UML como el lenguaje de diseño.

Dado que los objetivos de performance no están incluídos en la práctica usual de los ingenieros de software, se puede pensar que existe una necesidad de integrar un modelo de performance con las metodologías de desarrollo de software existentes. Entre las herramientas de modelación formales se pueden enumerar las cadenas de Markov, las redes de colas, las álgebras de procesos estocásticos y las Redes de Petri Estocásticas Generalizadas, siendo probablemente los paradigmas de modelación de performance mejor estudiados. De ellos, las GSPN son un instrumento especial para modelar sistemas paralelos y distribuídos, por su simplicidad matemática, su generalidad de modelación, su adecuación para expresar todas las semánticas básicas de concurrencia, su ubicación para estados y acciones y representación gráfica, sus bien desarrolladas técnicas de análisis cualitativo y cuantitativo, y la existencia de herramientas de análisis.

Existen experiencias en el proceso de integración de la modelación de performance dentro del proceso de desarrollo de software. A continuación se resumen las fases principales de un proceso de ingeniera de performance de software apropiado, basado en UML y GSPN.

III) Proceso de Performance de Software

Algunos trabajos se han propuesto para combinar UML y formalismos de modelación de performance, para analizar aspectos cuantitativos de sistemas de software. Todos comparten algunos principios básicos, y se podría argumentar que existe un proceso ampliamente aceptado entre la comunidad de SPE:

- La conducta y arquitectura del sistema se describe a través de un conjunto de diagramas UML, que corresponden al diseño del sistema,
- Este diseño UML es anotado en performance de acuerdo a un perfil OMG estándar, llamado el diseño anotado,
- El diseño anotado se convierte en un formalismo de modelación de performance,
- Se lleva a cabo un análisis cuantitativo, si el modelo formal lo permite,
- El modelo formal se analiza usando técnicas de análisis cuantitativas ya desarrolladas para el formalismo adoptado.

Entre los diagramas UML están incluidos: los casos de uso (UC), los diagramas de secuencia (SD), los diagramas de actividad (AD), los diagramas de estado (SC), y los diagramas de despliegue (DD).

La carga de trabajo, las utilizaciones, los tiempos de respuesta o el rendimiento caracterizan la vista de performance del diseño del sistema, el denominado diseño anotado. El perfil UML para especificación de la planificación, performance y tiempo (UML-SPT) es el estandard ampliamente usado para anotarlos.

IV) UML y ArgoSPE

ArgoSPE es una herramienta para la evaluación de performance de sistemas de software, y ha sido implementado como un conjunto de módulos Java, que son plugged en la herramienta de código abierto ArgoUML (figura nº 1), siguiendo la arquitectura propuesta en el estándar UML-SPT (figura nº 2).

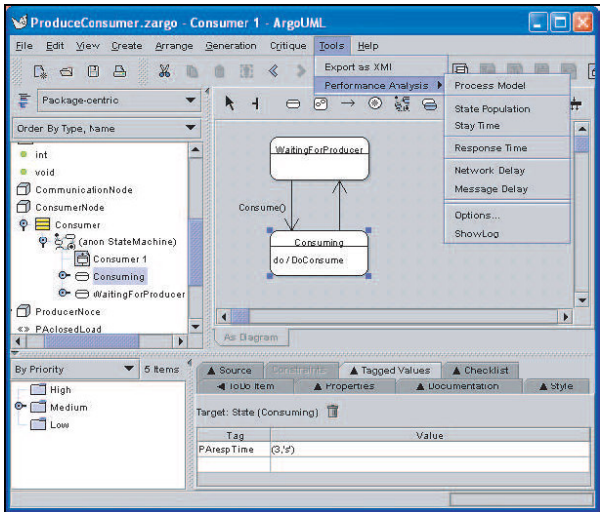

figura nº 1

Desde el punto de vista del usuario, ArgoSPE es un conjunto de consultas de performance que pueden ejecutarse para obtener el análisis cuantitativo del sistema modelado. Se entiende que una consulta de performance es un procedimiento donde el modelo UML se analiza para obtener automáticamente un índice de performance predefinido. Los pasos de este procedimiento se ocultan al usuario. Cada consulta de performance está relacionada a un diagrama UML donde es interpretada, pero su computo se obtiene en un modelo GSPN automáticamente para ArgoSPE. Para el analista en performance que tiene experiencia en modelación y análisis de Redes de Petri, puede usar directamente la herramienta GreatSPN para computar las métricas especificadas usando los modelos GSPN, que ArgoSPE genera automáticamente.

Como se indicó, ArgoSPE sigue la propuesta de UML-SPT. La herramienta CASE ArgoUML trabaja como el editor del modelo, mientras que el módulo ArgoSPE implementa y coordina las funciones: configurador del modelo, y el procesador del modelo (conversor del modelo, analizador del modelo y el conversor de resultados).

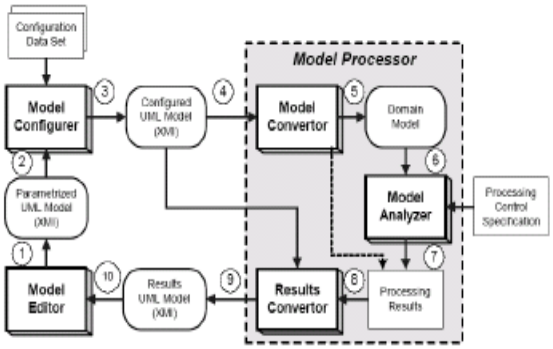

figura nº 2

V) GreatSPN

El modelo GSPN es una extensión de las Redes de Petri Ordinarias [10], donde están definidas transiciones temporizadas con retardos exponenciales negativos y además transiciones inmediatas. Es el modelo matemático utilizado por ArgoSPE

Formalmente un modelo GSPN es una 10-tupla [5]

$$M_{GSPN} = \{P, T, I, O, H, \Pi, W, PAR, PRED, MP\}$$

- **P** es un conjunto de lugares.
- **T** es un conjunto de transiciones.

  $T \cap P = \phi$

- **I, O y H** son entradas, salidas y función inhibición asociadas a **T**. Es un multiconjunto en **P**.
- Π es la función de prioridad de las transiciones que representa el nivel de prioridad con un número natural.
- **W** Es una función que introduce el componente estocástico en el modelo GSPN. Es la tasa en una transición temporizada (rate) y el peso (weight) en una transición inmediata.
- **PAR** es un conjunto de parámetros.
- **PRE** es un conjunto de restricciones del rango de parámetros.
- **MP** es una función asociada con cada lugar, un número o bien un rango de

parámetros del conjunto de los números naturales (marcado).
En una aplicación concreta, **PAR y PRE** están incorporados en el marcado inicial. Por tanto el modelo GSPN se reduce a una 8-tupla. En cualquier marcado todos los retardos de disparo de las transiciones temporizadas tiene una exponencial negativa (función densidad de probabilidad PDF) y todos los retardos independientes son variables aleatorias.
Varios pasos deben ser seguidos en el estudio de un sistema con GSPN, a saber:
1) El modelo debe ser construido, posiblemente usando una técnica estructurada, ya sea de arriba hacia abajo o de abajo hacia arriba, dependiendo del sistema a ser modelado.
2) El modelo luego debe ser validado usando los resultados del análisis estructural, proveyendo algunas propiedades de la conducta del modelo.
3) Los índices de performance de interés deben ser definidos en términos de marcado y de disparo de transiciones del GSPN.
4) El conjunto y el grafo de alcanzabilidad corresponde a una cadena de Markov de tiempo continuo (CTMC) [11].
5) La cadena de Markov se resuelve por:

$$\pi Q = 0$$

$$\sum_i \pi_i = 1$$

Donde Q es la matriz de tasa (tiempo probabilístico exponencial) de transición. Donde π es el vector de probabilidad en el estado estacionario. Es un sistema de ecuaciones lineales. Cada incógnita corresponde a un estado de marcado posible en el diagrama de estado.
6) Los índices de performance deben ser computados de la solución de la cadena de Markov en función del modelo de la aplicación; por ejemplo: tiempo medio de espera de un servicio, tiempo de ocupación de una memoria compartida, tiempo medio de ocupación de una máquina o procesador, rendimientos de productividad, índices de costos operativos, etc..
Todos estos pasos deben realizarse con una herramienta de software conveniente que provea el ambiente de modelado con GSPN, permitiendo la creación gráfica y modificación de un modelo, la definición de índices de performance, análisis estructural y el análisis estocástico del modelo. Uno de estos programas es el GreatSPN [12].
Se incluye el ejemplo del modelo GSPN para lectores-escritores que requieren acceder a una base de datos (figura nº3). El modelo GSPN se compone de 13 lugares y 13 transiciones, 5 temporizadas y 8 inmediatas. El marcado paramétrico inicial asigna K marcas a $P_{think,}$ o sea que K procesos son inicialmente asignados a sus memorias privadas. Una marca en P_{db} indica que la base de datos está desocupada. Una marca PLAN indica que la LAN esta inicialmente disponible. La transición T_{exec} modela la fase de ejecución de los procesos en su memoria privada; y es temporizada con el parámetro λ.

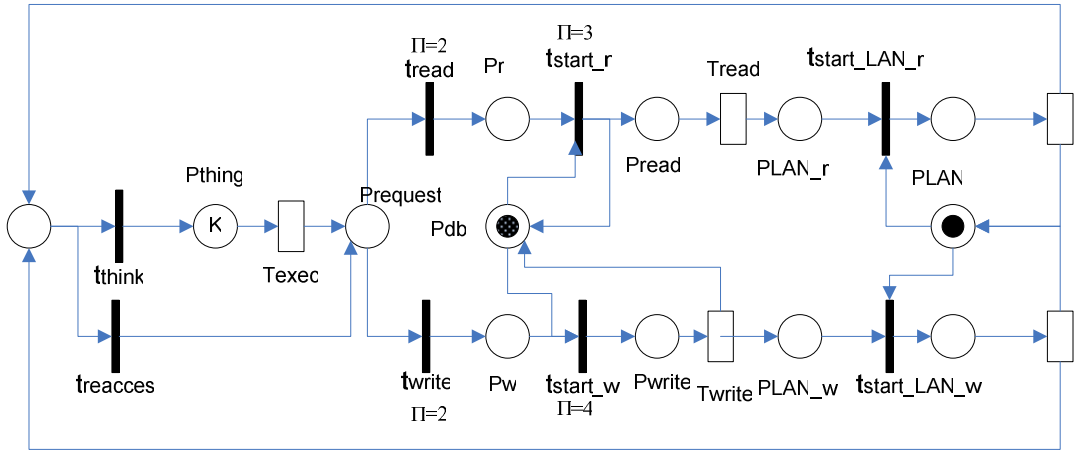

figura nº 3

La variación del tiempo de ejecución en la memoria privada es $1/\lambda$ y donde todas las ejecuciones proceden en paralelo con la semántica de infinito-servidor.
El disparo de T_{exec} modela el final de la fase de ejecución y la generación de un requerimiento de acceso a la base de datos colocando una marca en $P_{request}$. Un conflicto de libre-elección comprende las transiciones inmediatas t_{read} y t_{write} de niveles de prioridad 2, se modela el hecho que el requerimiento del acceso a la base de datos puede ser para leer o para escribir, generando una marca en p_r o en p_w respectivamente. Asumiendo que el 80% de los requerimientos son para leer y solamente el 20% para escribir, los pesos en t_{read} y t_{write} son 8 y 2 respectivamente.
Las prioridades de $t_{start-r}$ y $t_{start-w}$ se ponen en 3 y 4 respectivamente, tal que el requerimiento a escribir tiene prioridad sobre el de leer. El peso asociado a las dos transiciones es irrelevante por lo tanto se ponen en 1. El lugar p_{writw} puede contener a lo sumo una marca, y la transición T_{write} es temporizada con una tasa de μ_w con lo cual es inverso a la variación del tiempo de escritura; la semántica de esta transición es de simple-servidor. El disparo de T_{write} modela el final del acceso a la base de datos para una escritura y genera una marca en P_{LAN-w}. El lugar p_{read} puede contener varias marcas y la transición T_{read} es temporizada con una tasa μ_r; la semántica es de infinitos-servidores, donde todos los accesos a lectura son en paralelo. El disparo de T_{read} modela el final del acceso a la base de datos y coloca una marca en P_{LAN-r}. Las transiciones $t_{start-LAN-r}$ y $t_{start-LAN-w,}$ tienen igual prioridad de nivel 1 y esto forma un conflicto de no libre-elección de transiciones inmediatas. Sus pesos son ambos puesto a 1, esto indica que la asignación de la LAN a procesos que han completado una lectura o escritura en la base de datos es igualmente probable. Las transiciones T_{LAN-r} y T_{LAN-w} modela la actividad de transferencia de datos a la LAN. Ellas son temporizadas con tasas de ρ_r y ρ_w, respectivamente y sus semánticas es de simple-servidor, donde no mas de una LAN transfiere actividad a un tiempo. Transiciones t_{think} y $t_{reaccess}$ tienen igual prioridad 1. y forma un conflicto de libre-elección de transiciones inmediatas. Sus pesos son 9 y 1 respectivamente, indica que 10% del tiempo de los procesos inmediatamente reinicia un requerimiento de acceso a la base de datos después de completar el acceso previo, mientras el 90% del tiempo de los procesos vuelven a ejecutarse en la memoria privada.

VI) Referencias

1. Smith, C.U.: *Perf. Engineering of Software Systems.* Addison Wesley (1990)
2. United Modeling Language Specification. (http://www.uml.org)
3. UML Profile for Schedulabibity, Performance and Time Specification. (http://www.uml.org).
4. Object Management Group. (http:/www.omg.org).
5. Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: *Modelling with Generalized Stochastic Petri Nets*. John Wiley Series (1995)
6. Lopez-Grao, J.P., Merseguer, J., Campos, J.: From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering. (In: ACM WOSP'04) 25-36
7. The ArgoSPE project (http://argospe.tigris.org)
8. The ArgoUML project (http://argouml.tigris.org)
9. The GreatSPN tool (http://di.unito.it/~greatspn)
10. Manuel Silva – *Las Redes de Petri en la Automática y la Informática* – Editorial AC –España – 1985.
11. S.M. Ross – *Probability Models for Computer Science* - University of California Berkeley,CA - A Harcourt Science and Technology Co – 2002.
12. GreatSPN User`s Manual – Departamento de Informática - Universidad de Torino