LeGESD: Un marco de trabajo para la especificación y validación formal de sistemas concurrentes y distribuidos basado en un lenguaje gráfico con semántica sustentada en el álgebra de procesos

LeGESD: A framework oriented to the specification and formal validation of concurrent and distributed systems based on a graphical language and its process algebra semantics

Jorge Cortés Galicia[1], Felipe R. Menchaca García, Rolando Menchaca Méndez*

Centro de Investigación en Computación. Instituto Politécnico Nacional. Av. Juan de Bátiz s/n esq. Miguel Othón de Mendizabal. Ciudad de México, México.

(Recibido el 26 de mayo de 2010. Aceptado el 1 de junio de 2012)

Resumen

La especificación y validación formal de sistemas distribuidos es en general una tarea compleja debido a que requiere conocimientos profundos tanto en el área de teoría de algoritmos, como en el modelado de sistemas concurrentes o distribuidos. En este contexto, presentamos *LeGESD,* un marco de trabajo orientado a facilitar la especificación y validación formal de sistemas concurrentes y distribuidos. LeGESD está integrado por un lenguaje gráfico formal para la especificación y análisis de sistemas distribuidos, en el cual es posible incluir tanto los requerimientos funcionales como de comunicación del sistema que se está especificando. La semántica del lenguaje gráfico propuesto incorpora al álgebra de procesos en su definición. Esta semántica también está descrita en el presente documento y se le ha denominado como *Análisis y Diseño de Sistemas Distribuidos* (ADSD). ADSD es una especificación algebraica con semántica operacional definida para LeGESD, la cual presenta relaciones gráfico-algebraicas de equivalencia que son utilizadas en la especificación formal realizada con LeGESD. Finalmente, en el presente artículo se desarrolla un ejemplo que muestra la utilización y potencialidad tanto del lenguaje como de su semántica asociada.

* Autor de correspondencia: teléfono: + 52 + 555 + 729 60 00-52039, fax: + 52 + 555 + 729 60 00-56607, correo electrónico: jcortesg@ipn.mx (J. Cortés)

--------- *Palabras clave:* Especificación, sistemas distribuidos, álgebra de procesos

Abstract

The specification and formal verification of distributed systems is usually a complex task. It requires extensive knowledge of algorithm theory and modeling of distributed or concurrent systems. In this context, we present LeGESD which is a framework oriented to support the specification and formal validation of concurrent and distributed systems. LeGESD has a graphical formal language for the specification and analysis of distributed systems which is used to describe functional and of communication requirements. The semantics of LeGESD is based on the process algebra. This semantics is also described in this paper and it is called *Analysis and Design of Distributed Systems* (ADSD). ADSD is an algebraic specification with operational semantics defined to be used with LeGESD. ADSD provides graphical-algebraic equivalence relations which can be used to generate formal specifications of distributed systems. Lastly, this paper presents an example which shows the usefulness and potential of the language and its semantics.

--------- *Keywords:* Specification, Distributed systems, process algebra

Introducción

En la actualidad, debido a su complejidad tanto desde el punto de vista matemático como de ingeniería, así como a su impacto tanto en la industria como en la vida diaria, los sistemas distribuidos se han establecido como una de las líneas de investigación y desarrollo más activas e interesantes dentro de las ciencias de la computación. Estos sistemas presentan en su diseño, especificación, validación y construcción un reto de complejidad elevada, lo que ha dado lugar a un cúmulo de trabajos en los que se proponen herramientas y formalismos de especificación de sistemas distribuidos que ayuden a hacer más eficiente todo su ciclo de desarrollo [1, 2]. Uno de los aspectos más atractivos de especificar formalmente un sistema distribuido, es que a partir de dicha especificación es posible desarrollar técnicas y herramientas de construcción y validación automática. Una infraestructura de este tipo proporciona muchas ventajas al desarrollo de sistemas distribuidos ya que reduciría enormemente los esfuerzos y costos relacionados con su construcción. Más aún, los sistemas generados con herramientas automáticas son en general menos propensos a tener errores de programación lo que los haría más robustos y estables.

La construcción automática de sistemas distribuidos requiere necesariamente de varias etapas que auxilien durante todo el proceso de construcción; estas etapas incluyen de manera general los siguientes aspectos: (1) Un lenguaje de especificación de sistemas distribuidos. (2) Un verificador de modelos. (3) Un generador de código fuente para uno o varios paradigmas de programación. (4) Un simulador de modelos. Las tres primeras etapas son la base necesaria para una adecuada construcción de todo sistema distribuido, mientras que la última etapa es útil para realizar análisis de desempeño y experimentos preliminares.

En la construcción de sistemas distribuidos se emplean diferentes técnicas y herramientas para la especificación y generación de código como son los árboles sintácticos y semánticos, y los

autómatas de entrada/salida. Es en este contexto que se ha propuesto LeGESD (Lenguaje Gráfico de Especificación de Sistemas Distribuidos) [3] como herramienta para la especificación de sistemas distribuidos. LeGESD es un marco de trabajo conformado por un lenguaje gráfico usado para especificar de manera sencilla dichos sistemas. Los lenguajes gráficos presentan como principales fortalezas su sencillez, que son intuitivos y que su curva de aprendizaje es más elevada que la correspondiente a los lenguajes textuales. Esto se debe a la capacidad del cerebro humano para asociar ideas o construcciones mentales con elementos gráficos, como lo demuestran los mapas mentales, mapas conceptuales, y los árboles gráficos, los cuales son ampliamente utilizados en los modelos pedagógicos como herramientas de aprendizaje significativo [4]. La computación no está exenta del potencial de las representaciones gráficas, teniendo su mejor ejemplo en UML [5]. En LeGESD se reconoce este potencial y se utiliza como base de su lenguaje de especificación.

Para todo lenguaje de especificación es conveniente contar con un formalismo que respalde al lenguaje. Esta base formal es útil para proporcionar un análisis sobre la especificación del diseño elaborado a través del lenguaje [6]. Existen algunos métodos formales propuestos para la especificación de sistemas distribuidos, los cuales han usado notaciones textuales o gráficas, tal es el caso de Macedon [7], IOA [8], LfP [9], y PEDS [10]. Sin embargo, varios de ellos no cuentan con un formalismo que sustente al método. Esto ha motivado la utilización del algebra de procesos [11] para el desarrollo de un formalismo que sustente al lenguaje de LeGESD, y que apoye en la especificación y el diseño de sistemas distribuidos, combinando un método gráfico con uno algebraico.

Comparativamente, el álgebra de procesos presenta un mayor potencial para la especificación y verificación de sistemas distribuidos que otras técnicas como podrían ser las redes de petri, en las cuales es necesario generar extensas y complejas construcciones gráficas para la especificación y verificación de estos sistemas. Mientras que por otro lado, el álgebra de procesos simplifica significativamente la extensión y complejidad de la especificación resultante y de su verificación correspondiente. Lo anterior debido a las propiedades de las ecuaciones que se generan a través de la aplicación de sus operadores sobre los procesos involucrados en el sistema. Mediante estas ecuaciones, es posible realizar las actividades de verificación de forma más eficiente ya que las representaciones algebraicas son compactas. El álgebra de procesos aplicada en LeGESD tiene dos motivaciones principales: la primera, es que proporciona un formalismo que permite la rápida verificación de los modelos especificados en LeGESD de forma consistente y bien definida; la segunda, es que proporciona una semántica operacional sólida [12] a LeGESD. Esta semántica operacional está definida por el *Análisis y Diseño de Sistemas Distribuidos* (ADSD), que es una especificación a nivel algebraico que define una semántica operacional, así como relaciones de equivalencia entre la notación gráfica en LeGESD y la notación algebraica (notación textual) en ADSD. Estas relaciones de equivalencia son usadas durante la construcción del sistema distribuido y son de gran utilidad en las etapas de verificación del modelo y en la generación automática de su código fuente.

El resto del presente documento está organizado de la siguiente manera. Inicialmente se describe a LeGESD, enseguida a su semántica (ADSD), y después se muestra un ejemplo de la transformación entre ambos. Posteriormente se presentan los trabajos relacionados más relevantes, y finalmente, se proporcionan las conclusiones.

LeGESD

El Lenguaje Gráfico de Especificación para Sistemas Distribuidos (LeGESD) es un lenguaje visual formal para la especificación de sistemas distribuidos. LeGESD incluye los requisitos funcionales y de comunicación necesarios para el diseño de cualquier sistema distribuido, además de que el lenguaje permite una especificación modular, jerárquica y escalable. LeGESD

proporciona una notación gráfica que define la comunicación y aspectos dinámicos del comportamiento del sistema. La semántica de LeGESD se basa en una especificación algebraica con semántica operacional basada en el álgebra de procesos [13, 14] a la cual hemos denominado como Análisis y Diseño de Sistemas Distribuidos (ADSD). ADSD proporciona las relaciones de equivalencia del comportamiento del sistema que se pueden utilizar para verificar la correcta especificación realizada con LeGESD, es decir, la etapa de verificación del modelo diseñado.

El lenguaje LeGESD se basa en la visión de que un sistema distribuido consiste en dos niveles de componentes: (1) componentes de comunicación, llamados *medios*, los cuales utilizan un sistema finito de recursos comunicados entre sí para la ejecución de la comunicación del sistema, y que se sincronizan el uno con el otro; (2) componentes de comportamiento, llamados *trabajos*, los cuales utilizan un sistema finito de estados seriales para la ejecución del comportamiento. La descripción de un trabajo es representada por *diagramas de comportamiento*, los cuales usan un conjunto de elementos gráficos. Existen dos conjuntos de elementos gráficos principales: *estados* (figura 1a, 1b y 1d) y *enlaces* (figura 1c) tanto para trabajos como para medios. La transición de un estado asume tomar un conjunto de pre-condiciones, y generar un conjunto de post-condiciones una vez que la ejecución de un estado concluya. La transición de un estado está sujeta a la verificación de las pre-condiciones que utiliza. Por otra parte, la descripción del medio es representada por *diagramas de comunicación* los cuales también utilizan un conjunto de elementos gráficos. Este conjunto se divide en dos subconjuntos: *inicialización* y *ejecución*. El subconjunto de inicialización se refiere a actividades iniciales de la comunicación tales como apertura, establecimiento del enlace y comienzo. El subconjunto de ejecución se refiere al comportamiento interno de la comunicación. Estos subconjuntos utilizan los mismos elementos gráficos principales que los usados en un trabajo (estados y enlaces), pero definen otros símbolos para representar estados.

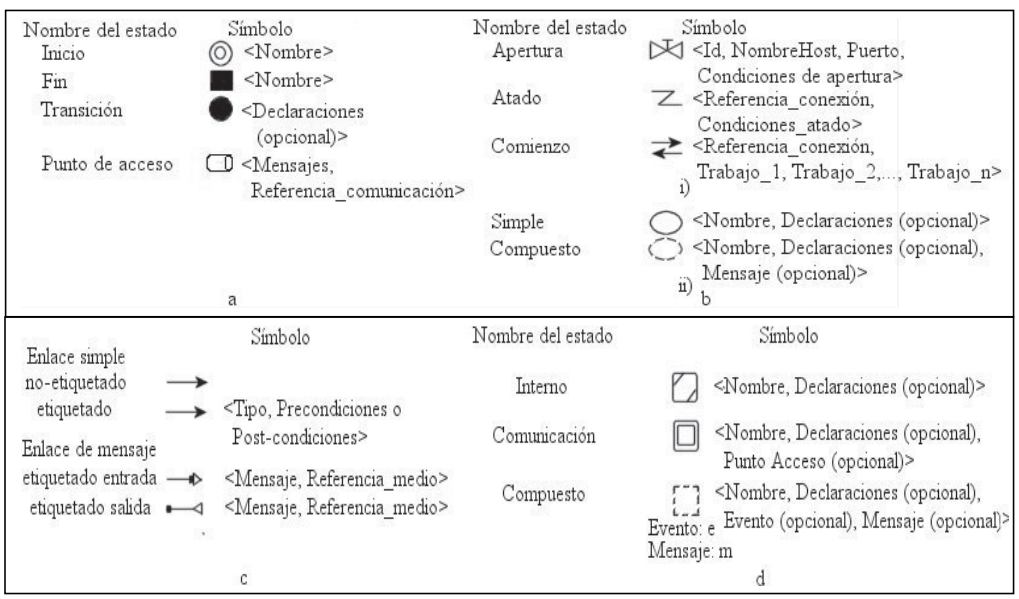

Figura 1 a. Conjunto de símbolos gráficos comunes a los estados de los trabajos y medios LeGESD, b. i) Conjunto de símbolos de inicialización y ii) conjunto de símbolos de ejecución de los estados de los medios, c. Conjunto de símbolos de los enlaces de los trabajos y los medios, d. Conjunto de símbolos de los estados de los trabajos LeGESD

LeGESD utiliza dos vistas principales para la especificación del nivel más alto:

1. Vista de sistema: Representa la estructura estática de un sistema distribuido que incluye los elementos estáticos del sistema y sus relaciones estáticas. Esta vista muestra los elementos globales del sistema en términos de los trabajos de más alto nivel y de los elementos de comunicación a utilizar entre ellos. Un trabajo representa un conjunto de elementos con estructuras similares (atributos) y comportamientos (operaciones). Puede existir una relación (asociación) entre dos o más trabajos. Una asociación indica el papel que un trabajo desempeña en la relación. La asociación conecta trabajos y es representada por los medios.

2. Vista de implantación: Describe las restricciones en la implantación del sistema (ambiente de ejecución seleccionado, lenguaje de programación usado, tecnología de comunicación) y la topología de la implantación (infraestructura). Esta vista es representada por anotaciones textuales en la vista del sistema. Un ejemplo de estas dos vistas se muestra en la figura 2, el ejemplo asume la especificación del problema clásico productor/consumidor.

La figura 2 muestra del lado derecho la Vista de sistema y del lado izquierdo la Vista de implantación. La Vista de sistema está conformada por el trabajo compuesto llamado "Prod/Cons", el cual es la representación global del sistema distribuido a especificar. El trabajo compuesto "Prod/Cons" incluye los elementos estáticos llamados "Prod" y "Cons", los cuales son trabajos compuestos. Se cuenta con tres trabajos compuestos "Cons" y uno llamado "Prod", lo que representa la existencia de un productor y tres posibles consumidores. Es posible que en un trabajo compuesto se tengan eventos y/o mensajes presentes que deban considerarse en la ejecución del trabajo (para el ejemplo no existen ni eventos ni mensajes). Finalmente, se tiene un trabajo *comunicación* llamado "ProdCons_ medio", que representa el medio por el cual los trabajos compuestos se comunicarán entre sí. Este trabajo representa la asociación que existirá entre el trabajo "Prod" y los tres trabajos "Cons".

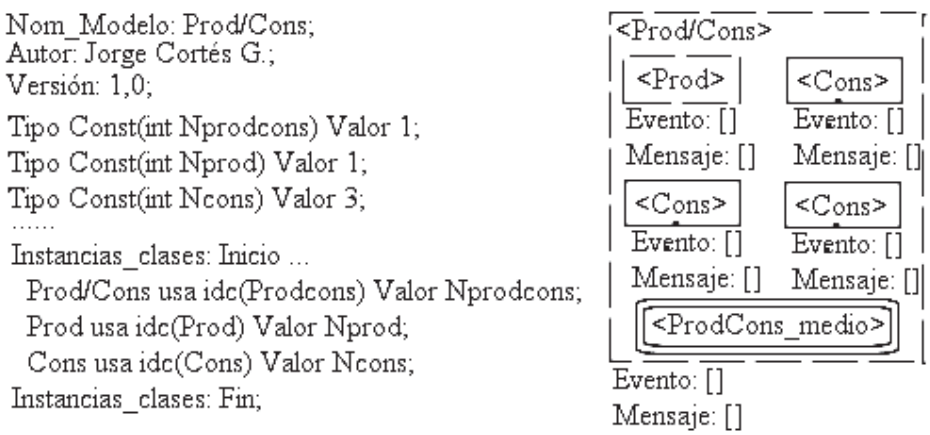

Figura 2 Vistas del sistema y de implantación de LeGESD

En la Vista de implantación para el ejemplo se muestran algunas partes de su declaración. Concretamente se tienen los datos generales de la especificación como son: nombre del modelo, nombre del diseñador y la versión. A continuación se declaran tres variables de tipo constante (usando la palabra reservada Tipo Const), cuyos nombres son Nprodcons, Nprod, Ncons que almacenarán los valores enteros 1, 1 y 3 respectivamente. Estos valores representan el número de instancias para

cada trabajo. Con la etiqueta Instancias_clases se declaran las instancias de cada uno de los trabajos definidos en la Vista de sistema, para ello se utiliza un identificador de clase (idc) el cual servirá para mantener la referencia de las clases que serán instanciadas. En esta declaración se usan los tipos de datos declarados previamente para indicar cuántas instancias de cada trabajo se generarán (1, 1 y 3).

Existen diversas secciones que pueden ser declaradas en esta vista, dependiendo de las necesidades del sistema a especificar. Ejemplos de estas secciones son: Lenguaje_prog, utilizada para declarar el lenguaje de programación en el que se generará automáticamente el código fuente de la especificación construida en LeGESD; Ambiente_ejec, utilizada para declarar características del ambiente de ejecución para el sistema especificado (tipos de sistemas operativos, tipos de procesadores, etc.).

Trabajos y medios LeGESD

Una vez que ha sido presentada la simbología del lenguaje y se ha entendido de manera informal lo que son trabajos y medios, se definirán formalmente estos elementos principales que forman al lenguaje gráfico de LeGESD, y que representan los elementos esenciales a utilizar durante la especificación del sistema distribuido.

Un trabajo LeGESD es una 7-tupla que contiene los siguientes elementos, *(S, i, L, Pr, Po, M, R)*, donde *Pr* es un conjunto de pre-condiciones, *Po* es uno de post-condiciones, *M* es uno de mensajes, y *R* es uno de referencias de medios. Por otra parte, *S* y *L* definen un grafo dirigido cuyo nodo inicial está en $i \subseteq S$. El conjunto de enlaces etiquetados $L \subseteq (S \times 1 \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (N \times Pr) \cup (N \times Po) \cup \{M \times R\}$, donde ε denota a la etiqueta vacía.

Debido a que los trabajos LeGESD son jerárquicos, el conjunto de estados *S* está definido a través de una *función jerárquica* Ψ, la cual se define por cada estado contenido en el conjunto de trabajos LeGESD. Esto es, $\Psi(s)=\{J_1,...,J_k\}$ si los trabajos LeGESD $J_1,\ldots,J_k$ están contenidos dentro del estado *s*. Dada la función jerárquica Ψ: un estado s es del tipo de uno de los estados posibles siguientes: inicio, interno, comunicación, punto de acceso, final, o transición, si $\Psi(s)=0$; un estado s es un estado compuesto si $\Psi(s)=\{J_1,...,J_k\}$.

Similarmente, el conjunto de referencias de medios *R* está definido por una *función compuesta* Λ, la cual se define (para cada medio) por los dos tipos de medios LeGESD que contiene. Esto es, $\Lambda(r)=\{I_m, E_m\}$ si los *medios LeGESD de inicialización* I_m y los *medios LeGESD de ejecución* E_m están contenidos en el medio *r*.

Un medio de inicialización I_m es una 6-tupla *(S, b, L, Pr, Po, No)* donde *Pr* es un conjunto de pre-condiciones, *Po* es uno de post-condiciones, y *No* es uno de nombres de medios. Por otra parte, *S y L* definen un grafo dirigido cuyo nodo inicial está en $b \subseteq S$. El conjunto de enlaces etiquetados $L \subseteq (S \times 1 \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (N \times Pr) \cup (N \times Po) \cup \{No\}$ donde ε denota una etiqueta vacía. El conjunto de estados *S* está definido por alguno de los siguientes estados posibles: inicio, apertura, atado, comienzo, final, punto de acceso, o transición. Un medio de ejecución E_m está definido de manera similar a un trabajo LeGESD. E_m es una 7-tupla *(S, b, L, Pr, Po, M, No)* donde *Pr* es un conjunto de pre-condiciones, *Po* es uno de post-condiciones, *M* es uno de mensajes, y *No* es uno de nombres de medios. *S y L* definen un grafo dirigido cuyo nodo inicial está en $b \subseteq S$. El conjunto de enlaces etiquetados $L \subseteq (S \times 1 \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (N \times Pr) \cup (N \times Po) \cup \{M \times No\}$ donde ε denota a la etiqueta vacía.

Debido a que E_m es jerárquico, el conjunto de estados *S* está definido a través de una *función jerárquica* Ω, la cual se define (para cada estado) por el conjunto de medios LeGESD de ejecución que contiene. Esto es, $\Omega(s)=\{E_{m1},...,E_{mk}\}$ si los medios LeGESD de ejecución $E_{m1},...,E_{mk}$ están contenidos dentro del estado *s*. Dada la función jerárquica Ω: un estado s es uno de los siguientes estados posibles: inicio, simple, punto de acceso,

final, o transición, si $\Omega(s)=0$; un estado s es un estado compuesto si $\Omega(s)=\{E_{m1},...,E_{mk}\}$.

LeGESD cuenta con una semántica clara y precisa del lenguaje, esta semántica será descrita a continuación, y se ha denominado como *Análisis y Diseño de Sistemas Distribuidos*.

Semántica de LeGESD: análisis y diseño de sistemas distribuidos

La semántica formal que sustenta al lenguaje de LeGESD se ha nombrado como Análisis y Diseño de Sistemas Distribuidos (ADSD). ADSD es un álgebra de procesos basada en CCS [15] que permite construir ecuaciones algebraicas a partir de la especificación gráfica realizada con el lenguaje LeGESD de un sistema dado. Partiendo de este grupo de ecuaciones algebraicas es posible realizar la verificación de la especificación así como la generación automática de código fuente de dicha especificación. A continuación se describen los detalles de la semántica formal de LeGESD, además se mostrarán algunas relaciones de equivalencia entre LeGESD y ADSD. Estas relaciones son importantes porque permiten transformar especificaciones gráficas en lenguaje LeGESD a expresiones en álgebra de procesos ADSD.

Semántica formal

La semántica de LeGESD está definida en dos fases. La primera consiste en identificar *elementos LeGESD-básicos*. Un elemento LeGESD-básico se define como un subconjunto de trabajos o medios LeGESD con una relación de equivalencia directa con ADSD, es decir, son elementos de LeGESD los cuales tienen una transformación a ADSD de acuerdo a la figura 3. La segunda fase consiste en definir un conjunto de transformaciones de elementos LeGESD-básicos a elementos de LeGESD, y viceversa. De esta manera, cada trabajo o medio LeGESD válido tiene una relación de equivalencia con un proceso ADSD. Esto se logra mediante una conversión inicial a un trabajo o medio LeGESD-básico, y al relacionar el trabajo o medio LeGESD-básico a su proceso ADSD equivalente. ADSD tiene como fundamento el álgebra de procesos CCS [17] con acciones de tiempo discreto. ADSD tiene un tipo de acción: acciones de ejecución, conocidas como *eventos*. Un evento en ADSD consiste de un elemento *(e)* donde *e* es una etiqueta. Permitiendo a la gramática *P* variar en el dominio de términos, *e* en el dominio de eventos etiquetados o de eventos internos (*t)*, *F* en un conjunto de eventos etiquetados, y *X* en el dominio de variables de término, la sintaxis de ADSD está dada por la gramática siguiente:

$$P ::= NIL \mid P_1 \overset{\alpha}{\rightarrow}{}^{\beta} P_2 \mid (e).P \mid P_1+P_2 \mid P_1||P_2 \mid P\Delta^b(P_1) \mid P/F \mid rec\ X.P \mid X$$

La semántica de procesos ADSD está definida en términos de un sistema de etiquetas de transición. *NIL* es un proceso que no ejecuta ninguna acción, por ejemplo, puede estar inicialmente bloqueado. El operador transición $\rightarrow$ $(P_1 \overset{\alpha}{\rightarrow}{}^{\beta} P_2)$ denota una transición del proceso P_1 al proceso P_2 bajo una acción α o β que P_1 debe ejecutar para llegar P_2. La transición puede darse sin una acción. *(e).P* ejecuta el evento *e* y procede a *P*, este operador corresponde al tipo de acción con la que trabaja ADSD (eventos). El operador elección + (P_1+P_2) representa dos posibilidades, alguno de los dos procesos P_1 ó P_2 puede ser elegido para ejecutarse. Esta elección depende del evento presente y de las limitantes en los recursos del entorno. El operador paralelo || $(P_1||P_2)$ representa la ejecución concurrente o paralela de P_1 y P_2. El constructor alcance Δ^b $(P\Delta^b(P_1))$ enlaza al proceso *P* mediante un evento de alcance. El constructor alcance puede terminar únicamente si *P* concluye exitosamente mediante la ejecución de un evento etiquetado como *b* (b puede ser cualquier etiqueta diferente a *t*), entonces el control procede al "manejador de éxito" P_1. El operador restricción / *(P/F)*, limita el comportamiento de *P*: eventos con etiquetas en *F* son permitidos para ejecutarse solamente si estos sincronizan y forman parte del evento interno *t*. El operador *rec (rec X.P)* representa recursión, permitiendo especificar comportamientos infinitos. El término X, sin algún enlace a *"rec"*, es una variable libre (variable externa e independiente al operador "rec") que pertenece a un conjunto infinito de variables de término libres.

Relaciones de equivalencia de LeGESD a ADSD

Las relaciones de equivalencia de LeGESD a ADSD definen la semántica de un subconjunto de elementos LeGESD válidos, conocidos como *LeGESD-básico*. LeGESD-básico tiene una correspondencia con ADSD, es decir, existe una relación de equivalencia entre LeGESD-básico hacia ADSD, y viceversa. Para definir la semántica de LeGESD se relacionan LeGESD-básico con LeGESD a través de un conjunto de transformaciones gráficas que sustentan a ADSD. Las transformaciones gráficas básicamente eliminan o adicionan enlaces y estados para transformar un trabajo LeGESD en un trabajo LeGESD-básico, o viceversa. Estas transformaciones permiten minimizar una especificación en LeGESD mediante la eliminación de estados y enlaces innecesarios, generándose una especificación más simplificada, equivalente a la original.

Un *trabajo LeGESD-básico* es un trabajo LeGESD válido tal que: 1) Cualquier estado transición tiene un enlace de salida simple etiquetado/no etiquetado, o dos enlaces de salida simples etiquetados; 2) cualquier estado compuesto es un estado de paro; 3) cualquier estado compuesto s, tiene uno o dos componentes anidados, es decir, p(s)={G} ó p(s)={G1, G2}; y 4) cualquier estado compuesto con enlaces de salida tiene un componente anidado y un enlace simple no etiquetado.

Cada trabajo LeGESD-básico corresponde a un proceso ADSD. La relación de equivalencia de un trabajo LeGESD consiste en obtener las equivalencias de los estados y los enlaces. La transformación inicia a partir del estado inicial y es recursivamente aplicada a los trabajos alcanzables desde este estado. Las transformaciones son combinadas utilizando un operador ADSD. La figura 3 muestra gráficamente las ocho transformaciones posibles de un trabajo LeGESD a un proceso ADSD, donde *T* representa una transformación desde la especificación LeGESD hasta los procesos ADSD. Estas transformaciones son resumidas a continuación y se muestran gráficamente en la figura 3. 1) Enlaza la transformación del trabajo LeGESD a la variable de proceso llamada P. 2) Un proceso de evento es creado a partir del evento que etiqueta el enlace de salida simple del estado transición. 3) Cada trabajo que inicia en el estado objetivo de un enlace no etiquetado es transformado y el resultado es combinado mediante el operador elección. 4) El estado final se transforma al proceso Nil. 5) El estado compuesto se transforma a una variable de proceso con el nombre de referencia. 6) El trabajo LeGESD dentro del estado compuesto es transformado, y los atributos de Evento y Mensaje del estado compuesto son usados en el operador restricción, los operadores son ignorados si ellos no han sido iniciados. 7) Los dos trabajos LeGESD dentro del estado compuesto se transforman y combinan a través del operador paralelo. Los atributos de Evento y Mensaje del estado compuesto son usados en el operador restricción. Los operadores son ignorados si no se inician. 8) El trabajo LeGESD dentro del estado compuesto se transforma y usa como el proceso principal en el operador alcance. La transformación del trabajo LeGESD que inicia en el estado objetivo del enlace no etiquetado genera la interrupción del proceso. Se ha desarrollado una herramienta computacional que genera especificaciones en LeGESD, y que automáticamente las convierte a los elementos LeGESD-básicos correspondientes.

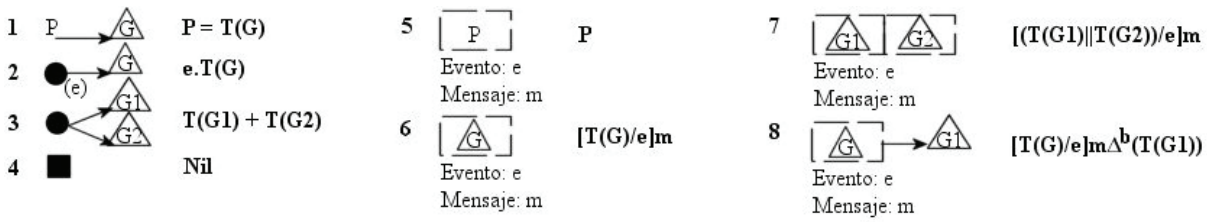

Figura 3 Transformaciones de trabajos LeGESD básicos a procesos ADSD

Ejemplo sencillo de transformación y verificación

Enseguida se muestra un ejemplo sencillo de transformación (realizada automáticamente) de LeGESD a ADSD y un análisis sobre su ejecución correcta ("correctness"). El ejemplo es básico pero útil para mostrar las bases de la transformación y la aplicación de ADSD en la verificación de la especificación. El sistema distribuido a especificar se mostró en la figura 2, se relaciona con el problema clásico productor/consumidor implementado de manera distribuida, y presenta tanto la vista de sistema como la de implantación de LeGESD. Diversos estados compuestos en la vista del sistema se representan con sus respectivos trabajos LeGESD y el estado de comunicación con sus respectivos medios LeGESD si el punto de acceso correspondiente existe. También se puede observar que el estado compuesto global Prod/Cons no es un trabajo LeGESD-básico, pero dentro de este estado compuesto global existen trabajos LeGESD-básicos (Prod y Cons) según la condición número tres de trabajos LeGESD-básicos definida en las relaciones de equivalencia de LeGESD a ADSD. Así, cada trabajo LeGESD-básico corresponde a un proceso ADSD.

Las transformaciones se presentan para cada trabajo LeGESD-básico dentro del estado compuesto Prod/Cons en la figura 4. La figura 5 muestra la versión del trabajo LeGESD para el estado compuesto Cons, conteniendo el diagrama de comportamiento de éste, el cual presenta el flujo de ejecución del consumidor. Los estados comunicación Consumir y Esperar son enlazados mediante estados transición, enlaces etiquetados y no etiquetados simples (que conectan a los estados del trabajo), y se enlazan al punto de acceso ProdCons_medio mediante enlaces de mensaje etiquetados de recepción y envío para realizar el acceso al medio, interactuando de esta manera con el productor. Para cada estado comunicación se especifican las etiquetas necesarias de cada enlace, con los atributos correspondientes a las pre-condiciones y las post-condiciones generadas en las transiciones de estado. Para los enlaces de mensaje etiquetados de recepción y envío se especifican los mensajes a intercambiar con el medio a través de cada punto de acceso.

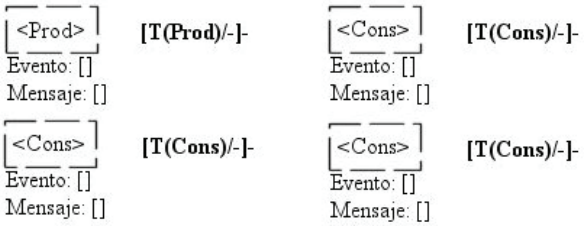

Figura 4 Transformaciones de trabajos LeGESD básicos a procesos ADSD

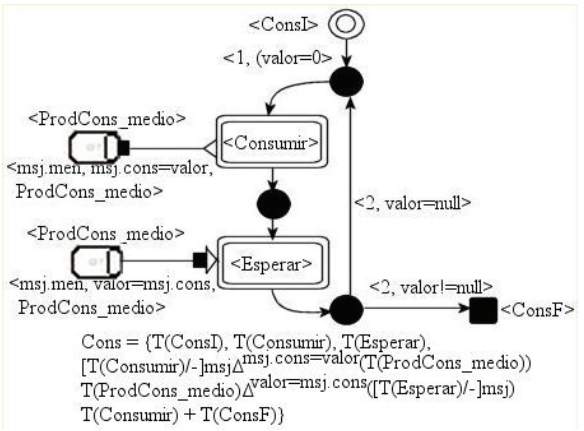

Figura 5 Diagrama de comportamiento para el estado compuesto Cons

El diagrama de comportamiento de la Figura 5 muestra trabajos LeGESD-básicos de acuerdo a las condiciones uno y tres definidas en las relaciones de equivalencia de LeGESD a ADSD. Estos trabajos LeGESD-básicos corresponden a procesos ADSD y sus transformaciones se muestran en la parte baja de la figura 5. La expresión Cons contiene los procesos ADSD resultantes de la transformación. Esta expresión está en orden directo según las distintas transiciones del diagrama de comportamiento. Dentro de la expresión Cons, la cuarta y quinta transformación utilizan los operadores restricción y alcance para las transformaciones de medios LeGESD a procesos ADSD, donde msj representa el mensaje enviado/recibido hacia/desde el medio. Además, la última transformación de la expresión Cons utiliza el operador elección.

Partiendo de la expresión Cons se puede realizar una prueba de ejecución correcta ("correctness") de la especificación gráfica en LeGESD transformada a procesos ADSD. Para ello se construye el conjunto de ecuaciones algebraicas de transición (ecuación 1) correspondientes al diagrama:

$$
\begin{aligned}
&Tr_1 := T(ConsI)^{valor} \rightarrow T(Consumir)\\
&Tr_2 := T(Consumir) \rightarrow T(Esperar)\\
&Tr_3 := [T(Consumir)/\text{-}]msj\Delta^{msj.cons=valor}(T(ProdCons\_medio))\\
&Tr_4 := T(ProdCons\_medio)\Delta^{valor=msj.cons}([T(Esperar)/\text{-}]msj)\\
&Tr_5 := T(Esperar) \rightarrow^{valor} T(Consumir)+T(ConsF)
\end{aligned}
\qquad (1)
$$

$$Tr_{51} := T(Esperar) \rightarrow^{valor=NULL} Tr_2 \qquad (A)$$

$$Tr_{52} := T(Esperar) \rightarrow^{valor!=NULL} T(ConsF) \qquad (B)$$

La ejecución correcta de estas ecuaciones es aquella que permite al estado compuesto Cons obtener un valor distinto de nulo desde el productor, a través de los estados de comunicación Consumir y Esperar. Analizando este grupo de ecuaciones de transición se puede comprobar que la transición que se alcanza en (1B) representa la ejecución correcta de la especificación. Por otra parte, la transición que se alcanza en (1A) representa un ciclo, con lo cual se puede generar un posible bloqueo. Este análisis permite comprobar que ADSD puede utilizarse para la verificación formal de la especificación construida con el lenguaje de LeGESD.

Trabajos relacionados

Entre los trabajos relacionados con LeGESD es posible destacar a IOA [8], Macedon [7] y PEDS [10]. IOA está basado en la especificación de un sistema distribuido a través de un autómata de entrada/salida. Esta especificación es completamente textual a diferencia de LeGESD, donde la especificación es gráfica y proporciona un mecanismo más intuitivo, influyendo directamente en el tiempo de especificación. Macedon y PEDS están basados en una especificación gráfica: Macedon está orientado para especificar comunicación en redes a través de diagramas de estados, mientras que LeGESD no sólo considera la especificación de aspectos de la comunicación del sistema distribuido, también considera los aspectos de especificación del comportamiento interno del sistema distribuido. Por su parte PEDS especifica sistemas distribuidos enfocando su especificación desde los aspectos lógicos hasta los aspectos físicos del sistema, lo cual hace compleja y extensa la especificación de cualquier sistema. LeGESD no considera los aspectos físicos del sistema como lo hace PEDS, en cambio concentra su especificación en los aspectos fundamentales de un sistema distribuido: su funcionamiento interno y sus funciones de comunicación, dejando los aspectos de ubicación física fuera de la especificación para mantener dicha especificación lo menos compleja y extensa posible. Adicionalmente, LeGESD proporciona, con respecto a los tres trabajos relacionados, una especificación modular,

jerárquica y escalable que es adecuada precisamente a algunas de las características intrínsecas que tienen los sistemas distribuidos como son escalabilidad, disponibilidad y confiabilidad.

Conclusiones

En este artículo se ha presentado el marco de trabajo LeGESD, que está integrado por un lenguaje gráfico de especificación y su semántica basada en álgebra de procesos. El lenguaje LeGESD se compone de una simbología y diagramas que permiten la especificación de sistemas distribuidos de manera modular, jerárquica y escalable. El lenguaje considera tanto los requerimientos funcionales como de comunicación del sistema, haciéndolo idóneo para la especificación de cualquier sistema concurrente o distribuido. La semántica que sustenta al lenguaje LeGESD se ha denominado como ADSD, la cual utiliza el álgebra de procesos para proporcionar una semántica precisa que describe la interacción entre los trabajos y medios LeGESD. Basado en el estudio presentado en este artículo, encontramos que ADSD tiene capacidades útiles para la verificación de sistemas distribuidos especificados con LeGESD como son: uso del álgebra de procesos, modularidad y una estructura jerárquica en la construcción de la expresión ADSD. Con ADSD basada en un álgebra de procesos se tiene una semántica de procesos algebraica precisa, la cual puede apoyar al análisis mediante la verificación formal de la especificación de sistemas complejos, apoyándose en las características de modularidad y jerarquía de LeGESD, y de la integración de los requerimientos de comunicación y de procesamiento de éstos. Finalmente con LeGESD, los diseñadores pueden especificar y verificar un sistema a través de su crecimiento incremental de componentes.

Reconocimientos

Los autores agradecen al Instituto Politécnico Nacional su apoyo en la elaboración de este trabajo.

Referencias

1. L. Arief, M. Little, S. Shrivastava, N. Speirs, S. Wheater. “Specifying distributed system services”. *BT Technical Journal. (Special Issue)*. Vol. 3. 1999. pp. 120-128.

2. N. Lynch, M. Tuttle. “An introduction to input/output automata”. *CWI-Quarterly*. Vol. 3. 1989. pp. 219-246.

3. J. Cortes, F. Menchaca. *Graphical Specification Language for Distributed Systems*. Proceeding IEEE on 15th International Conference on Computing. México D.F. (México). 2006. pp. 120-126.

4. I. Kinchin, D. Hay. “How a qualitative approach to concept map analysis can be used to aid learning by illustrating patterns of conceptual development”. *Educational Research*. Vol. 42. 2000. pp. 43-57.

5. J. Rumbaugh, I. Jacobson, G. Booch. “The unified modeling language reference manual”. *Object Technology Series*. Vol. 1. 1998. pp. 13-62.

6. D. Luckham. “Specification and analysis of system architecture”. *IEEE Transactions on Software Engineering.* Vol. 21. 1995. pp. 336-355.

7. A. Rodriguez, C. Killian. “MACEDON: Methodology for automatically creating, evaluating, and designing overlay networks”. *Proceedings of the NSDI*. Vol. 1. 2004. pp. 20-34.

8. S. Garland, N. Lynch. “The IOA language and toolset: Support for designing, analyzing, and building distributed systems”. *MIT Press Technical Report.* Vol. 1. 1998. pp. 1-42.

9. D. Regep, F. Kordon. *LfP: A specification language for rapid prototyping of concurrent systems*. Proceedings of the 12th International IEEE Workshop on Rapid System Prototyping. Monterey (Estados Unidos de America). 2001. pp. 90-97.

10. D. Zhang, K. Zhang. *A Visual Programming Environment for Distributed Systems*. Proceedings of the 11th International IEEE Symposium on Visual Languages. Washington (Estados Unidos de America). 1995. pp. 310-317.

11. J. Cortes, F. Menchaca. *Algebra de Procesos Aplicada a la Especificación Formal de Sistemas Distribuidos*. 1er Congreso Internacional en Sistemas Computacionales y Electrónicos. México D.F. (México). 2006. pp. 30-38.

12. H. Hermanns, U. Herzog. “Process algebra for performance evaluation”. *Theoretical Computer Science*. Vol. 274. 2002. pp. 43-87.

13. M. Bravetti, M. Bernardo. *Compositional asymmetric cooperations for process algebras with probabilities, priorities and time*. 1st International Workshop on Models for Time Critical Systems. Pennsylvania (Estados Unidos de America). 2000. pp. 3-16.

14. K. Honda, K. Tokoro. "On Asynchronous Communication Semantics". *Object-Based Concurrent Computing*. Vol. 612. 1992. pp. 21-51.

15. R. Milner. *A calculus of communication systems*. Ed. Springer Verlag. Nueva York (Estados Unidos de América). 1980. pp. 126-157.