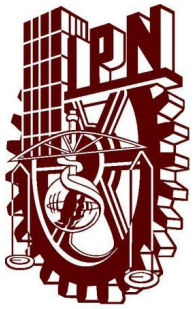


INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Lenguaje Gráfico de Especificación de Sistemas Distribuidos

TESIS

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

M. en C. Jorge Cortés Galicia

Directores de tesis:
Dr. Rolando Menchaca Méndez
Dr. Felipe Rolando Menchaca García

México D.F. Septiembre 2014

SIP-14 BIS

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 16:00 horas del día 28 del mes de Abril de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Lenguaje gráfico de especificación de sistemas distribuidos"

Presentada por el alumno:

CORTÉS	**GALICIA**	**JORGE**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	0	1	1	3	9	8

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Rolando Menchaca Méndez

Dr. Felipe Rolando Menchaca García

Dr. Jesús Alberto Martínez Castro

Dr. Francisco Hiram Calvo Castro

Dr. Rolando Quintero Téllez

Dr. José Giovanni Guzmán Lugo

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D.F. el día 23 del mes Mayo del año 2014, el (la) que suscribe Jorge Cortés Galicia alumno (a) del Programa de Doctorado en Ciencias de la Computación con número de registro B011398, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Rolando Menchaca Méndez y Dr. Felipe Rolando Menchaca García y cede los derechos del trabajo intitulado Lenguaje Gráfico de Especificación de Sistemas Distribuidos, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección jcortesg@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Jorge Cortés Galicia

Nombre y firma

Resumen

El diseño, especificación, validación y construcción de sistemas distribuidos representa un reto de complejidad elevada tanto desde el punto de vista matemático como de ingeniería. Aunando a esto, el impacto que tienen estos sistemas, tanto en la industria como en la vida diaria, los hace una de las líneas de investigación y desarrollo más activas e interesantes dentro de las ciencias de la computación.

En la presente tesis, se desarrolla un lenguaje gráfico de especificación, un álgebra de procesos, un algoritmo de transformación y una herramienta matemática que en conjunto forman un marco de trabajo para la especificación y validación de sistemas distribuidos. El lenguaje de especificación, llamado Lenguaje Gráfico de Especificación de Sistemas Distribuido (LeGESD), está basado en una sintaxis gráfica, la cual permite construir elementos sintácticos gráficos mediante el seguimiento de reglas sintácticas claramente establecidas. El algoritmo de transformación está sustentado por un conjunto de teoremas de transformación que permiten transformar la especificación de un sistema distribuido realizada en el lenguaje gráfico a una especificación equivalente en un álgebra de procesos llamada Análisis y Diseño de Sistemas Distribuidos (ADSD).

La herramienta matemática está compuesta del teorema de equivalencia gráfico-algebraica y de una serie de teoremas que establecen la equivalencia entre la especificación gráfica y la especificación algebraica obtenida a partir de ejecutar el algoritmo de transformación sobre dicha especificación gráfica. El resultado principal de la herramienta es establecer que las validaciones de propiedades de seguridad demostradas en la especificación algebraica son igualmente válidas para la especificación gráfica, y viceversa.

Finalmente, se desarrollan dos ejemplos que muestran la utilización tanto del lenguaje como de su semántica asociada, una vez que esta última es obtenida a partir de la ejecución del algoritmo de transformación.

Abstract

The design, specification, validation and construction of distributed systems are highly complex task that pose a set of challenges from the mathematical and engineering perspectives. Additionally, the impact of these systems both in industry and in everyday life makes them one of the most active and interesting research topics in computer science.

In this thesis we present a graphic specification language, a process algebra, a transformation algorithm and a mathematical tool that compose a framework for the specification and verification of distributed systems. The specification language is called ''Lenguaje Gráfico de Especificación de Sistemas Distribuidos'' (LeGESD) and is based on a graphical syntax which allows the construction of graphical syntactic elements using well established syntactic rules. The transformation algorithm is supported by a set of transformation theorems that allow to convert graphical specifications of a distributed system into an equivalent specification in a process algebra that we have called ''Análisis y Diseño de Sistemas Distribuidos'' (ADSD).

The mathematical tool consists of a graphical-algebraic equivalence theorem and a set of auxiliary theorems that establish the equivalence between the graphical specification and the algebraic specification obtained by executing the transformation algorithm over the graphical specification. This main result of the mathematical framework establishes that the verification of safety properties of the algebraic specification also apply to the graphical specification and vice versa.

Finally, we show two examples where both language and its semantics are used. This semantics is obtained from the transformation algorithm execution.

Agradecimientos

A mi madre, principal impulso y motivación para superarme tanto académica como personalmente, quien siempre ha estado a mi lado aconsejándome y guiándome para ser cada vez mejor estudiante y ser humano, siendo siempre un ejemplo de cómo conducirme con responsabilidad y ética. Gracias mamá por tu cariño y guía a lo largo de mi vida, sin tu cariño y ejemplo no sería el hombre que soy ahora.

A mis hermanas y hermano, por todo su apoyo y consejos dados, brindándome la certeza de que cuento con su cariño, respaldo y ejemplo. En especial a mi hermana Guadalupe, que desde donde se encuentra ahora sé que seguirá cuidándome como siempre lo hizo desde que era un niño, lo cual le agradezco infinitamente.

A mis asesores, Dr. Rolando Menchaca Méndez y Dr. Felipe Rolando Menchaca García, por todo su apoyo, enseñanzas, confianza, interés y asesoría a lo largo de estos años, gracias a los cuales he logrado concluir mis estudios doctorales.

Al Centro de Investigación en Computación, por brindarme un espacio propicio de aprendizaje e investigación que ha fomentado el desarrollo de mis conocimientos, habilidades y valores como investigador.

Al Instituto Politécnico Nacional, por apoyar mi formación como investigador durante mis estudios de doctorado, los cuales he realizado en esta apreciable casa de estudios.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

El campo de estudio de la presente tesis doctoral son los sistemas distribuidos, particularmente su especificación a través de un lenguaje gráfico que permita detallar su especificación y apoyar en su implementación. El objetivo del presente capítulo es especificar la meta que se propone alcanzar en la tesis, a partir de la identificación de la problemática existente en el campo de estudio, y presentando la propuesta de solución que se plantea ante el problema en cuestión.

1.1. 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 especificación, verificación 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 verificación automática. Una infraestructura de este tipo proporciona múltiples ventajas al desarrollo de sistemas distribuidos, debido a 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 dos primeras etapas son la base fundamental para la adecuada construcción de todo sistema distribuido, mientras que las dos últimas etapas son útiles tanto para la construcción automática de estos sistemas así como 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 propone al Lenguaje Gráfico de Especificación de Sistemas Distribuidos (LeGESD) [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 requiere menor tiempo 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 (Lenguaje de Modelado Unificado) [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 elaborada a través del lenguaje [6]. Existen algunas técnicas formales propuestas para la especificación de sistemas distribuidos, las cuales han usado notaciones textuales o gráficas, tal es el caso de Macedon [7], IOA [8], LfP [9], y PEDS [10]. Sin embargo, varias de ellas no cuentan con un formalismo adecuado que sustente a la técnica. 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 de sistemas distribuidos, combinando una técnica gráfica con una algebraica.

Para lograr esta combinación de técnicas, LeGESD propone un teorema de equivalencia gráfico-algebraica que permite validar la transformación entre la especificación gráfica en LeGESD a su especificación algebraica, y viceversa, de tal manera que dichas especificaciones sean equivalentes. Es decir, las afirmaciones que se realicen a partir de la especificación algebraica son válidas para la especificación gráfica, y viceversa.

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 el álgebra de procesos simplifica significativamente la extensión y complejidad de la especificación resultante y de su verificación correspondiente. Lo anterior se debe 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. Cualquier verificación que se realice mediante la especificación algebraica es igualmente válida para la especificación gráfica, debido al teorema propuesto.

El álgebra de procesos aplicada en LeGESD tiene dos motivaciones principales: la primera es porque proporciona un formalismo que permite la verificación de los sistemas especificados en LeGESD de forma consistente y bien definida; la segunda es porque proporciona una semántica formal operacional [12] a LeGESD. Esta semántica formal está definida por el Análisis y Diseño de Sistemas Distribuidos (ADSD), la cual es una semántica algebraica que cuenta con un algoritmo de transformación y un teorema de equivalencia entre la especificación gráfica en LeGESD y la especificación algebraica en ADSD. Este teorema de equivalencia es de gran utilidad para la verificación de propiedades de seguridad del sistema distribuido especificado.

1.2. Formulación del problema

En los sistemas distribuidos se presenta un reto importante, el cual radica en el grado de complejidad para construir sistemas de este tipo, por consiguiente el tiempo que lleva la construcción de estos sistemas es en muchas ocasiones elevado. La complejidad en su construcción involucra la participación de diversos diseñadores para dar forma al diseño del sistema en cuestión, involucrando diferentes puntos de vista y diferentes maneras de diseñar. Al no contar con un lenguaje de especificación que permita uniformizar criterios de diseño así como agilizar la construcción de los sistemas distribuidos, se tienen diseños diferentes sobre un mismo sistema que en muchas ocasiones propicia necesidades de reestructuración del diseño para generar un producto final funcional.

Si esta reestructuración tiene múltiples fases entonces repercutirá en el tiempo de construcción del sistema, lo cual es totalmente indeseable tratándose de sistemas que serán desarrollados en el ámbito industrial, donde se requiere una construcción e implantación con tiempos límites previamente establecidos y que deben de respetarse al máximo.

Por otra parte, en la construcción de sistemas de información (no sólo de los sistemas distribuidos) lo que se diseña frecuentemente no es lo que se implanta, teniéndose dos versiones diferentes del mismo sistema. Estas diferencias pueden ser pequeñas o grandes dependiendo de factores relacionados con el ambiente de ejecución, rompiéndose con la idea de: lo que se diseña es lo que se construye.

Lo anterior es consecuencia de no tener un lenguaje de especificación para sistemas distribuidos; aspectos como verificación formal de los modelos diseñados, así como posibles simulaciones de los mismos no son factibles de realizarse, provocando que los diseños realizados puedan mal funcionar en ciertas etapas y que solo serán visibles estas fallas hasta el momento de su ejecución. De presentarse estas fallas obligará a efectuar una reestructuración del diseño con los problemas que esto ocasionará y que han sido mencionados anteriormente.

Esto es debido a la complejidad elevada que involucra el diseñar, construir y verificar sistemas distribuidos, ya que es difícil visualizar a priori el espacio de posibles estados del sistema, las pruebas de ejecución no evidencian que los algoritmos distribuidos son correctos, y la demostración de las propiedades de un algoritmo distribuido es compleja debido al no determinismo inherente de los estos sistemas.

Un lenguaje gráfico como herramienta de especificación de sistemas distribuidos, frecuente-

mente carece de una semántica formal que lo sustente. Esta semántica debe fundamentar formalmente la forma en que transforma la especificación gráfica del lenguaje a la especificación textual que se utilice como semántica del lenguaje. Lo anterior hace necesario una herramienta matemática que permitan establecer la transformación y equivalencia entre la especificación gráfica del lenguaje a la especificación textual, y viceversa, de tal manera que dichas especificaciones sean equivalentes. De tal manera que las afirmaciones que se realicen a partir de la especificación textual sean consideradas como válidas para la especificación gráfica, y viceversa.

1.2.1. Hipótesis del trabajo

Presentado el contexto de la problemática en la construcción de los sistemas distribuidos, esta tesis plantea la siguiente hipótesis de trabajo:

Es posible definir un lenguaje gráfico para la especificación de sistemas distribuidos que incorpore como su semántica formal un álgebra de procesos, de tal manera que la especificación gráfica de un sistema distribuido sea equivalente a su especificación algebraica permitiendo la verificación de propiedades de seguridad de la especificación gráfica a partir de su especificación algebraica.

1.2.2. Necesidades existentes

La hipótesis anterior, así como la problemática planteada conlleva a la detección de una serie de necesidades en los sistemas distribuidos, las cuales son:

1. Contar con un marco de trabajo que permita la especificación de sistemas distribuidos.
2. El marco de trabajo debe definir un lenguaje de especificación, así como un conjunto de herramientas que permitan establecer la equivalencia entre la sintaxis y la semántica del lenguaje, la verificación de los sistemas especificados, la simulación de dichos sistemas, y la generación automática del código fuente de los sistemas.
3. El lenguaje de especificación debe ser lo suficientemente completo, es decir contar con una sintaxis y semántica, como para especificar sistemas distribuidos comunes.
4. La verificación de los sistemas debe efectuarse a través de la especificación efectuada con el lenguaje.
5. La verificación de los sistemas debe proporcionar resultados verídicos y posibles correcciones ante conflictos encontrados.
6. La función de simulación de los sistemas debe contemplar diversos ambientes de ejecución.
7. La función de simulación de los sistemas debe permitir la variación dinámica de variables y características que puedan llegar a afectar el desempeño del sistema simulado.
8. La integración de código fuente debe de efectuarse de forma automática (sin intervención del diseñador en la mayor medida posible).

9. La integración de código fuente debe apegarse a la filosofía de lo que se diseña es lo que se construye.

10. La integración de código fuente debe basarse en paradigmas actuales de programación y soporte de arquitecturas de software de comunicación para sistemas distribuidos.

1.2.3. Problemas específicos a resolver y propuesta de solución

Evidentemente el número de necesidades requeridas a resolver es grande. Esta tesis pretende solucionar un conjunto de ellas, las más importantes para establecer una base de investigación a partir de la cual nuevas investigaciones puedan proseguir, esto con la finalidad de fundar una línea de investigación relacionada con la construcción automática de sistemas distribuidos.

Específicamente los problemas que el presente trabajo intentará resolver son:

1. Definir un lenguaje gráfico de especificación de sistemas distribuidos que permita la especificación de este tipo de sistemas, considerando dos propiedades fundamentales que deberá tener este lenguaje:

   a) Debe permitir la especificación de elementos gráficos complejos, resultantes de la agrupación de diversos símbolos gráficos, a través de la definición de una sintaxis y sus reglas sintácticas asociadas, las cuales permitan la construcción de estructuras gráficas válidas.

   b) Debe tener una semántica formal, la cual refleje las relaciones entre los símbolos gráficos y su significado algebraico, de manera tal que posibilite a la especificación gráfica ser entendida independientemente de cualquier implementación que se realice de esta especificación [13].

2. El lenguaje gráfico de especificación debe ser lo suficientemente completo (contar tanto con una sintaxis y una semántica) como para especificar sistemas distribuidos comunes.

Para dar solución a estos problemas se propone la definición del Lenguaje Gráfico de Especificación de Sistemas Distribuidos (LeGESD), el cual es un marco de trabajo basado en un lenguaje gráfico utilizado para la especificación de sistemas distribuidos.

Concretamente la definición de LeGESD está conformada por dos etapas que incluyen la descripción de:

- Una notación gráfica (símbolos) que define aspectos relacionados con el comportamiento y la comunicación de sistemas distribuidos.

- Una semántica asociada a LeGESD dada por el Análisis y Diseño de Sistemas Distribuidos (ADSD), la cual está basada en una especificación algebraica (álgebra de procesos) con semántica operacional. Esta semántica tiene como apoyo una herramienta matemática consistente de un algoritmo para la transformación de la especificación gráfica a su especificación algebraica y de un teorema de equivalencia gráfico-algebraica que sustenta la equivalencia entre las especificaciones gráfica y algebraica.

LeGESD proporcionará características como son:

- Tener una notación gráfica tanto para el comportamiento interno como para la comunicación del sistema distribuido especificado.
- Tener una semántica formal del lenguaje basada en un álgebra de procesos.

LeGESD no es un punto de partida radicalmente diferente a lenguajes de modelado ampliamente conocidos y utilizados como UML, Booch, OMT, o OOSE utilizados en sistemas monolíticos, o como IOA y LfP utilizados en sistemas distribuidos y concurrentes.

LeGESD es una evolución natural como siguiente paso en el desarrollo de lenguajes de especificación, cuya propuesta principal está enfocada en la incorporación de un álgebra de procesos para propósitos de verificación de propiedades de seguridad en la especificación del sistema realizada con el lenguaje.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general que pretende cumplir esta tesis es:

- Desarrollar un lenguaje gráfico para la especificación de sistemas distribuidos que posea una semántica formal que permita demostrar propiedades de seguridad de los sistemas especificados.

1.3.2. Objetivos específicos

Como objetivos específicos se tienen los siguientes:

- Definir una notación gráfica (símbolos) para la especificación de sistemas distribuidos.
- Definir los elementos del lenguaje gráfico y sus reglas sintácticas para la especificación de sistemas distribuidos.
- Desarrollar una herramienta para validar la sintaxis del lenguaje a través de sus reglas sintácticas definidas para la construcción de estructuras sintácticas válidas.
- Definir una semántica algebraica para el lenguaje gráfico.
- Diseñar y verificar un algoritmo de transformación que tome como entrada una especificación gráfica en LeGESD y devuelva una especificación algebraica en ADSD.
- Desarrollar una herramienta matemática consistente de un teorema de equivalencia gráfico-algebraica para establecer la equivalencia entre la especificación gráfica y su especificación algebraica.
- Mostrar la aplicación del lenguaje propuesto y del algoritmo de transformación por medio de dos casos de estudio.
- Verificar las propiedades de seguridad de los dos casos de estudio.

1.4. Justificación

La construcción automática de sistemas computacionales especificados a partir de representaciones gráficas es actualmente usada de forma amplia en ambientes de ingeniería de software asistidos por computadora (CASE). Entre los lenguajes de modelado gráficos ampliamente utilizados en la ingeniería de software está UML [5], el cual permite modelar sistemas orientados a objetos con una serie de diagramas en los que se detalla el comportamiento de los sistemas a crear.

Sin embargo, gran parte de los lenguajes de especificación existentes para la creación de sistemas de software aún carecen de una construcción completamente automática de lo que se está especificando. Y, concretamente refiriéndose al área de los sistemas distribuidos, son pocos los lenguajes de especificación existentes que permitan una construcción automática de dichos sistemas. Esto presenta un espacio propicio y de interés para desarrollar propuestas que den alternativas para especificar y automatizar la construcción de sistemas distribuidos.

Una primera pregunta probable es el por qué tratar a los sistemas distribuidos como objeto de estudio, la respuesta es sencilla, es debido a su importancia actual en la computación. Hoy en día la tendencia en los sistemas de información es la descentralización y distribución de información en diferentes ubicaciones para su procesamiento, siguiendo una filosofía de divide y vencerás se pretende obtener un mejor aprovechamiento de los recursos de cómputo que se dispongan, y para proporcionar eficientemente ese soporte de procesamiento de información y aprovechamiento de los recursos es recomendable contar con un sistema distribuido.

Un sistema de información desarrollado como un sistema distribuido necesita generalmente satisfacer requisitos de distribución de trabajo [14]. El costo de una implantación manual de dicho sistema distribuido será frecuentemente alto. Por lo tanto, es deseable contar con la posibilidad de construir de una manera automática el sistema propuesto evitando ese alto costo. Para hacer esto, es importante evaluar los requisitos del nuevo sistema a desarrollar y traducirlos a una especificación (un diseño), y para realizar esta traducción es indispensable contar con un lenguaje que permita efectuar la especificación del diseño. El proceso del diseño [15] ayuda a los desarrolladores del sistema tanto a entender mejor los requisitos del mismo así como a evitar ideas falsas sobre el sistema.

A partir de una especificación, un programa generador de código fuente puede ser utilizado para implantar la especificación creada y obtener una aplicación ejecutable del sistema propuesto [16], [17]. La implantación rápida de estos sistemas proporciona todos los datos y resultados necesarios sobre el desempeño del sistema implantado, y a partir de estos datos y resultados, el funcionamiento del sistema se puede mejorar y probarse de inmediato en forma real, complementándose como una alternativa de evaluación con los sistemas básicos de simulación [18].

Una segunda pregunta probable es por qué sustentar la especificación formal del lenguaje en un álgebra de procesos, la respuesta se justifica en las características que posee el álgebra de procesos (explicadas anteriormente), estas características son convenientes para el sustento formal del lenguaje gráfico permitiendo la especificación de sistemas reactivos, incluyendo aquellos sistemas que se encuentran en el área distribuida [19].

El álgebra de procesos es una técnica para describir formalmente sistemas de cómputo reactivos, especialmente aquellos que cuentan con componentes de comunicación y ejecución concurrente [20]. El álgebra se enfoca a la especificación de sistemas y sus comportamientos, proporcionando un adecuado apoyo para la especificación, verificación, prueba y otras actividades del ciclo de vida del sistema [21], como puede ser la construcción automática de dicho sistema.

Siendo un aspecto de importancia y en crecimiento, el estudio en la construcción automática de sistemas distribuidos aporta una línea de investigación de interés que puede comenzar a cimentarse y desarrollarse con un futuro promisorio respecto a su posible impacto en los avances que pueda llegar a ofrecer a estos sistemas.

Y como en toda nueva construcción, es necesario proporcionar una base. La definición del Lenguaje Gráfico para la Especificación de Sistemas Distribuidos (LeGESD), pretende ser justamente esa base de la cual pueda partir la construcción automática de dichos sistemas.

1.5. Organización de la tesis

La presente tesis está dividida en nueve capítulos a través de los cuales se dará un panorama y solución a los problemas planteados.

El capítulo I ha abarcado la presentación del trabajo a realizar, se planteó la problemática del objeto de estudio así como la hipótesis de trabajo y la propuesta de solución que la tesis pretende proporcionar; se definió el objetivo general y los objetivos específicos a cumplir, y finalmente se explicó la justificación para la realización de esta tesis.

El capítulo II explicará los antecedentes que rodean a los lenguajes gráficos (visuales).

El capítulo III establecerá el marco conceptual sobre álgebra de procesos y sistemas distribuidos, describiendo los conceptos principales.

El capítulo IV explicará los trabajos relacionados con el objeto de estudio.

El capítulo V desarrollará la propuesta de la sintaxis gráfica para LeGESD.

El capítulo VI desarrollará la propuesta de la semántica basada en el álgebra de procesos ADSD para LeGESD.

El capítulo VII presentará los casos de estudio de la tesis.

El capítulo VIII presentará la herramienta computacional desarrollada como parte de la presente tesis.

Finalmente, el capitulo IX proporcionará las conclusiones y trabajos a futuro que se desprenden del presente trabajo.

2 Lenguajes visuales

Un formalismo para la descripción y análisis de manera jerárquica y composicional, relacionado con el comportamiento funcional de sistemas es el álgebra de procesos. Las características específicas de los sistemas distribuidos y el tipo de problemas que en ellos se plantean suponen un reto para la ingeniería de software, la cual ha tenido que replantear muchos de los métodos y paradigmas de programación existentes para los sistemas tradicionales. El objetivo del presente capítulo es proporcionar un panorama general de la teoría de los lenguajes visuales.

2.1. Introducción

Un área de investigación de reciente actividad se relaciona con el desarrollo de los lenguajes visuales. Estos lenguajes utilizan imágenes como símbolos, una técnica para la construcción de entornos orientados a lenguajes visuales es caracterizar un entorno genérico con un lenguaje de especificación. Los mecanismos para especificar lenguajes visuales pueden utilizarse como la base para el desarrollo de un entorno de programación con lenguajes visuales. Básicamente, cualquier lenguaje visual consiste en una gramática, la cual contiene la estructura sintáctica y la semántica del lenguaje visual.

Los lenguajes visuales utilizan símbolos gráficos para representar algún tipo de comportamiento computacional, además aprovechan el hecho de que una representación gráfica frecuentemente resulta ser más representativa que una representación textual sobre la misma información que se está representando. Por ejemplo, un autómata de estados finitos puede ser descrito tanto de forma textual como gráfica, sin embargo el diagrama de un autómata es mucho más fácil de comprender.

Los lenguajes visuales pueden aumentar la capacidad de los ingenieros para modelar y comprender los sistemas complejos. Sin embargo, para utilizar con eficacia los modelos visuales, la sintaxis y la semántica de los lenguajes deben definirse con precisión. Dado que los modelos visuales que actualmente se utilizan para especificar sistemas pueden ser descritos como tipo grafos, la gramática de grafos se ha identificado como un posible formalismo adecuado para describir la sintaxis abstracta de lenguajes visuales.

2.2. Lenguajes visuales

La gran mayoría de los esfuerzos en investigación sobre programación visual han sido orientados en el desarrollo de lenguajes visuales específicos. Por ejemplo, lenguajes visuales han sido desarrollados para programación funcional, diagramas de flujo basados en iconos, autómatas de estados finitos, flujo de datos, entre otros. La implementación de estos lenguajes ha requerido la construcción de ambientes con características particulares para la creación, manipulación y procesamiento de los aspectos visuales de los lenguajes.

Algunas de las razones por las cuales se ha seguido este enfoque en la implementación de los lenguajes visuales es la falta de modelos formales para la especificación de la sintaxis y semántica de los lenguajes visuales, así como la construcción de herramientas para la utilización de las especificaciones realizadas con los lenguajes. Para lenguajes textuales se cuenta con modelos formales tales como las gramáticas libres de contexto, las cuales proporcionan un mecanismo para la especificación sintáctica, sin embargo para los lenguajes visuales no se cuenta con modelos formales similares que sean considerados como estables para la especificación sintáctica gráfica.

El término lenguaje visual es utilizado para describir una amplia variedad de áreas de aplicación, por ejemplo se tienen lenguajes visuales para la manipulación de información, lenguajes para apoyar interacciones visuales, lenguajes para programación con expresiones visuales, entre otros. Los lenguajes visuales pueden ser clasificados [22] en alguno de los siguientes tipos:

- Lenguajes basados en iconos.
- Lenguajes basados en formas.
- Lenguajes basados en diagramas.

Estos tres tipos de lenguajes comparten la característica de que todos utilizan imágenes (símbolos gráficos) para crear las especificaciones de los programas a construir, La diferencia entre ellos radica en su sintaxis y en su semántica, es decir en el conjunto de símbolos gráficos que cada uno de los lenguajes utiliza y en el significado que cada uno le proporciona a las construcciones sintácticas que se forman. Un símbolo gráfico representa el elemento de nivel más bajo de un lenguaje visual, el cual puede ser visto como una colección de elementos primitivos como son líneas, polígonos y cadenas de texto, este símbolo gráfico es análogo a un elemento léxico de un lenguaje textual.

Los elementos primitivos de un símbolo gráfico se agrupan dentro de formas más complejas que dan origen propiamente al símbolo, un símbolo a su vez puede agruparse con otros símbolos en formas más complejas dando origen a elementos gráficos compuestos, y así sucesivamente. Este agrupamiento de símbolos gráficos y las construcciones gráficas que forman se considera la estructura sintáctica del lenguaje visual.

Los lenguajes visuales y los lenguajes textuales son similares en el hecho de que ambos forman sus estructuras sintácticas a través de un alfabeto básico, el cual puede llevar a la formación de estructuras bien definidas, sin embargo estos dos tipos de lenguajes difieren en dos aspectos importantes:

1. Los elementos sintácticos de los lenguajes textuales (cadenas textuales) son unidimensionales y por lo tanto tienen una naturaleza lineal en su composición, mientras que los elementos

sintácticos de los lenguajes visuales (símbolos gráficos) son objetos bidimensionales y no necesariamente tienen una naturaleza lineal en su composición.

2. La naturaleza lineal de los lenguajes textuales es reflejada en una estructura de construcción tipo árbol, mientras que la naturaleza no lineal de los lenguajes visuales es reflejada en una estructura donde un símbolo puede agruparse con otros para crear una forma compleja, y esta agrupación no necesariamente tiene que ser con algún símbolo gráfico que se encuentre en la vecindad a la derecha o izquierda del símbolo a agrupar. De esta manera la estructura de construcción en los lenguajes visuales es frecuentemente un grafo dirigido [23].

La especificación de la relación entre un símbolo gráfico de un lenguaje visual y su significado, por ejemplo la relación existente entre un diagrama de Venn y su significado en la teoría de conjuntos, puede ser definida mediante un lenguaje gráfico de especificación. Un lenguaje gráfico de especificación debe satisfacer los requerimientos siguientes:

- El lenguaje debe permitir la especificación de elementos gráficos complejos, resultantes de la agrupación de diversos símbolos gráficos, como una composición jerárquica de los diversos símbolos que componen a los elementos complejos. Debe permitir las relaciones topológicas, geométricas y semánticas entre los símbolos gráficos expresadas como restricciones de alto nivel.

- El lenguaje debe tener una semántica formal, la cual refleje las relaciones entre los símbolos gráficos y su significado, de manera tal que posibilite a la especificación gráfica, ser entendida independientemente de cualquier implementación que se realice de la especificación.

- El lenguaje debe tener una semántica operacional eficiente en su implementación tanto para la generación como para el reconocimiento de los símbolos gráficos del lenguaje visual.

La forma tradicional en los lenguajes visuales de especificar la composición jerárquica de los símbolos gráficos es utilizando reglas, las relaciones gráficas y semánticas entre los símbolos gráficos son especificados declarativamente a través de restricciones. Estas restricciones pueden ser combinadas en cláusulas para definir nuevas restricciones, y la lógica de primer orden aplicada a estas cláusulas proporcionan la fundamentación teórica. La teoría de modelos otorga una semántica declarativa, la cual determina el significado de la especificación de los símbolos gráficos, y la teoría de prueba otorga la semántica operacional, la cual determina cómo generar o reconocer símbolos gráficos a partir de la especificación [24].

Como se ha mencionado anteriormente, los lenguajes visuales se utilizan en una diversidad de ámbitos tradicionales de ingeniería, incluyendo diseño de circuitos electrónicos, ingeniería química y diseño arquitectónico. Los lenguajes visuales se han vuelto muy importantes en la ingeniería de software, como han demostrado las tendencias hacia lenguajes visuales de modelado (como el Lenguaje Unificado de Modelado UML), y entornos de programación visual. Estudios empíricos han demostrado que estos lenguajes visuales tienen la ventaja de ser más intuitivos y rápidos de comprender en comparación con los lenguajes textuales. Además, los patrones comunes de especificación son más fáciles de reconocer en las representaciones visuales.

Sin embargo, para utilizar eficazmente estos lenguajes visuales, es necesario definir claramente su sintaxis y semántica. Esto se debe a que los diagramas de un lenguaje visual a menudo se utilizan

para la comunicación entre las diferentes partes interesadas. Una semántica ambigua puede dar lugar a malentendidos y requerir un esfuerzo adicional para explicar el significado caso por caso. Una anotación de sintaxis poco específica también es difícil de aprender y aplicar.

Por otra parte, si la sintaxis y la semántica son formalmente especificadas, pueden proveerse herramientas de soporte, debido a que todas las herramientas de desarrollo deben estar basadas en definiciones inequívocas. Análogo a la sintaxis de lenguajes textuales, en la sintaxis de un lenguaje visual debe distinguirse una sintaxis concreta, que describe la notación de los símbolos gráficos y su representación gráfica en el lenguaje, y una sintaxis abstracta que define la representación interpretable para la computadora. La sintaxis concreta define al lenguaje en términos de objetos visuales como iconos o elementos gráficos. Los elementos de la sintaxis concreta se deben asignar a la estructura subyacente de la sintaxis abstracta, que es una representación de máquina interpretable.

Para definir la sintaxis abstracta de un lenguaje visual, dos enfoques básicos se utilizan actualmente. El primer enfoque, tal como se utiliza en el lenguaje de definición de UML, define un meta-modelo del lenguaje y restringe las instancias de este meta-modelo para las oraciones sintácticamente correctas y con significado, o diagramas de este lenguaje.

El segundo enfoque, más constructivo, utiliza una gramática del lenguaje de definición que describe cómo los símbolos se especifican en el meta-modelo y se utilizan para formar frases correctas. La definición de una gramática requiere mayor elaboración, sin embargo, hay varias ventajas en comparación con el primer enfoque. Las principales ventajas del enfoque basado en la gramática son la posibilidad de definir una semántica estructurada sobre la base de las reglas de la gramática y la capacidad de crear transformaciones dirigidas por la sintaxis.

La estructura subyacente de la mayoría de los lenguajes visuales puede ser definida por grafos dirigidos. En consecuencia, las gramáticas de grafos son un medio natural para especificar una sintaxis abstracta y para analizar los lenguajes visuales. Las gramáticas de grafos utilizan las reglas de transformación de grafos para generar conjuntos de gráficos válidos para el lenguaje visual.

Por consiguiente, de forma análoga a las gramáticas de cadena o basadas en árbol, una gramática de grafo especifica el conjunto de frases válidas visuales que se pueden derivar de un gráfico inicialmente con una secuencia finita de producciones gramaticales gráficas. Para definir la semántica de un lenguaje visual la sintaxis abstracta se debe asignar a un dominio semántico.

2.3. Lenguajes visuales y sus ámbitos

Los lenguajes visuales tienen múltiples ámbitos de aplicación en la computación. En este apartado se presentan tres de ellos:

- En la educación.
- En la calidad de modelos de software.
- En los sistemas móviles distribuidos.

En el ámbito educativo, el aumento de las tecnologías nuevas y avanzadas de aprendizaje, tales como ”e-learning”, aprendizaje móvil, juegos extremos y simulaciones, a menudo en combinación con la introducción de nuevos modelos de aprendizaje, tales como el aprendizaje basado en

problemas, en casos, el aprendizaje basados en competencias, entre otros, ha aumentado significativamente la complejidad de los procesos de enseñanza y aprendizaje. Esto requiere un diseño más avanzado de los procesos de desarrollo en los que la comunicación se apoya, haciendo uso de lenguajes visuales para el diseño de instrucción.

La introducción de la tecnología para el aprendizaje en la educación está provocando que el diseño de cursos y materiales de instrucción digitales sea una tarea cada vez más compleja. Los lenguajes visuales para el diseño de instrucción se consideran como herramientas conceptuales para lograr mayores estándares y, al mismo tiempo diseñar soluciones más creativas, así como mejorar la comunicación y transparencia en el proceso de diseño. Existen tres lenguajes visuales para el diseño de instrucción (E2ML, PoEML, coUML). Los tres tienen aspectos cognitivos, los cuales son de relevancia para el aprendizaje de un lenguaje visual, para la creación de modelos con él, y para la comprensión de estos modelos.

En respuesta, los lenguajes visuales para el diseño de instrucción (VIDLs) orientados a diseñadores y desarrolladores instruccionales, se están convirtiendo en una herramienta conceptual nueva para hacer frente a esta complejidad. Por ejemplo, dos manuales sobre lenguajes visuales para el diseño de instrucción [25], [26] y un capítulo sobre el mismo tema se han publicado recientemente en el Manual de Investigación AECT [27]. Sin embargo, hasta ahora existe una discrepancia entre la atención prestada a VIDLs en la investigación y el uso real que hacen los diseñadores instruccionales. En la práctica, los diseñadores instruccionales tienen dificultades para utilizar VIDLs debido a su desconocimiento y la complejidad intrínseca del lenguaje utilizado.

Por lo tanto, en la actualidad las concepciones acerca de la utilidad y la eficacia cognitiva de VIDLs son de relevancia principalmente práctica, en donde la finalidad es proporcionar una base sólida para la evaluación y comparación de VIDLs existentes, así como orientar a los profesionales en la elección de un lenguaje apropiado. La facilidad en el uso del lenguaje visual para el diseño de instrucción influye positivamente en la usabilidad percibida, y es probable que esta característica de los VIDLs tenga una buena influencia en el deseo del usuario a familiarizarse con un VIDL. La literatura existente que compara VIDLs se centra principalmente en los aspectos formales del lenguaje, y las evaluaciones desde el punto de vista del usuario son raras hasta ahora. Hay algunos estudios que evalúan la usabilidad de VIDLs específicos, pero poca investigación se ha llevado a cabo para la evaluación comparativa de VIDLs.

En comparación con lenguajes de modelado de propósito general, como el Lenguaje Unificado de Modelado (UML), los VIDLs son lenguajes de modelado de propósito específico, cuyo ámbito es el diseño de instrucción. El propósito de VIDL es proporcionar un modelo de información semántica, describiendo el contenido y proceso de una entidad de aprendizaje para apoyar en su reutilización e interoperabilidad.

En la calidad de modelos de software, garantizar la calidad de un modelo es un factor clave para el éxito en muchas áreas de informática, y se ha vuelto crucial en los últimos paradigmas de ingeniería de software, como es el caso que propone el desarrollo de software dirigido por modelos. Las herramientas de apoyo para obtener las medidas y los rediseños de los modelos, se han vuelto esenciales para ayudar a los desarrolladores a mejorar la calidad de sus modelos. Sin embargo, el desarrollo de estas herramientas de apoyo para la gran variedad de notaciones visuales utilizadas por los ingenieros de software, es una tarea difícil y repetitiva que no aprovecha los avances anteriormente realizados.

Un enfoque para la creación de estas herramientas de apoyo son los lenguajes visuales específicos de un dominio (DSVLs). Para este ámbito concretamente, el dominio se refiere a la medición

y rediseño de software [28]. El objetivo de estos lenguajes es facilitar la tarea de definir las mediciones y rediseños de cualquier software, así como la generación de herramientas utilizadas para tal especificación reduciendo o eliminando la necesidad de codificación. Las bases de diseño del lenguaje están sustentadas en el uso de patrones visuales para la especificación de los elementos pertinentes para cada tipo de medición y rediseño. Además, el lenguaje permite la especificación de cualquier rediseño o procedimiento, por medio de reglas de transformación gráficas aplicadas a los símbolos gráficos.

Los diseños se pueden activar cuando las mediciones alcanzan un cierto umbral, de esta manera, cuando un DSVL está diseñado, es posible especificar las mediciones y los rediseños que estarán disponibles en el entorno de modelado final, generado por el lenguaje.

Las notaciones mediante diagramas están inmersas en muchas actividades de desarrollo de software. Se utilizan en las fases de planificación, análisis y diseño como un medio para especificar, comprender y razonar sobre el sistema que se construirá. La notación utilizada abarca desde los lenguajes de uso general (como UML para el análisis y diseño de sistemas orientados a objetos) hasta otros DSVLs orientados a un dominio de aplicación particular.

Los DSVLs proporcionan un alto nivel de abstracción, que tiene el potencial de aumentar la productividad del usuario para la tarea de modelado específico y la calidad de los sistemas que hacen uso de ellos. En general, son menos propensos a errores y más fáciles de aprender que los lenguajes textuales de uso general, porque la diferencia semántica entre el modelo mental del usuario y el modelo DSVL es menor. Además, el uso de DSVLs es central a los paradigmas recientes para el desarrollo de software, tales como el desarrollo dirigido por modelos (MDD). En este paradigma, los modelos (a menudo expresados mediante un DSVL) son el objeto principal, del cual el código se genera automáticamente. Por esta razón, las técnicas para asegurar la calidad del modelo antes de la generación de código, son esenciales.

En los sistemas móviles distribuidos, la sensibilidad al contexto juega un papel importante en los sistemas móviles distribuidos, ya que permite la adaptación de los dispositivos móviles de los usuarios. Sin embargo, uno de los principales desafíos es la preservación de la privacidad de los usuarios. El Perfil de Modelado de Contexto (CMP), una ligera extensión de UML, es un lenguaje visual para los modelos de contexto en los sistemas móviles distribuidos. El modelo resultante materializa la meta-información del contexto, es decir, la fuente y validez de la información del contexto, y está sujeto a restricciones de privacidad. El perfil proporciona varias reglas bien establecidas para los modelos de contexto y apoya el desarrollo de aplicaciones móviles sensibles al contexto a través de un lenguaje visual de modelado adecuado.

Los dispositivos móviles actuales poseen alto poder computacional, interfaces de usuario avanzadas y capacidades de comunicación inalámbrica, pero aún tienen que lidiar con problemas de usabilidad [29]. Las limitaciones de la interfaz de las computadoras portátiles se traducen en el uso complicado de los dispositivos móviles, así como de las aplicaciones alojadas en ellos. La sensibilidad al contexto es un tema de investigación interesante en la informática móvil, ya que permite a las tecnologías disminuir el problema de las limitaciones de la interfaz y aprovechar el dispositivo móvil para ayudar de manera general al usuario. El descubrimiento de servicios en sistemas distribuidos, se puede mejorar haciendo coincidir las descripciones de los servicios enriquecidos con el contexto actual del usuario, lo que aumenta la idoneidad de los resultados de búsqueda de los servicios.

La sensibilidad al contexto es el fundamento del contexto basado en la provisión de servicios, que permite la adaptación de la prestación del servicio al cliente de manera individual. El uso de

información de contexto facilita el diseño de interfaces de usuario y disminuye las restricciones de la interfaz de dispositivos móviles. También permite el desarrollo de aplicaciones que se pueden adaptar a cada usuario. Estas aplicaciones pueden proporcionar una mayor personalización, un comportamiento dinámico y un uso más cómodo que resulta en un mejor soporte de usuario. Por lo tanto, se puede mejorar la capacidad general de los dispositivos móviles, considerándose como un enfoque para generar dispositivos inteligentes.

El contexto puede definirse como el conjunto de información sobre el propio usuario y su entorno. Típicamente, el contexto del usuario incluye la información que representan el nombre del usuario, su ubicación, su actividad actual y personas cercanas. Esta información puede ser incorporada por aplicaciones sensibles al contexto y facilita la adaptación de la aplicación con el contexto actual del usuario, resultando en un soporte mejorado y más adecuado del usuario. En el proceso de desarrollo de una aplicación sensible al contexto, el programador debe especificar la información de contexto que requiera la aplicación en detalle, de tal manera que la adaptación que pretende, pueda lograrse. Esto significa que la estructura del contexto, es decir, el conjunto de información de contexto requerida, las propiedades y las relaciones entre la información de contexto, tiene que ser definida. El desarrollador de aplicaciones tiene que construir un modelo de aplicación específico del contexto del usuario, lo que resulta en un proceso de desarrollo: el modelo de contexto.

Sin embargo, el contexto de un usuario no está compuesto por un conjunto duradero de información. En un sistema distribuido móvil, que está constituido por los dispositivos interconectados móviles de los usuarios del sistema, el contexto de usuario está cambiando irregularmente debido a la movilidad del usuario. La ubicación actual de una persona representa una información sobre el usuario que cambia con frecuencia. En contraste con esto, el lugar de nacimiento de un usuario representa un hecho estático que nunca se altera. La validez de la información de contexto representa la meta-información del contexto.

El conocimiento de esta meta-información es esencial para que las aplicaciones se basen en un marco de trabajo para aplicaciones sensibles al contexto. Los servicios que son proporcionados por el marco, es decir, el soporte de almacenamiento de información de contexto o el control de la validez del contexto, depende altamente de esta meta-información. Por lo tanto, el modelo de contexto no sólo debe proporcionar la definición de la estructura del contexto del usuario, sino que también debe reflejar la meta-información del contexto.

Los lenguajes visuales juegan un papel importante en la ingeniería de software porque las representaciones gráficas de los modelos complejos son más comprensibles y más aparentes a los seres humanos. El lenguaje visual más popular es UML. UML proporciona diferentes tipos de diagramas para visualizar distintos aspectos del sistema de software modelado. La separación obtenida en estos diagramas aumenta la modularidad del modelo visualizado.

Sin embargo UML no proporciona un soporte adecuado para el modelado de sistemas distribuidos en general, debido a que es un lenguaje visual de propósito general que omite las características propias de los sistemas distribuidos, es por ello que la presente tesis propone un lenguaje visual de dominio específico relacionado con los sistemas distribuidos.

Con esto se concluye la presentación de los aspectos teóricos más relevantes de los lenguajes visuales. En el siguiente capítulo, se revisarán los aspectos teóricos más importantes del álgebra de procesos y de los sistemas distribuidos.

3 Álgebra de procesos y sistemas distribuidos

Un formalismo para la descripción y análisis de manera jerárquica y composicional, relacionado con el comportamiento funcional de sistemas es el álgebra de procesos. Las características específicas de los sistemas distribuidos y el tipo de problemas que en ellos se plantean suponen un reto para la ingeniería de software, la cual ha tenido que replantear muchos de los métodos y paradigmas de programación existentes para los sistemas tradicionales. El objetivo del presente capítulo es proporcionar un panorama general de la teoría del álgebra de procesos y de los sistemas distribuidos.

3.1. Álgebra de procesos

3.1.1. Introducción

La construcción de especificaciones para realizar un análisis sobre el desempeño y eficiencia de sistemas computacionales es un trabajo difícil que requiere inteligencia y experiencia. Debido al constante incremento en tamaño y complejidad de estos sistemas, por ejemplo sistemas inmersos y distribuidos, existe una creciente necesidad para desarrollar métodos útiles para disminuir la complejidad en la especificación de dichos sistemas. El diseño de modelos no solamente produce extensas especificaciones sino que además, debido a la intrincada relación que existe entre los múltiples componentes que un sistema puede contener, produce una estructura altamente irregular que es compleja de entender y controlar.

Modelos de especificación tradicionales como las cadenas de Markov y redes encoladas son ampliamente aceptadas como simples pero efectivas técnicas de especificación de sistemas en diferentes áreas, aun considerando que éstas carecen de características para la especificación de sistemas jerárquicos donde la descomposición de un sistema en componentes ha probado su utilidad para modelar sistemas complejos. Sin embargo, la falta de consideración de estas características en las técnicas anteriormente mencionadas obstaculiza la adecuada especificación de sistemas complejos.

Un prominente ejemplo de formalismo para la descripción y análisis de manera jerárquica y composicional, relacionado con el comportamiento funcional de sistemas es el álgebra de procesos [30]. El álgebra de procesos ofrece un marco de trabajo matemático bien elaborado para el análisis de la estructura y comportamiento de un sistema de manera composicional, es decir, a través de la descomposición del sistema en componentes. El álgebra de procesos incluye mecanismos de abstracción que permiten el manejo de los componentes de un sistema como cajas negras, encapsulando su estructura interna. Las diversas álgebras de procesos están típicamente integradas por una semántica operacional [31], [32] con una estructura formalmente definida que traduce los términos del álgebra de procesos en sistemas de transición etiquetados [33] y viceversa, todo esto con un enfoque composicional.

Los sistemas de transición etiquetados consisten de un conjunto de estados y una relación de transición que describe cómo el sistema evoluciona desde un estado hasta otro. Estas transiciones son etiquetadas con nombres de acciones que representan las interacciones que pueden causar las transiciones. Tales sistemas de transición pueden ser representados con grafos dirigidos con enlaces etiquetados, donde los estados pueden considerarse como nodos y las transiciones como enlaces (etiquetados con nombres de acciones).

3.1.2. Conceptualización

En este apartado se introduce un marco de trabajo sencillo del álgebra de procesos que será utilizado a través del presente trabajo, con la finalidad de proporcionar un entendimiento intuitivo de los aspectos claves del álgebra de procesos. El termino álgebra de procesos es utilizado en diferentes contextos. En primer lugar se considerará la palabra proceso. Un proceso se refiere al comportamiento de un sistema [34]. Un sistema es cualquier cosa mostrando un comportamiento, en particular la ejecución de un sistema computacional, las acciones de una máquina o aún las acciones del comportamiento humano.

El comportamiento es la totalidad de los eventos o acciones que un sistema puede desarrollar, el orden en el cual se ejecutan y puede contener otros aspectos de ejecución como tiempos o probabilidades [35], [36]. Siempre se describen algunos aspectos del comportamiento, omitiendo otros aspectos, por lo que se dice que se obtiene una abstracción o idealización del comportamiento real del sistema. Es posible considerar que se tiene una observación del comportamiento y una acción importante es la selección de la unidad de observación. Normalmente las acciones son discretas: su ocurrencia es en algún momento del tiempo, y diferentes acciones son separadas en el tiempo. Por esta razón es que un proceso algunas veces es llamado también sistema de eventos discretos.

La palabra álgebra denota que se considera una aproximación algebraica/axiomática para describir el comportamiento de un sistema, es decir, se utilizan métodos y técnicas del álgebra universal. Como ejemplo de comparación se considera la definición de grupo en el álgebra matemática.

Un grupo se define como la tupla $(G, *, u, ^{-1})$ con las siguientes leyes o axiomas:

- a * (b * c) = (a * b) * c
- u * a = a = a * u
- a * a^{-1} = a^{-1} * a = u

De esta manera un grupo es cualquier estructura matemática con operadores que satisfacen los axiomas del grupo, o dicho en otras palabras, un grupo es cualquier modelo de la teoría de ecuaciones de grupos. De igual manera se puede decir que un álgebra de procesos es cualquier estructura matemática que satisface los axiomas dados para los operadores básicos, y un proceso es un elemento del álgebra de procesos. Usando los axiomas es posible entonces realizar cálculos con los procesos.

3.1.3. Fundamentos

El modelo más simple de comportamiento es considerar al comportamiento como una función de entrada/salida, así un valor o entrada es dado al inicio del proceso, y algún tiempo después se genera otro valor como salida. Este modelo fue utilizado por la ventaja que proporciona su sencillez en la descripción del comportamiento de un programa de cómputo en ciencias de la computación en la mitad del siglo veinte. Además fue la base en el desarrollo de la teoría de autómatas, en la teoría de autómatas un proceso es modelado como un autómata, un autómata tiene un conjunto de estados y un conjunto de transiciones yendo de un estado a otro.

Una transición denota la ejecución de una acción, la cual es la unidad básica del comportamiento. Se cuenta también con un estado inicial (algunas veces puede ser más de uno) y un conjunto de estados final. Un comportamiento es el recorrido desde el estado inicial hasta el estado final. En un autómata la noción de equivalencia se maneja a través de la igualdad de dos autómatas. Un álgebra basada en ecuaciones que permite razonar sobre un autómata finito determinista es el álgebra de expresiones regulares.

Posteriormente se encontraron carencias en el modelo de entrada/salida, básicamente por su omisión de la noción de interacción: durante la ejecución desde un estado inicial hasta el estado final, un sistema puede interactuar con otro sistema. Esto resulta ser una necesidad imprescindible en la descripción de sistemas distribuidos o paralelos, o también conocidos como sistemas reactivos [37]. Cuando se trabaja con sistemas de interacción (sistemas reactivos) se habla del uso de la teoría de concurrencia. De esta manera, la teoría de concurrencia no es más que la teoría de interacción necesaria en la descripción del comportamiento de sistemas distribuidos y paralelos. Al hablar del álgebra de procesos usualmente se considera como una aproximación de la teoría de concurrencia, así el álgebra de procesos puede especificar adecuadamente el comportamiento de tales sistemas.

De esta forma se puede decir que el álgebra de procesos es el estudio del comportamiento de sistemas reactivos (paralelos y distribuidos) por medios algebraicos, ofreciendo medios para describir o especificar tales sistemas, proporcionando las herramientas para hablar de composición paralela o distribuida. Referente a la composición se pueden mantener dos tipos de composición: composición alternativa (elección) y composición secuencial (secuencia). Con todo esto, es posible analizar estos sistemas utilizando álgebra, es decir utilizando ecuaciones, logrando con ello la

posibilidad de realizar verificación [38], [39] sobre dichos sistemas, es decir, es posible establecer que un sistema cumpla con cierta propiedad.

Pero surge una pregunta inmediata, ¿cuáles son las leyes básicas del álgebra de procesos?, a continuación se listan las leyes estructurales o estáticas del álgebra de procesos. Se inicia dando el conjunto de acciones atómicas y el uso de los operadores básicos para generar procesos más complejos. Como operadores básicos se tienen el operador + el cual denota composición alternativa, el operador ; denota composición secuencial y el operador $\|$ denota composición paralela. Algunas leyes básicas son las siguientes (considere que + tiene la precedencia más baja y que ; tiene la precedencia más alta):

a) x + y = y + x (propiedad conmutativa de la composición alternativa)

b) x + (y + z) = (x + y) + z (propiedad asociativa de la composición alternativa)

c) x + x = x (propiedad de idempotencia de la composición alternativa)

d) (x + y) ; z = x ; z + y ; z (propiedad distributiva de + sobre ;)

e) (x ; y) ; z = x ; (y ; z) (propiedad asociativa de la composición secuencial)

f) x $\|$ y = y $\|$ x (propiedad conmutativa de la composición paralela)

g) (x $\|$ y) $\|$ z = x $\|$ (y $\|$ z) (propiedad asociativa de la composición paralela)

Estas leyes son conocidas como estáticas debido a que no mencionan explícitamente acciones de ejecución sino que únicamente mencionan propiedades generales de los operadores a utilizar.

De esta manera se puede decir que cualquier estructura matemática con tres operaciones binarias que satisfagan estas siete leyes es un álgebra de procesos, las siete leyes listadas anteriormente constituyen el corazón del álgebra de procesos. Frecuentemente las estructuras matemáticas son formuladas en términos de un autómata comúnmente llamado sistema de transición. Esto significa que un sistema de transición tiene un conjunto de estados y transiciones, un estado inicial y un conjunto de estados finales.

Como resumen hasta este punto se puede decir que: el álgebra de procesos es el estudio de teorías de ecuaciones y sus modelos, mientras que el estudio de sistemas de transición, sus estructuras y equivalencias entre ellas es conocido como teoría de procesos.

3.1.4. Sistemas de transición etiquetados

Diagramas de estados, autómatas y modelos similares son ampliamente utilizados para describir el comportamiento de sistemas. Éstos consisten de un conjunto de estados S y representaciones de posibles cambios de estado, la relación entre estados y cambios es dada normalmente a través de una función o relación sobre los estados, es decir, un subconjunto del producto cartesiano $(S \times S)$. Intuitivamente, un par de estados (B,C) están en la relación si es posible cambiar del estado B al estado C en un paso único, tal relación de transición es comúnmente denotada a través de una flecha $(\rightarrow)$, así que $(B,C) \in \rightarrow$ puede ser escrita en notación infija como $B \rightarrow C$, para representar el posible cambio de estado entre B y C.

En el contexto del álgebra de procesos, los sistemas de transición aparecen en una forma específica, a través de sistemas de transición etiquetados. Los sistemas de transición etiquetados forman una clase particular donde los cambios de estados son condicionados a la ocurrencia de acciones pertenecientes a un conjunto de acciones, o alfabeto $A(a \in A)$. Un cambio de estado entre B y C necesita la ocurrencia de una acción relacionada, así la relación de transición $\rightarrow$ es un subconjunto de $S \times a \times S$ en vez de una relación binaria sobre solo S. Nuevamente será conveniente denotar $(B,a,C) \in \rightarrow$, usando un tipo de notación mezclada como $B \xrightarrow{a} C$. En esta notación la acción aparece como la etiqueta de la transición, de aquí surge el nombre de sistema de transición etiquetado.

Definición 3.1. *Un sistema de transición etiquetado es una tupla $(S,A,\rightarrow)$, donde*

- *S es un conjunto de estados no vacío,*
- *A es un conjunto de acciones, y*
- *$\rightarrow \subset$ SxAxS es un conjunto de acciones de transición etiquetadas.*

Para utilizar un sistema de transición etiquetado como un modelo operacional de sistemas, es una práctica común identificar un estado inicial B en el sistema de transición donde comienza la operación. Un sistema de transición con un estado inicial es llamado un proceso.

Definición 3.2. *Un proceso es una tupla $(S,A,\rightarrow,p)$, donde $(S,A,\rightarrow)$ es un sistema de transición etiquetado y $p \in S$ es el estado inicial.*

3.1.5. Procesos básicos

El álgebra de procesos es un medio para especificar procesos y analizarlos. Para conseguir este propósito se utiliza un lenguaje algebraico basado en combinadores, es decir, en operadores que combinan procesos para formar otros procesos nuevos. El término de lenguaje algebraico, las expresiones de comportamiento, son interpretadas como sistemas de transición etiquetados con un estado inicial distinguible, es decir, como un proceso de acuerdo a la Definición 3.2. Para lograr esta interpretación se utilizan reglas de semántica operacional estructurada, esta interpretación semántica induce igualdades entre diferentes expresiones de comportamiento dando como resultado un cálculo de ecuaciones para el análisis de procesos.

El lenguaje algebraico se define a través de una gramática con estilo BNF, se asume un conjunto contable de variables V dado, que es usado para expresar un comportamiento repetitivo. También se asume un conjunto de acciones A, además con $a,b,\ldots$ como elementos de A_τ. Se asume un elemento τ representando acciones internas (ocultas), y se denota a A_τ como el conjunto de $A \cup \tau$.

Definición 3.3. *Sea $a \in A_\tau$ y $X \in V$. Se define el lenguaje AP como el conjunto de expresiones dado por la siguiente gramática.*

$$\kappa = 0 \mid a.\kappa \mid \kappa_1 + \kappa_2 \mid X \mid [X := \kappa]_i$$

$[X := \kappa]$ es una notación para indicar un conjunto finito arbitrario de ecuaciones de la forma $[X_1 := \kappa_1, X_2 := \kappa_2, \ldots, X_n := \kappa_n]$, con $X_i \in V$, y κ_i cumpliendo con la anterior gramática.

Cabe hacer notar que se utilizan $\kappa, \kappa_1, \kappa_2, \ldots$ como expresiones arbitrarias de AP. El significado intuitivo del lenguaje es el siguiente:

- El símbolo terminal 0 denota un comportamiento final.
- La expresión $a.\kappa$ interactúa sobre una acción a y posteriormente pasa a la expresión κ. Se puede decir que κ es una ejecución predefinida por a.
- La expresión $\kappa_1 + \kappa_2$ combina dos alternativas. Puede seleccionar el comportamiento de la expresión κ_1 o el comportamiento de la expresión κ_2. El símbolo terminal $+$ es llamado operador de selección. La selección entre κ_1 y κ_2 es llevada a cabo con la interacción con otros procesos en unión con las acciones iniciales tanto de κ_1 cómo de κ_2.
- La expresión $[X := \kappa]_i$ define un comportamiento en términos del conjunto $[X := \kappa]$ para la definición de comportamientos recursivos. Su significado es como sigue: $[X := \kappa]i$ se comporta como κ_i, donde el comportamiento de las variables de recursión es obtenido con sustitución hacia atrás siempre y cuando el comportamiento de X_j sea alcanzado, así pues éste es reemplazado por su definición $[X := \kappa]_j$.

Las expresiones anteriores están restringidas a que cada ocurrencia de la variable X_j está limitada por la definición de una ecuación $X_j := \ldots$, tales expresiones son conocidas como expresiones cerradas. Una expresión $\kappa \in AP$ es cerrada si cada variable $X_j \in V$ que aparece en κ solamente aparece dentro del alcance de un conjunto de ecuaciones definidas, es decir, dentro de una expresión $[\ldots, X_j := \ldots, \ldots]_i$. El conjunto de expresiones cerradas es denotado AP_c.

La anterior interpretación intuitiva puede formalizarse dando una semántica operacional en términos de sistemas de transición etiquetados. Se define una semántica para expresiones cerradas de AP trasladando el lenguaje completo AP_c a un sistema de transición universal. El espacio de estados de este sistema de transición es el conjunto de todas las expresiones cerradas de acuerdo a la Definición 3.3. Una vez que cada $\kappa \in AP_c$ aparece en algún lugar en el sistema de transición, la semántica correspondiente es determinada por el espacio de estados alcanzable desde esta expresión.

Las primeras reglas de la semántica operacional necesarias están dadas en la Tabla 3.1. Las reglas tienen el formato $B/C\ A$, para expresar que si A se cumple, entonces B implica C, donde A, B, y C son declaraciones acerca de la existencia de transiciones etiquetadas de estado. La notación $\kappa_1\kappa_2/X$ es usada para representar el resultado de una sustitución simultanea de cada ocurrencia de una variable X por la expresión κ_2 en la expresión κ_1.

Definición 3.4. *El sistema de transición universal U es dado por la tupla $(AP_c, A, \rightarrow)$, donde $\rightarrow \in (AP_c \times A \times AP_c)$ es la última relación que satisface las reglas operacionales de la Tabla 3.1.*

Esta definición proporciona una semántica para cada elemento de $\kappa \in AP_c$, a través del fragmento de $\rightarrow$ alcanzable desde el estado κ en U. Para una expresión cerrada κ, $R(\kappa)$ denota el conjunto de estados alcanzables desde κ en la relación de transición universal $U : R(\kappa) = \{\kappa' \mid (\kappa, \kappa') \in T^*\}$ donde T es la relación de transición no etiquetada en U, es decir, $T = \{(\kappa_1, \kappa_1') \in AP_c \times AP_c \mid \exists a \in A.\kappa_1 \rightarrow \kappa_1'\}$.

Definición 3.5. *La semántica de una expresión de comportamiento cerrada $\kappa \in AP_c$ es un proceso $(S, A, \rightarrow', \kappa)$, donde $S = R(\kappa)$ y $\rightarrow' = \rightarrow \cap (S \times A \times S)$.*

Regla semántica
$/a.\kappa \xrightarrow{a} \kappa$
$\kappa \xrightarrow{a} \kappa' / \kappa + \kappa_1 \xrightarrow{a} \kappa'$
$\kappa_1 \xrightarrow{a} \kappa_1' / \kappa + \kappa_1 \xrightarrow{a} \kappa_1'$
$\kappa_i\{[X := \kappa]_i / X_i\} \xrightarrow{a} \kappa' / [X := \kappa]_i \xrightarrow{a} \kappa'$

Tabla 3.1: Reglas semánticas operacionales para AP_c.

Usando esta definición se puede adoptar la convención general de identificar un proceso con su estado inicial. De esta forma, las expresiones cerradas son llamadas procesos.

Con esto se concluyen los aspectos conceptuales más relevantes del álgebra de procesos necesarios para establecer una base para la especificación de un álgebra dirigida a los sistemas distribuidos que se describirá en el capítulo 6 de la presente tesis, en el siguiente apartado se abordarán los aspectos conceptuales de los sistemas distribuidos.

3.2. Sistemas distribuidos

3.2.1. Conceptualización

Los sistemas distribuidos surgen como una idea para permitir una eficiente división del trabajo y el compartimiento de recursos para llevar a cabo este trabajo. La motivación para estos sistemas tuvo como base llevar el concepto de tiempo compartido a una real implantación, donde diferentes tareas pudiesen estar efectuándose simultáneamente, pero con la diferencia de que serán realizadas en un grupo de computadoras más que en una sola, es decir, se extiende el concepto de tiempo compartido y se lleva más allá de la frontera que implica el tener una sola computadora para efectuar los trabajos a desarrollar.

Para continuar avanzando en el estudio de los sistemas distribuidos es conveniente definirlos primeramente, así un sistema distribuido se puede definir como:

"Un conjunto de computadoras independientes interconectados a través de una red de comunicación y equipado con software distribuido que le permitan coordinar sus actividades y compartir recursos de hardware, software o datos" [40].

"Un conjunto de computadoras interactuando entre sí con el objetivo de dar la apariencia de un único sistema ante los usuarios del mismo" [41].

De estas definiciones es posible darse cuenta de dos aspectos a considerar, el primero de ellos es la existencia de una infraestructura de conexión, específicamente una red de comunicación de datos, la cual da el soporte de hardware necesario para construir un sistema distribuido. El segundo aspecto es el soporte de software necesario para dar la apariencia de único sistema ante el usuario, este software está compuesto por diferentes tecnologías las cuales se aplicarán en distintos niveles según el aspecto concreto sobre el cual sean requeridas en el sistema distribuido.

Entre los ejemplos de sistemas distribuidos más comunes en la actualidad podemos citar los

siguientes: sistemas operativos distribuidos, computación móvil, Internet e intranets [41]. En este punto es conveniente distinguir entre dos tipos de sistemas distribuidos, el primer tipo de sistemas distribuidos son los básicos cuya finalidad principal es la de compartir recursos y datos aún cuando la apariencia de único sistema no se cumpla en su totalidad, en este tipo entrarían los tres últimos ejemplos (computación móvil, Internet e intranets). El segundo tipo de sistemas distribuidos son los formales cuya finalidad principal es la creación de un sistema que se apegue al máximo a la definición de sistema distribuido, tal es el caso del primer ejemplo, los sistemas operativos distribuidos.

Todo sistema distribuido tiene en su concepto y diseño las siguientes características principales a cumplir [40]:

- Heterogeneidad. Es la capacidad del sistema para permitir que los usuarios accedan a servicios y ejecuten aplicaciones sobre diferentes y variadas plataformas de redes y computadoras.
- Extensibilidad. Es la capacidad del sistema para permitir la adición de nuevos servicios y aplicaciones.
- Seguridad. Es la capacidad del sistema para proporcionar confidencialidad en el acceso, integridad de la información utilizada y disponibilidad constante del sistema.
- Escalabilidad. Es la capacidad del sistema para permitir el crecimiento del mismo en relación con el número de recursos y usuarios.
- Tolerancia a fallas. Es la capacidad del sistema para continuar en disponibilidad ante la presencia de cualquier tipo de falla corregible.
- Transparencia. Es la capacidad del sistema para ocultar al usuario y al programador de aplicaciones la separación de los componentes que integran al sistema, el objetivo es que se perciba el sistema como un todo.

El cumplimiento en mayor o menor medida de estos objetivos dará al sistema distribuido el tipo al que pertenece de acuerdo a la clasificación que se ha dado previamente, además también definirá la robustez de su diseño. Relacionado con el diseño de los sistemas distribuidos, un punto importante a establecer es justo el modelo arquitectónico que se seguirá para diseñar un sistema distribuido. En el siguiente apartado se proporcionará una revisión a los modelos existentes para el diseño de estos sistemas.

3.2.2. Modelos de sistemas

Al momento de diseñar cualquier tipo de sistema es necesario contar con un modelo arquitectónico, y los sistemas distribuidos no están exentos de este aspecto. Fundamentalmente, un modelo arquitectónico de un sistema distribuido es una descripción sobre la ubicación de sus componentes y la relación que existe entre ellos [42].

Para definir un modelo arquitectónico es necesario contar con una arquitectura de software. Esta arquitectura es la estructura del sistema en términos de componentes especificados por separado, el objetivo global es asegurar que la estructura satisfará las demandas presentes y futuras sobre él.

Los modelos arquitectónicos empleados en los sistemas distribuidos son dos principalmente:

- Modelo arquitectónico cliente-servidor.
- Modelo arquitectónico procesos de igual a igual.

El modelo cliente-servidor es el más utilizado en los sistemas distribuidos, su estructura es en extremo sencilla, existen dos entidades principales: los procesos clientes y los procesos servidores. Los procesos servidores ofrecen una serie de servicios bien determinados a los procesos clientes, los procesos clientes son los consumidores de estos servicios e interactúan con los procesos servidores al solicitarles servicios y recibir como respuesta los resultados del servicio solicitado. Es importante entender en este punto que toda la comunicación en el modelo está basada en el paso de mensajes entre clientes y servidores.

Además, los servidores pueden, a su vez, ser clientes de otros servidores, y aún más, un servicio específico que ofrece el sistema puede estar implantado por un número mayor a uno de servidores capaces de interactuar entre ellos para satisfacer el servicio solicitado; en la Figura 3.1 se muestra el modelo arquitectónico cliente-servidor.

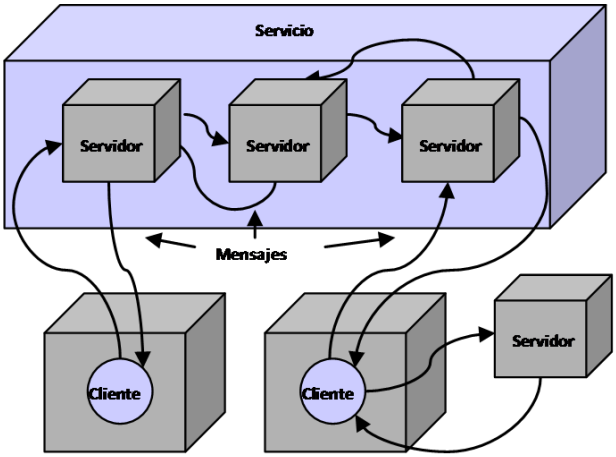


Figura 3.1: Modelo arquitectónico cliente-servidor.

Por otra parte, en el modelo procesos de igual a igual, su arquitectura está basada en procesos en los que todos ellos desempeñan tareas iguales o semejantes, los procesos interactúan cooperativamente como iguales para realizar una actividad o cómputo distribuido sin distinción entre clientes y servidores. En este modelo, el código en los procesos iguales mantiene la consistencia de los recursos y sincroniza las acciones a nivel de la aplicación cuando es necesario, en la Figura 3.2 se muestra el modelo arquitectónico procesos de igual a igual. En general, n procesos iguales podrán interactuar entre ellos dependiendo del patrón de comunicación, y la eliminación de la entidad servidor ocupada en el anterior modelo reduce en cierta medida los retardos de comunicación entre procesos.

Para el diseño de cualquier arquitectura distribuida es necesario cumplir con una serie de requisitos esenciales para un adecuado diseño, estos requisitos [43] son:

- Capacidad de respuesta. Debe ser rápida y consistente.
- Productividad. Debe ser elevada y congruente con el trabajo que se desarrolla.

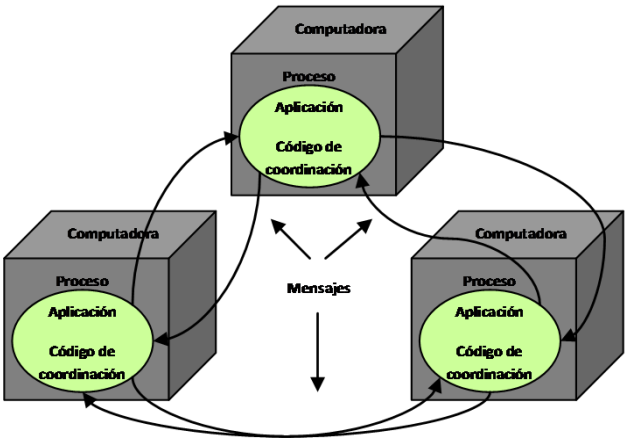


Figura 3.2: Modelo arquitectónico proceso de igual a igual.

- Balanceo de carga. Debe dar la posibilidad de migración.
- Confiabilidad. Debe asegurar trabajo constante y correcto.
- Tolerancia a fallas. Debe mantener esquemas de prevención, detección y corrección de fallas.
- Seguridad. Debe proporcionar acceso controlado, integridad de los datos que utilice y alta disponibilidad.

Los modelos arquitectónicos vistos anteriormente, aunque diferentes, comparten algunos esquemas fundamentales en su especificación, concretamente comparten: un esquema de interacción entre los procesos, un esquema de tolerancia a fallas, y un esquema de seguridad.

El esquema de interacción está relacionado con la manera en que los procesos interactúan entre ellos dentro del modelo, ya sea utilizando procesos clientes-servidores o procesos iguales. Existen dos factores significativos que afectan a los procesos que interactúan en un sistema distribuido. El primer factor está relacionado con las limitantes que producen los medios de comunicación utilizados, el segundo factor está relacionado con la inexistencia de un tiempo global en el sistema. Relacionado con este último factor existen dos variantes del esquema de interacción, estos son los sistemas distribuidos síncronos y los sistemas distribuidos asíncronos. Los primeros están basados en el establecimiento de cotas de tiempo bien conocidas para su trabajo, mientras que los segundos están basados en el establecimiento de un orden estricto de eventos.

El esquema de tolerancia a fallas define las formas en que puede ocurrir una falla para dar una comprensión de los efectos que provocará esta falla en el sistema, siendo las fallas en el canal de comunicación y en los procesos los puntos de falla comunes que se contemplan en este esquema. Las formas de fallas que pueden ocurrir son: aquellas que ocurren sin conseguir la realización de acciones que se supone deberían de haberse realizado por los procesos o canales de comunicación, éstas son conocidas como fallas por omisión; aquellas que ocurren al producirse respuestas falsas o equivocadas por parte de los procesos o canal de comunicación, éstas son conocidas como fallas bizantinas; y aquéllas que se producen por errores en un temporizador en los sistemas distribuidos síncronos, conocidas como fallas de temporización.

El esquema de seguridad está orientado a proteger la confiabilidad en el acceso al sistema, la integridad de la información que se manejará en el modelo y la disponibilidad en todo momento del

sistema, esta seguridad puede lograrse asegurando los procesos y canales de comunicación empleados para sus interacciones y protegiendo los objetos que soportan contra posibles intrusiones. Las técnicas principalmente empleadas en este esquema son; autenticación de usuarios, encriptación de datos, uso de canales seguros, y replicación de información.

Hasta aquí se ha dado un panorama del software necesario para el diseño de un sistema distribuido, en el siguiente apartado se dará un panorama del *hardware* necesario para la construcción de un sistema distribuido completando nuestro estudio.

3.2.3. Redes e interconexión

Un sistema distribuido en su parte de hardware utiliza una red de comunicación de datos para interconectar las distintas computadoras que estarán interactuando entre sí dentro del sistema [44]. Las redes de paquetes y los protocolos basados en capas son la base de la comunicación en los sistemas distribuidos, los tipos de redes que pueden ser empleadas son de área amplia (WAN), de área local (LAN), o de área metropolitana (MAN), sin embargo las redes con mayor uso para los sistemas distribuidos son las redes LAN.

Las redes de área local se basan en la difusión de paquetes en un medio común y Ethernet es la tecnología dominante para llevar a cabo esta tarea, por otra parte, algunos sistemas distribuidos hacen uso también de una red de área amplia, la cual se basa en la conmutación de paquetes para encaminar los paquetes transmitidos hacia sus destinos correspondientes a través de la red. El enrutamiento es el mecanismo clave en la comunicación a través de la red y existen varios algoritmos de encaminamiento que pueden ser utilizados, de los cuales el de vectores de distancia es el más básico pero efectivo para cumplir el propósito.

Las redes son una tecnología basada en el almacenamiento de los paquetes a transmitir dentro de búferes de comunicación, esto tiene como consecuencia que sea necesario un control de la congestión o tráfico en la red para prevenir el desbordamiento de estos búferes en el receptor y en los nodos intermedios.

Básicamente existen dos factores que caracterizan a una red de comunicación de datos, la topología de interconexión y la tecnología del medio de transmisión; la topología de interconexión se refiere a la forma lógica o física en que se conectarán las computadoras que formarán parte de la red. Existen diversas topologías entre las más usuales se encuentran la de bus y la de estrella. La tecnología del medio de transmisión se refiere al tipo de medio físico que se empleará en la conexión de la red, esto incluye tanto a la tarjeta de red como al cable. Actualmente existen dos medios de transmisión ampliamente utilizados, el cable de cobre y la fibra óptica. El cable de cobre ofrece las ventajas de bajo costo y fácil manejo, sus desventajas están en su vulnerabilidad al ruido y su atenuación de señal. La fibra óptica ofrece la ventaja de inmunidad al ruido y sin atenuación en la señal, sus desventajas radican en su alto costo y un manejo que implica grandes cuidados.

Se ha descrito hasta ahora la parte física de una red, pero ¿cómo trabaja de forma lógica una red para la transmisión de información? Previamente habíamos mencionado que el encaminamiento es la clave, pues bien, este encaminamiento es posible gracias a la existencia de protocolos agrupados en capas que dan forma a modelos de interconexión de redes, entre los modelos de interconexión existentes están el modelo OSI y el modelo ATM. El modelo OSI presenta una serie de siete capas y es el modelo más popular de uso en la interconexión de redes, el modelo ATM es más reciente y presenta una serie de tres capas que reducen al mínimo el tiempo de flujo de la información a

través del modelo, además ofrece un ancho de banda muy alto en comunicaciones asíncronas con calidad de servicio garantizada.

Sin embargo, la parte esencial en la transmisión de información son los protocolos, los cuales interpretan y dan sentido a la información que se intercambiará entre las distintas computadoras que integran al sistema distribuido, este protocolo debe ser común y estar presente en cada computadora en la red. Existen diferentes protocolos desarrollados de acuerdo al tipo de necesidad de transporte de información, como un ejemplo de protocolos ampliamente usados están los protocolos de Internet TCP/IP, los cuales hacen posible que las computadoras en Internet se comuniquen con cualquier otra de una forma uniforme, independientemente de que se encuentren en la misma red de área local o en una red de área amplia en otros países. Los estándares de Internet incluyen varios protocolos de aplicación que resultan adecuados para su uso en aplicaciones distribuidas a gran escala. Una versión del protocolo IP de reciente aparición es IPv6, el cual tiene el espacio de direccionamiento necesario para la evolución futura de Internet y aporta los medios necesarios para satisfacer los requisitos de los sistemas distribuidos tales como calidad de servicio y seguridad.

La aparición de redes nuevas como las inalámbricas han dado base para la creación de protocolos orientados a dar soporte en ambientes móviles de comunicación, dando a los usuarios móviles soporte de IP móvil en su deambular por áreas amplias, y por las LAN inalámbricas basadas en el estándar IEEE 802.11 para una conectividad local. Estas nuevas redes utilizan como medio de transmisión señales de radio o señales infrarrojas, transmitiendo a través de estas señales los paquetes de información transportados por el protocolo definido.

Es así como, a través de estas tecnologías que se han desarrollado en las redes de comunicación de datos, es posible conseguir la interconexión necesaria para la creación de sistemas distribuidos, finalmente queda por revisar en el siguiente apartado la forma en que los procesos de un sistema distribuido se comunican entre sí para interactuar y cumplir el objetivo para el cual fueron creados.

3.2.4. Comunicación entre procesos

En este apartado se revisarán las características de los protocolos para la comunicación entre procesos en un sistema distribuido. Primeramente, se debe entender que toda la comunicación en un sistema distribuido está basada en el paso de mensajes y estos mensajes serán transportados a través de los protocolos de comunicación. El paso de mensajes entre un par de procesos se puede basar en dos operaciones básicas, enviar y recibir, definidas en función del emisor y del receptor del mensaje [43].

La secuencia general de comunicación de mensajes para que un proceso se pueda comunicar con otro, es como sigue: el proceso envía un mensaje (una secuencia de bytes) a un destino y otro proceso en el destino recibe el mensaje. Esta actividad implica la comunicación de datos desde el proceso emisor al proceso receptor y puede implicar también la sincronización de los procesos.

De esta forma a cada destino de mensajes se asocia una cola, los procesos emisores producen mensajes que serán añadidos a las colas remotas mientras que los procesos receptores eliminarán mensajes de las colas locales, la comunicación entre los procesos emisor y receptor puede ser de dos tipos; síncrona y asíncrona. En la comunicación síncrona, los procesos emisor y receptor se sincronizan con cada mensaje, es decir, las operaciones enviar y recibir son bloqueantes, lo que quiere decir que a cada operación de *envía* producida, el proceso emisor se bloquea hasta que se produce la correspondiente operación *recibe*, de forma similar cuando se produce una operación *recibe* el proceso receptor queda bloqueado hasta que se recibe el mensaje del proceso emisor.

En la comunicación asíncrona, la utilización de la operación *envía* es no bloqueante, de modo que el proceso emisor puede continuar tan pronto como el mensaje haya sido copiado en un búfer local, y la transmisión del mensaje se lleva a cabo en paralelo con el programa del proceso emisor. La operación *recibe* puede tener variantes tanto bloqueante como no bloqueante, en la variante no bloqueante el proceso receptor sigue con su programa después de invocar la operación *recibe*, la cual proporciona un búfer que será llenado en segundo plano y se le informará al proceso receptor que existe un mensaje en el búfer a través de una interrupción o por medio de alguna otra señal.

La manera en que los mensajes serán encaminados a su destino correspondiente se basa en un direccionamiento construido por pares, dirección IP y puerto local. Una dirección IP es una dirección Internet mientras que un puerto local es un buzón lógico proporcionado por el sistema operativo y que servirá como el destino del mensaje en la computadora, usualmente es representado por un número entero comprendido entre 1 y 65535. Un puerto tiene exactamente asociado un proceso receptor pero puede recibir a través de múltiples procesos emisores, además los procesos pueden utilizar múltiples puertos desde los cuales pueden recibir mensajes y cualquier proceso que conozca el número del puerto receptor puede enviarle mensajes.

Un servicio de mensajes como el empleado en los sistemas distribuido se dice que es confiable si se garantiza que los mensajes se entregan a pesar de poder perderse un número pequeño de ellos, además los mensajes son entregados en el orden en que fueron transmitidos por el emisor.

Los mecanismos de transmisión de mensajes más conocidos en los sistemas distribuidos, además de ser los más empleados, son los sockets y las llamadas a procedimientos remotos (RPC). Un socket es un punto de acceso para transmitir mensajes y proporciona los puntos extremos de la comunicación entre procesos, mientras las RPCs son funciones construidas de forma remota (implantación remota), y que son accedidas a través de otros procesos mediante una interfaz bien definida. Los protocolos asociados a estos mecanismos de transmisión son TCP y UDP, los cuales se encargan del transporte de los mensajes. TCP es un protocolo orientado a conexión que mantiene una conexión constante entre procesos emisor y receptor, mientras que UDP es un protocolo no orientado a conexión y solo establece la conexión cuando es requerido enviar o recibir un mensaje. Estos protocolos en conjunto con IP son los que logran proporcionar el encaminamiento correcto de los mensajes que se transmitirán en la comunicación punto a punto entre procesos en los sistemas distribuidos.

Alternativamente a la comunicación punto a punto, en algunas ocasiones será conveniente mantener una comunicación por multidifusión, es decir una comunicación en grupo. En este tipo de comunicación se envía un único mensaje desde un proceso emisor a cada uno de los procesos pertenecientes a un grupo de procesos, normalmente de forma tal que la pertenencia al grupo resulte transparente al proceso emisor. Esta forma de comunicación en grupo da una alternativa a la transmisión de mensajes en un sistema distribuido.

3.2.5. Procesos concurrentes

Debido a que en la construcción de sistemas distribuidos es muy frecuente encontrarse con la utilización de procesos concurrentes, es conveniente entender qué son los procesos concurrentes. Así pues los procesos concurrentes son los procesos lanzados en procesadores con múltiples unidades funcionales o múltiples CPUs [42].

Pueden ejecutar varias instrucciones a la vez, además el hardware permite la existencia de varios procesos y, el procesador puede conmutar entre ellos y ejecutar instrucciones de varios

procesos conmutando entre ellos. Existen instrucciones en la programación concurrente de alto nivel, para que un proceso denominado padre cree uno o varios procesos denominados procesos hijo. El inconveniente de los procesos concurrentes es que realicen operaciones sobre variables comunes. Esto genera varios problemas que podemos observar en el siguiente ejemplo:

Se realiza la creación de cinco procesos concurrentes que constan de una sola instrucción.

```
Inicio
    P1: a:=4;
    P2: b:=3;
    P3: c:=a - b;
    P4: c:=c + 1;
    P5: c:=c + 1;
Fin
```

Estos cinco procesos comparten las variables a, b y c. Si se supone que el orden de la ejecución es el descrito, el valor de c es c=3. Evidentemente no podemos presuponer ningún orden en la ejecución. Supóngase la sucesión de eventos de la Tabla 3.2:

Inicialmente	Valores de variables
P3: c:=a - b;	{a=?, b=?, c=?}
P1: a:=4;	{a=4, b=?, c=?}
P2: b:=3;	{a=4, b=3, c=?}
P4: c:=c + 1;	{a=4, b=3, c=?}
P5: c:=c + 1;	{a=4, b=3, c=?}

Tabla 3.2: Primera sucesión de eventos concurrentes.

Al finalizar los procesos se desconoce el valor de c. Esto ocurre porque no se ha respetado la precedencia en la ejecución de los procesos. Ahora se supondrá que la sentencia c:=c + 1 se ejecuta mediante tres instrucciones máquina independientes:

```
registroCPU:=c;
registroCPU:=registroCPU+ 1;
c:=registroCPU;
```

Supóngase ahora la sucesión de eventos de la Tabla 3.3:

Se observa que el proceso P4 ha sido interrumpido mientras operaba sobre c. Los incrementos en c que deberían haber generado como resultado c=3 han dado c=2. Al recurso al que pueden acceder varios procesos concurrentes a la vez se le denomina sección crítica.

Se han observado dos errores en el ejemplo:

- Incumplimiento de la precedencia.
- Accesos múltiples a la región critica.

Inicialmente	Valores de variables
P2: b:=3;	{a=?, b=3, c=?}
P1: a:=4;	{a=4, b=3, c=?}
P3: c:=a - b;	{a=4, b=3, c=1}
P4: registroCPU:=c;	{a=4, b=3, c=1}
P4: registroCPU:=registroCPU+ 1;	{a=4, b=3, c=1}
P5: registroCPU:=c;	{a=4, b=3, c=1}
P5: registroCPU: =registroCPU+ 1;	{a=4, b=3, c=1}
P5: c:=registroCPU;	{a=4, b=3, d=2}
P4: c:=registroCPU;	{a=4, b=3, d=2}

Tabla 3.3: Segunda sucesión de eventos concurrentes.

La solución a los problemas de sincronización y exclusión mutua se pueden resolver mediante la herramienta del semáforo para sistemas monolíticos. Un semáforo S es una variable entera que posteriormente a su iniciación, sólo puede ser accedida por dos operaciones estándar e indivisibles denominadas Prueba (P) e Incremento (V).

La definición clásica de P y V es:

P(S): Si S>0 entonces S:= S - 1
en otro caso realizar espera

V(S): Si hay procesos en cola entonces liberar el primero
en otro caso S:=S + 1

Para sistemas distribuidos, la solución a los problemas de sincronización y exclusión mutua se puede obtener a través del intercambio de mensajes, como se mostrará en el presente trabajo al desarrollar, como caso de estudio de LeGESD, el problema clásico de sincronización productor-consumidor en el capítulo 7.

Con esto se concluye la presentación del marco conceptual con los aspectos teóricos más relevantes sobre el álgebra de procesos y los sistemas distribuidos. En el siguiente capítulo, se revisarán los antecedentes y trabajos relacionados con el objeto de estudio del presente trabajo.

4 Lenguajes de especificación de sistemas distribuidos

La especificación de los sistemas distribuidos hace necesario el desarrollo de nuevas técnicas que permitan el diseño, la construcción e implantación de sistemas distribuidos de una forma rápida y eficiente. Los lenguajes de especificación surgen como parte de estas nuevas técnicas. El objetivo del presente capítulo es proporcionar una revisión general de los lenguajes de modelado en sistemas computacionales y del estado del arte en la definición de lenguajes de especificación para el diseño de sistemas distribuidos.

4.1. Lenguajes de Modelado en Sistemas Computacionales

4.1.1. Conceptualización

A lo largo de los años el software se ha vuelto cada vez más complejo, el incremento en la potencia de los equipos de cómputo y la aparición de modernos dispositivos y tecnologías ha generado la necesidad de nuevos sistemas operativos y aplicaciones de mayor complejidad. Simultáneamente, esta complejidad ha traído consigo la demanda de nuevos métodos y herramientas que nos ayuden en la construcción de este software.

Particularmente, uno de los problemas más comunes en el desarrollo de software es que la mayoría de programadores y analistas no contaban con metodologías y herramientas que permitieran el diseño del software a crear, además cada miembro del equipo de trabajo utilizaba una serie de símbolos familiares para él pero no para los demás.

Otro problema muy habitual, relacionado con el anterior, es la absoluta escasez de procedimientos establecidos dentro de la industria para desarrollar software, como mucho, las industrias

de software suelen seguir el método de desarrollo en cascada (requisitos, análisis, diseño, implantación y pruebas).

La especificación de sistemas, en general, es usada para describir los componentes de un sistema, así como también para describir todas las relaciones existentes entre dichos componentes, esta descripción es realizada mediante diferentes técnicas: modelos relacionales de información, modelos orientados a objetos, modelos basados en máquinas de estados finitos, modelos matemáticos dinámicos, o lenguajes de modelado descriptivos auxiliados por herramientas de modelado y metodologías. Es justamente en esta última técnica donde se enfocará la atención en el desarrollo del presente trabajo.

Por otra parte, como complemento al modelado de sistemas, tenemos la implantación de sistemas y la integración de componentes, los cuales son utilizados para el diseño detallado de sistemas y su implantación, éstos incluyen las más recientes herramientas para el desarrollo rápido de sistemas, tales herramientas son: plataformas ”middleware” y de componentes, lenguajes de cuarta generación, implantación de componentes con modelos basados en reglas, componentes de interfaz humano-máquina, y ambientes de ingeniería de software asistidos por computadora (CASE).

Existen lenguajes de modelado de sistemas ampliamente utilizados en el ámbito industrial, entre estos lenguajes uno que destaca es el Lenguaje de Modelado Unificado (UML). En la siguiente sección se describirá en sus aspectos más importantes este lenguaje de modelado.

4.1.2. Lenguaje de Modelado Unificado UML

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar para especificar, visualizar, construir, y documentar los diversos componentes que integran los sistemas de software [15], [16]. UML representa una parte importante del software orientado a objetos tanto en el modelado como en el desarrollo, además UML tiene una característica particular, utiliza notaciones gráficas para expresar el diseño de los sistemas a modelar. UML se basa en anteriores especificaciones tales como la de Diseño Orientado a Objetos (OOD) de Booch, la Técnica de Modelado de Objetos (OMT) de Rumbaugh y el Análisis Orientado a Objetos (OOA) de Coad-Yourdon. Básicamente UML hace una división de cada modelo en un número de diagramas que conforman las diferentes vistas del sistema. Estos diagramas juntos son los que representan el modelado del sistema.

Entre los objetivos principales de UML se encuentran:

- Proporcionar a los usuarios un lenguaje de modelado visual para que éstos puedan desarrollar e intercambiar modelos de sistemas.
- Proporcionar mecanismos de extensibilidad y de especialización para ampliar los conceptos base empleados en el modelado.
- Proveer independencia sobre lenguajes de programación particulares.
- Proporcionar una base formal para entender los lenguajes de modelado.
- Apoyar el desarrollo del paradigma orientado a objetos.

UML surge como respuesta para contar con un lenguaje estándar para modelar diseños de software y muchos han creído ver a UML como solución para todos los problemas, sin saber en

muchos casos que no necesariamente lo es en realidad. UML es una notación estándar para el modelado de sistemas de software, resultado de una propuesta de estandarización promovida por el consorcio OMG (Object Management Group) [45], del cual forman parte las empresas más importantes que se dedican al desarrollo de software, en 1996.

Es importante recalcar que, erróneamente a lo que muchos piensan, UML sólo se trata de una notación, es decir, de una serie de reglas y recomendaciones para representar modelos. UML no es un proceso de desarrollo, es decir, no describe los pasos sistemáticos a seguir para desarrollar software. UML sólo permite documentar y especificar los elementos creados mediante un lenguaje común describiendo modelos.

En este punto es conveniente preguntarse acerca del concepto de modelado, lo que implica. En todos los ámbitos de la ingeniería se construyen modelos, en realidad, simplificaciones de la realidad, para comprender mejor el sistema que se ha de desarrollar: los arquitectos utilizan y construyen planos (modelos) de los edificios, los grandes diseñadores de coches preparan modelos en sistemas CAD/CAM con todos los detalles y los ingenieros de software deberían igualmente construir modelos de los sistemas software.

Un enfoque sistemático permite construir estos modelos de una forma consistente demostrando su utilidad en sistemas de cierto tamaño. Cuando se trata de un programa de cincuenta, cien líneas, la utilidad del modelado parece discutible pero cuando involucramos a cientos de desarrolladores trabajando y compartiendo información, el uso de modelos y el proporcionar información sobre las decisiones tomadas, es vital no sólo durante el desarrollo del proyecto, sino una vez finalizado éste, cuando se requiere algún cambio en el sistema. En realidad, incluso en el proyecto más simple los desarrolladores hacen algo de modelado, si bien informalmente.

Para la construcción de modelos, hay que centrarse en los detalles relevantes mientras se ignoran los demás, por lo cual con un único modelo no tenemos bastante. Varios modelos aportan diferentes vistas de un sistema los cuales nos ayudan a comprenderlo desde varios frentes. Así, UML recomienda la utilización de nueve diagramas para representar las distintas vistas de un sistema. Estos diagramas de UML son los siguientes:

- Diagrama de Casos de Uso: modela la funcionalidad del sistema agrupándola en descripciones de acciones ejecutadas por un sistema para obtener un resultado.
- Diagrama de Clases: muestra las clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan entre sí.
- Diagrama de Objetos: muestra una serie de objetos (instancias de las clases) y sus relaciones.
- Diagrama de Secuencia: enfatiza la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos.
- Diagrama de Colaboración: igualmente, muestra la interacción entre los objetos resaltando la organización estructural de los objetos en lugar del orden de los mensajes intercambiados.
- Diagrama de Estados: modela el comportamiento de acuerdo con eventos.
- Diagrama de Actividades: simplifica el Diagrama de Estados modelando el comportamiento mediante flujos de actividades.

- Diagrama de Componentes: muestra la organización y las dependencias entre un conjunto de componentes.
- Diagrama de Despliegue: muestra los dispositivos que se encuentran en un sistema y su distribución en el mismo.

Con esto se concluye la presentación de los antecedentes más importantes de los lenguajes de modelado en sistemas computacionales, en el siguiente apartado se describirán los trabajos relacionados más relevantes al objeto de estudio de la presente tesis.

4.2. Trabajos relacionados en la definición de lenguajes de especificación de sistemas distribuidos

Existe un conjunto de trabajos de investigación relevantes en la definición de lenguajes de especificación de sistemas distribuidos, en esta sección se mencionarán sus características más importantes y de interés para la presente tesis.

4.2.1. Ambiente de programación visual para sistemas distribuidos (PEDS)

PEDS [10] es un ambiente de programación visual para programar sistemas paralelos y distribuidos. Como lenguaje visual, PEDS es una herramienta de especificación de alto nivel para aplicaciones distribuidas. Los elementos visuales de PEDS son diseñados especialmente para utilizar varios paquetes de software (las plataformas de programación de paquetes de software pueden incluirse en los elementos visuales).

Entre las características principales de PEDS se pueden mencionar que:

- Apoya el diseño jerárquico a través de múltiples niveles de abstracción.
- Utiliza un lenguaje visual para el modelado de aplicaciones a alto nivel mediante grafos y proporciona un conjunto de primitivas gráficas para la construcción de programas.
- Permite que recursos heterogéneos de cómputo (incluyendo hardware y software) sean integrados para solucionar un problema en un marco de trabajo uniforme.
- Maneja los recursos de software y hardware de manera tal que las condiciones de transformación de programas visuales sobre recursos puedan fácilmente proporcionarse y utilizarse. Múltiples procesos pueden trasladarse automáticamente sobre procesadores según la especificación del usuario y la topología del sistema distribuido.

En PEDS, plataformas de programación heterogéneas pueden incluirse en unidades visuales, es decir, en elementos del lenguaje visual. Las unidades visuales se pueden utilizar para modelar aplicaciones en el nivel del lenguaje visual. En el nivel de recurso, debido a que una unidad visual tiene una plataforma de programación específica, se utilizan recursos dedicados. Por lo tanto, PEDS puede utilizar varios paquetes de software a través del apoyo de su nivel bajo y enlazar diversos paquetes de software con el lenguaje visual de alto nivel. El lenguaje visual y los programas de recurso en conjunto forman una especificación completa.

Como ambiente de programación visual, PEDS apoya la programación de dos dimensiones a través de sus características. Permite que el usuario manipule los elementos de programa que usan directamente el ratón y el teclado, y la programación en PEDS es un proceso interactivo. Los programas se pueden modelar por grafos, donde cada nodo del grafo representa un proceso paralelo y un arco que conecta un par de nodos representa la dependencia de los datos o la relación de la comunicación.

Una especificación gráfica tiene generalmente un formato de dos dimensiones, así los grafos de PEDS describen cómo repartir el programa y la granularidad de paralelismo basados en una estructura de proceso estática, donde los procesos y sus conexiones de la comunicación se establecen antes de la ejecución.

Los objetos funcionales en PEDS se llaman unidades visuales. Cada unidad visual es una clase visual, correspondiendo a un objeto y a una clase en un lenguaje orientado a objetos. Hay siete tipos de clases visuales utilizadas por PEDS: datos, nodo, estructura, hardware, software, PEDS-plataforma, y usuario-plataforma. Un nombre de clase se utiliza para representar una clase. Una clase datos puede ser un número entero o una secuencia. Una clase nodo contiene tres ranuras: una ranura de entrada, una ranura de salida, y una ranura para una unidad de programación (PU). Una ranura de entrada se utiliza para recibir mensajes de otros nodos. Una ranura de salida se utiliza para exhibir resultados visualmente en la pantalla o enviarlos a otros nodos. Los mensajes y los resultados están dentro de la clase datos. Una unidad de programación es cualquier clase PEDS-plataforma que apoye a PEDS o una clase usuario-plataforma que sea definida por el usuario para la programación.

Una unidad de programación procesa los mensajes que vienen de la ranura de entrada, produce resultados, y pone los resultados en la ranura de salida. Una PU es un ambiente de programación autónomo, y se puede definir con mayor detalle por un nivel inferior PU. PU se puede diseñar especialmente para cualquier paquete de software particular, por ejemplo un ambiente de programación de C es una PU.

En PEDS, debido a que la especificación se realiza en base a una PU, se puede decir que una PU es el medio visual de la especificación. Al construir un grafo, un nodo puede representar una abstracción con sus tres ranuras sin llenar. Cuando las tres ranuras de un nodo se llenan, el nodo se llama un nodo ejecutable. Los datos y los nodos ejecutables se pueden ensamblar en una clase estructura, que se puede también ensamblar en otras clases estructura. Una clase estructura proporciona facilidad para ocultar la información.

Los datos o la estructura usados en un nodo deben ser conocidos por la clase usuario-plataforma, a través de la clase software. Una clase software define un paquete de software disponible en el sistema y es una clase visual a través de la cual un programa se puede construir en un nodo. Cuando una unidad de programación se agrega a PEDS, debe ser instanciada por una clase software. Para un nodo lleno, es decir un nodo ejecutable, la clase software consumirá los datos en la ranura de entrada del nodo y accionará la unidad de programación.

La clase hardware describe el procesador utilizado en el ambiente distribuido, sobre el cual PEDS trabaja. Varias clases hardware que representan procesadores heterogéneos se pueden ligar para formar una red distribuida. Pueden también ser ligados a las clases software para indicar las relaciones entre ellas.

Las relaciones entre las clases que se han descrito anteriormente forman la jerarquía del ambiente PEDS. La arquitectura del ambiente PEDS se apoya en cuatro niveles de jerarquía: nivel físico, nivel de soporte, nivel visual del lenguaje, y nivel de uso. El nivel físico especifica los pro-

cesadores y la interconexión de la red que sirven como ambiente distribuido. La especificación del nivel físico se puede hacer por las clases hardware y las relaciones entre ellas. El nivel de soporte acomoda los paquetes de software. Las clases software son utilizadas para representar paquetes de software. El nivel visual del lenguaje tiene un lenguaje visual para construir usos. Las clases visuales se especifican en este nivel. El nivel de uso incluye los programas y los nodos visuales que se crean dentro de PEDS. La relación entre dos niveles cualquiera, es uno de los siguientes: relación de red - en el nivel físico, los procesadores (hardware) se ligan juntos, de modo que puedan comunicarse uno con otro; relación de recurso - si un paquete de software puede trabajar en un procesador, el paquete de software y el procesador tienen una relación de recurso; relación 'tras' - si una clase visual v se diseña para utilizar los paquetes de software $s1 \ldots sN$, entonces existe una relación 'tras' existe entre v y $s1 \ldots sN$; relación 'tras' visual - si cualquier especificación en una clase usuario-plataforma $u1$ se puede convertir a la especificación en otra clase usuario-plataforma $u2$, una flecha de la clase $u1$ a $u2$ especifica la relación 'tras' visual entre él. Esta relación refleja el control y la abstracción de la clase usuario-plataforma y representa la relación de conversión entre las clases usuario-plataforma. Por ejemplo, un programa escrito en la plataforma ANSI-C se puede convertir a otras plataformas de la programación de C si son de la extensión de ANSI-C; relación de la composición - si una unidad visual v consiste de otras unidades visuales $v1 \ldots vN$, entonces v y $v1 \ldots vN$ tienen una relación de composición.

La programación visual en PEDS es un proceso de componer y ligarse. Las relaciones anteriores restringen los recursos que las unidades visuales pueden obtener. Por estas relaciones, cada unidad visual puede encontrar su hardware en el cual pueda funcionar. Para cada unidad visual, el usuario puede especificar las operaciones basadas en reglas. Usando estas reglas, cada unidad visual de un programa visual puede asignarse a un sistema de procesadores en los cuales puede funcionar. Por lo tanto, PEDS puede enviar las unidades visuales a los procesadores en una estrategia predefinida.

En la especificación formal para el lenguaje visual de PEDS, tanto textos como cuadros gráficos sirven como los medios para transportar la información que es llamada el significado lógico. Como en un lenguaje textual, se utiliza la estructura de datos para representar y para almacenar un tipo de información. La información de una clase corresponde a los datos de la estructura que representa la clase. Una estructura s con un dato d se representa como $d : s$. Semejante a un lenguaje textual, una representación visual tiene su propia estructura de datos que representa su aspecto geométrico y características manipulables, y se le llama significado geométrico.

Al programar, los objetos visuales se pueden manipular en la pantalla, en esta manipulación se cambian los datos de su estructura de datos correspondiente. El cambiar los datos es reflejado en su perspectiva por un intérprete predefinido de esa estructura de datos. No solamente una operación visual (por ejemplo, mover el ratón) puede cambiar la perspectiva de un objeto visual, sino también otros cambios en el ambiente (por ejemplo, cambio de características visuales subyacentes) puede también afectar el objeto visual. PEDS llama a una operación que pueda afectar un objeto visual como un acontecimiento. Los acontecimientos se pueden dividir en tres categorías: acontecimientos geométricos que afectan solamente la perspectiva de un objeto visual; acontecimientos lógicos que afectan solamente los datos lógicos; y acontecimientos visuales que afectan las piezas lógicas y geométricas.

4.2.2. Lenguaje de especificación y herramientas de construcción (IOA)

IOA [8] es un lenguaje y conjunto de herramientas para el diseño, el análisis, y la construcción de sistemas distribuidos, este lenguaje para la especificación de sistemas distribuidos está siendo desarrollado en el laboratorio de ciencias de la computación del Instituto de Tecnología de Massachusetts (MIT), el grupo de trabajo está dirigido por Nancy Lynch, apoyada por Mandana Vaziri, Joshua A. Tauber y Michel Tsai. IOA es un lenguaje de especificación, acompañado por un conjunto de herramientas que se utilizan en la producción de software distribuido.

El proceso de desarrollo de IOA comienza con una especificación de alto nivel, a continuación se refina esa especificación a través de diseños sucesivos más detallados, y finaliza con la generación automática de programas distribuidos [46], [47]. El lenguaje y las herramientas de IOA utilizan la descomposición del sistema, lo cual ayuda a hacer los programas distribuidos comprensibles y fáciles de modificar. Lo más importante es que las herramientas proporcionan una variedad de métodos de la validación (teorema de prueba, verificación del modelo, y simulación), que se pueden utilizar para asegurarse de que los programas generados son correctos, conforme a estados indicados, los cuales son referentes a los servicios externos que proporciona el sistema (por ejemplo, los servicios de comunicación).

IOA se basa en el modelo del autómata de entrada/salida. Este modelo se ha utilizado para describir y verificar muchos algoritmos distribuidos, también se ha utilizado para expresar la imposibilidad de llegar a ciertos resultados. Las características del modelo son: sus conceptos fundamentales son matemáticos (más que lingüísticos), proporciona nociones del comportamiento externo, y utiliza un conjunto de métodos compuestos para prueba.

El modelo del autómata de entrada/salida fue desarrollado originalmente para razonar sobre algoritmos teóricos, pero se ha aplicado de forma práctica a servicios de sistema tales como memoria compartida distribuida, comunicación de grupo, y servicios de comunicación tales como TCP. Su uso ha ayudado a resolver ambigüedades y corrección de errores lógicos en estos servicios. Los éxitos prácticos han convencido de que el modelo del autómata de entrada/salida puede desempeñar un papel importante en el desarrollo de verdaderos sistemas distribuidos. Sin embargo, hasta este momento, existen barreras serias en su uso. El problema principal, que se aplica a todos los esquemas formales de validación, es la carencia de una conexión clara y formal entre los diseños verificados y el código final correspondiente. Aunque es factible verificar la corrección del diseño abstracto de un sistema distribuido usando un teorema de prueba, no hay manera conveniente de extender esta prueba al código real. Por ello, el diseño verificado se debe recodificar en un lenguaje de programación como C++ o Java antes de que pueda ser ejecutado el sistema distribuido.

Este paso de recodificación implica la costosa duplicación de esfuerzo y puede introducir errores. Un problema relacionado es la carencia de un lenguaje de programación para sistemas distribuidos que sea conveniente para la verificación y generación del código. Las características que hacen un lenguaje conveniente para la verificación (estilos axiomáticos, simplicidad, no determinístico), son diferentes de las que hacen conveniente un lenguaje para la generación del código (estilo operacional, potencia expresiva, determinístico). Otro problema para los autómatas de entrada/salida ha sido la indisponibilidad de herramientas ligeras de validación tales como los simuladores y los inspectores del modelo. Tales herramientas pueden otorgar una rápida retroalimentación para ayudar en la depuración, antes de realizar una prueba formal que consuma gran cantidad de tiempo.

El diseño del lenguaje de IOA logra un equilibrio razonable entre los requisitos para la verifi-

cación y la generación del código. La semántica de IOA se basa firmemente en las matemáticas, de tal modo proporciona una base sólida para la verificación. IOA permite el no determinismo, de modo que los usuarios puedan expresar diseños tan generales como sea posible. IOA permite las descripciones axiomáticas y operacionales para las transiciones, por separado o en combinación, puesto que ninguno de los dos por sí solo es suficiente para todos los propósitos. Cuando determinada expresión en un programa presenta problemas para la herramienta tal como el generador de código, otra herramienta permite al usuario refinar el programa de tal manera que ambos eliminan las expresiones problemáticas. Las herramientas de IOA permiten al usuario establecer su diseño a un rango de métodos de validación, incluyendo una prueba completa utilizando un teorema de prueba interactivo, un estudio de pruebas de ejecución seleccionadas utilizando un simulador, y un exhaustivo estudio de casos pequeños del diseño utilizando un inspector de modelo. Las herramientas de IOA asisten al programador en la descomposición del diseño en componentes que trabajan separadamente, basado en el autómata de entrada-salida, y en la refinación de éste usando niveles de abstracción, basados en inclusión de rastros y en relaciones de simulación. Características nuevas de IOA incluyen ayuda para niveles de abstracción, ciertos aspectos del método de simulación (por ejemplo, simulaciones en pares, para probar si un autómata refina otro), y ciertos aspectos del método para la generación de código (por ejemplo, canales abstractos para incorporar servicios estándares de comunicación en sistemas distribuidos).

El generador de código de IOA compila código del lenguaje de IOA en código ejecutable para un sistema distribuido. El sistema que sirve de caso de estudio es una configuración arbitraria de nodos y de canales de comunicaciones. El código generado será bastante eficiente para utilizarse en aplicaciones reales.

El proceso de generación de código, tanto como sea posible, se desarrolla dentro del marco de IOA. El diseño procede, posiblemente en varios niveles de la abstracción, desde un punto donde se conforma la descripción final de IOA muy cerca al hardware disponible y a los servicios de software. La descripción final de IOA consiste en una colección de autómatas de 'nodo', uno para cada máquina real, y los autómatas del 'canal', para los servicios de comunicación proporcionados externamente. Entonces cada autómata de nodo se traduce automática y simplemente al código ejecutable que, en un sentido exacto, implementa el autómata de nodo. Los autómatas de canal se implementan por los servicios de comunicación externos proporcionados.

Se procura permanecer dentro del marco de IOA hasta el límite permitido por el modelado formal y los recursos de verificación. Estos recursos se pueden utilizar para asegurarse de que el sistema final coloca descripciones de alto nivel de IOA, conforme a características asumidas de los servicios externos proporcionados. Estos recursos se pueden también utilizar para proveer al sistema final toda la estructura que el marco puede expresar (con la composición y niveles de la abstracción).

Cabe resaltar que los programas IOA de los cuales se genera código real que empate con el hardware y servicios disponibles, y requiriendo los programas nodo para tolerar retardos en la entrada de información, evitan la necesidad de la costosa sincronización global en la implantación del sistema especificado.

El sistema de IOA utiliza la programación con canales abstractos manteniendo bibliotecas de descripciones de IOA para canales abstractos y reales, además de implantaciones de IOA de canales abstractos en términos de canales reales. El sistema también asiste en probar la adecuada implantación de canales abstractos.

Los sistemas de autómatas de IOA se describen como composición de los autómatas de nodo y

de los autómatas de canal. Los autómatas del canal son abstracciones que describen el comportamiento en el mundo real de los servicios de comunicación (por ejemplo, TCP, MPI, UDP, etc.) no implementados inicialmente en IOA. A menudo será provechoso describir el diseño del sistema como composición de los autómatas de nodo y de los autómatas de canal abstracto. Tales autómatas abstractos serán típicamente mucho más simples que los autómatas para los canales reales. Cada autómata abstracto del canal se puede colocar entonces en ejecución en términos de los autómatas del canal real y de un autómata del protocolo asociados a cada nodo implicado. El autómata de nodo es (formalmente) la composición del autómata de nodo con los autómatas del protocolo para ese nodo que aparecen en la implantación del canal.

Es este autómata compuesto (transformado en una forma primitiva, y eliminando su no determinismo), el que se consigue trasladarlo por el generador de código al lenguaje Java.

4.2.2.1. Generación de código basada en autómatas de entrada/salida

De acuerdo a lo expuesto en el apartado anterior se puede decir que el modelo de autómatas de entrada/salida especifica componentes en sistemas concurrentes asíncronos como sistemas de transición etiquetados. Se utiliza un lenguaje preciso para la descripción del autómata de entrada/salida y para la descripción del estado de sus propiedades (IOA) [48]. Así IOA utiliza especificaciones para definir la semántica de tipos de datos abstractos y de autómatas de entrada/salida.

Basados en IOA, se pretende crear un generador de código cuyo objetivo es crear una colección de programas que se ejecuten en un ambiente distribuido y cuya correcta ejecución de dichos programas haya sido probada, sujeta a estados asumidos relacionados con los servicios del sistema proporcionados externamente (servicios de comunicación), y sujeta a asumir la correcta implantación de los tipos de datos proporcionados de manera manual.

El generador de código pretende eliminar todas las formas de indeterminismo existentes [49]. Esto consiste en eliminar todas las instancias de indeterminismo implícito, transformándolas en declaraciones explícitas, y teniendo el usuario que remplazar todas las declaraciones explícitas con declaraciones determinísticas. Las transformaciones que hacen indeterminismos implícitos en explícitos, se describen a nivel programas en IOA, a esto se le llama transformación sintáctica NAD (”Next-Action Deterministic”), por otra parte también se describen a nivel del modelo del autómata de entrada/salida, a esto se le llama transformación semántica NAD. Se utiliza la transformación semántica NAD para probar que la transformación sintáctica realizada es correcta de acuerdo a lo definido.

Un autómata de entrada/salida A es una tupla consistente de los siguientes componentes:

- SIG(A): una firma, consistente de tres diferentes conjuntos de acciones de entrada INA, de acciones internas INTA, y de acciones de salida OUTA.
- STATES(A): un conjunto de estados.
- STATES0(A): Un conjunto no vacío de STATES(A) conocido como estados iniciales.
- TRANS(A): una relación de transición de estados.

Un fragmento de ejecución de A es una secuencia de estados y acciones $S_0, \pi_0, S_1, \pi_1, \ldots$, iniciando en un estado tal que $(S_i, \pi_i, S_{i+1}) \in TRANS(A)$. Una ejecución de A es un fragmento de

ejecución que comienza en un estado inicial. Un trazo de A es la subsecuencia de una ejecución de A consistente de todas las acciones de entrada y salida. La transformación de un autómata de entrada/salida en un autómata determinístico de próxima-acción, es un paso dentro de la generación de código.

El generador de código actualmente está bajo desarrollo (hasta la última información recabada), el cual está diseñado para transformar programas en IOA en programas ejecutables escritos en un lenguaje estándar. Para transformar el algoritmo original expresado en IOA en un programa ejecutable, el programador es guiado a través de una serie de refinamientos sucesivos para crear una forma equivalente del programa que está siendo transformado. Generalmente el programador comienza con una descripción global simple del comportamiento del sistema y sus interfaces con su ambiente (modelo a alto nivel). Por otra parte, a bajo nivel existe una versión equivalente de la especificación, el cual es una colección de autómatas interactivos cuyas formas corresponden a la naturaleza distribuida del ambiente escogido, el cual preserva las propiedades importantes y la interfaz del sistema.

El proceso de generación de código comienza con un autómata diseñado para ejecutarse en un nodo computacional del sistema distribuido. Para generar código de un sistema completo, el proceso mencionado anteriormente se repite para cada nodo algorítmicamente diferente.

El proceso consiste de una serie de refinamientos del autómata que especifica el algoritmo. El autómata es conectado al sistema de servicios externos y es transformado hasta eliminar indeterminismos implícitos. El programa en IOA que resulta puede entonces ser transformado automáticamente en un lenguaje imperativo que se elija, y ligado a bibliotecas que implementen el sistema de servicios externos.

El generador de código está estructurado como un conjunto de pequeños módulos, cada uno de los cuales realiza una transformación. Las transformaciones son hechas sobre los códigos fuente en IOA hasta el último paso de la transformación. Solamente en este último paso es transformado el programa en IOA a un programa en el lenguaje elegido.

El generador de código no genera todo el código para el sistema distribuido completo, sólo genera el código especializado necesario para implantar el algoritmo especificado en cada nodo, lo cual representa una desventaja.

Para cada sistema de servicio externo, se diseñan al menos dos especificaciones. La primera especificación es un modelo abstracto que describe la interfaz y el comportamiento del servicio que el programador desea utilizar (por ejemplo, servicio punto a punto, servicio de canales FIFO). La segunda especificación es un modelo concreto que corresponde a la interfaz y el comportamiento del servicio actual con que se cuenta (por ejemplo, MPI). Se introducen uno o más autómatas auxiliares, escritos en IOA, que interactúan con el modelo concreto para implementar el servicio abstracto. Para crear un ejecutable completo en cada nodo, el generador de código actualmente genera código para la composición del autómata del algoritmo y todos los autómatas auxiliares para el sistema de servicios que el algoritmo utilice.

Como se ha mencionado, el generador de código consiste en una serie de programas de transformación. Cada uno de ellos acepta uno o más programa en IOA como entrada y produce otros programas en IOA como salidas (excepto en el último paso de transformación, el cual traslada el programa en IOA en un programa en el lenguaje de programación escogido).

El primer módulo para cualquier programa en IOA de entrada es el analizador de sintaxis (parser) y verificador de semántica, el cual produce una representación intermedia que puede utilizarse por otras herramientas. El segundo módulo es el generador de interfaz, quien conecta un autómata

del algoritmo con la consola del sistema de servicios externos. La herramienta de composición es el tercer módulo, y convierte la descripción de un autómata de composición en una forma primitiva mediante representaciones explícitas de sus acciones, estados, transiciones, y tareas. El cuarto módulo es el determinante de próxima-acción (NAD), quien convierte el programa en IOA de entrada en un programa determinístico de próxima-acción equivalente. Finalmente, el quinto módulo es el emisor de código, el cual transforma un autómata de nodo primitivo y determinístico en código ejecutable para el lenguaje elegido que implementa el autómata en el nodo correspondiente.

4.2.3. Lenguaje de especificación y herramientas de construcción (LfP)

LfP [9] es un lenguaje y conjunto de herramientas para el diseño, el análisis, y la construcción de sistemas distribuidos. Este lenguaje para la especificación de sistemas distribuidos está siendo desarrollado en el laboratorio de informática de la Universidad de París VI (Pierre et Marie Curie), el grupo de trabajo está dirigido por Fabrice Kordon, apoyado por Claude Girault, Dan Regep, Frédéric Gilliers y Jean-Pierre Velu.

LfP es un lenguaje de especificación para el desarrollo rápido de prototipos de sistemas distribuidos, básicamente LfP tiene como característica ser un lenguaje gráfico que proporciona las facilidades de un Lenguaje de Descripción de Arquitectura (ADL), además LfP se puede enlazar a una metodología basada en UML [50]. Este lenguaje intenta proporcionar una descripción de UML con información estructurada que permita la verificación formal y la generación automática de programas distribuidos. Para su desarrollo, LfP toma como base los principios contenidos en ODP.

Como se ha mencionado, LfP propone ser un lenguaje de especificación para los sistemas distribuidos, siendo sus metas [51], [52] las siguientes:

- Ampliar UML para hacerlo tanto un ADL y como un lenguaje de coordinación gráfico.
- Permitir una verificación formal utilizando modelos semánticos equivalentes en redes de Petri.
- Prever constructores para la generación automática del código de las aplicaciones distribuidas.

La idea principal detrás de LfP es desarrollar aplicaciones distribuidas usando diversos aspectos combinados: software de descripción de arquitectura, verificación formal de modelos propuestos e implantación de aplicaciones en una arquitectura determinada. LfP utiliza dos vistas ortogonales como lo establece ODP:

- la vista funcional (generada como diagrama)
- la vista de implantación (generada como anotaciones textuales).

El diagrama funcional de LfP explica cómo los componentes del sistema (objetos de software), se comportan e interactúan recíprocamente. Este diagrama también describe la arquitectura del software del sistema. Una notación gráfica inspirada en lenguajes de coordinación con capacidades

de ADL se utiliza para definir un diagrama funcional. Los diagramas funcionales reutilizan conceptos de ODP (Open Distributed Processing) [53], tales como objetos computacionales y objetos básicos de ingeniería.

La vista de implantación de LfP define la topología de la implantación del sistema, es decir, cómo se implantan los componentes del sistema (por ejemplo, sistema operativo a utilizar, lenguaje de programación). La vista de implantación se inspira de los puntos de vista de ingeniería y tecnológicos manejados por ODP.

La aproximación de LfP se divide en cinco pasos:

1. Los modelos existentes de UML se transforman en las representaciones (esqueletos) correspondientes de LfP. Las equivalencias entre UML y LfP deberán basarse en lo establecido por LfP. Este paso debe ser automático.

2. El esqueleto de LfP tiene que ser enriquecido para expresar correctamente las interacciones entre los componentes.

3. Las especificaciones formales, tales como redes de petri, se derivan automáticamente de los diagramas funcionales de LfP para permitir la prueba del sistema.

4. Otro enriquecimiento se requiere antes de la implantación de la aplicación. Los diseñadores tienen que especificar la vista de implantación por medio de las anotaciones agregadas al diagrama funcional de LfP.

5. Las dos vistas de LfP se proporcionan como entrada al generador de código para sintetizar la aplicación distribuida correspondiente.

LfP está conformado por dos entidades principales: clases y medios. Una clase de LfP corresponde a una clase completa instanciable de UML. De esta manera, las clases virtuales o abstractas de UML no tienen ninguna correspondencia en LfP. Los medios se utilizan para conectar clases, básicamente especifican el contrato de interacción y la semántica de la comunicación. Los medios corresponden a una asociación, una agregación o una composición de UML.

4.2.3.1. Generación de código

La generación de código en LfP es una etapa que hasta la última información recabada esta en desarrollo. Básicamente lo que se tiene es la idea de cómo deberá de realizarse esta generación [54].

La generación de código asegura que debe de ser independiente de la fase de especificación del sistema. La vista de implantación será la base a partir de la cual se comenzará la generación. Esta vista define para estos propósitos lo siguiente:

- Los puntos de entrada de la partición y de la ejecución de la aplicación.
- El lenguaje de implantación y el ambiente de ejecución a utilizar.

Además, para cada clase de LfP, el diseñador del sistema tiene que proporcionar la siguiente información:

- Modelo de ejecución: unidades activas de ejecución (procesos o hilos) o unidades pasivas (bibliotecas).
- Lenguaje de programación y ambiente de ejecución a utilizar.
- Membresía de la partición de clases (nombre del paquete si existe).

Para cada uno de los medios de LfP se especifican las siguientes cualidades:

- Modelo de ejecución (activo o pasivo).
- Arquitectura de comunicación a utilizar: pipa, socket, u otro sistema de comunicación.

La estructura general del generador de código proporciona el sistema de primitivas requeridas para implantar cada instrucción del modelo. De esta forma la generación del código debe implantar el autómata de cada componente de la especificación, traduciendo instrucciones de LfP en el lenguaje de programación a utilizar, o insertando llamadas en tiempo de ejecución donde sea apropiado. La estructura del generador tiene los siguientes aspectos:

- Los componentes de LfP (clases y medios) son implantados como clases de Java.
- los tipos de LfP son implantados como clases de Java, cada tipo LfP de elementos ordenados tienen un patrón para construir las correspondientes clases de Java.
- Los métodos de LfP son trasladados a métodos de la clase de Java correspondiente.
- Los atributos de una clase de LfP son trasladados a atributos de la clase de Java.
- Variables locales de la especificación hecha con LfP son declaradas en el bloque de instrucción que implanta la construcción donde son declaradas.

El aspecto dinámico de los diagramas de LfP es manejado creando el código para los estados y las transiciones del autómata. El diagrama principal se genera en un método principal, el cual es un método por defecto que servirá como inicio. La generación del código para los métodos sigue el mismo patrón, sólo cambia el nombre del método de Java.

Como se puede apreciar, el grupo de trabajo de LfP está en etapa aún de conclusión de la herramienta de generación de código mientras que el grupo de trabajo de IOA y de PEDS tienen mayores adelantos. Sobre la idea manejada en LfP se puede detectar una posible limitación de su metodología al orientarla a sistemas concurrentes inmersos, lo cual puede dejar restringida su aplicación en sistemas distribuidos a gran escala. Por otra parte, el tratar de tender un puente de unión entre UML y ADL a través de LfP puede resultar en una metodología dependiente de esta unión siendo no deseable quizá en algunos diseños de sistemas distribuidos.

Con esto se concluye la revisión de los antecedentes y trabajos relacionados con el objeto de estudio. En el siguiente capítulo, se desarrollará la propuesta de solución que propone la presente tesis.

5 Lenguaje Gráfico de Especificación de Sistemas Distribuidos

El Lenguaje gráfico de especificación de sistemas distribuidos constituye la propuesta que realiza la presente tesis como parte de las nuevas técnicas que permitan la especificación, construcción e implantación de sistemas distribuidos de una forma rápida y eficiente. El objetivo del presente capítulo es presentar el Lenguaje Gráfico de Especificación de Sistemas Distribuidos (LeGESD), su notación gráfica y sintaxis asociada.

5.1. Introducción

El objeto de estudio de la presente tesis son los sistemas distribuidos y su especificación para desarrollar prototipos de los mismos, y posibilitar su posterior refinación a través de la verificación de la especificación. Para iniciar con la descripción de la propuesta que se realiza en este trabajo, es necesario definir qué es lo que se entenderá como sistema distribuido y cuáles son sus elementos relevantes a considerar. En el capítulo 3 se mencionaron dos definiciones que se encuentran en la literatura clásica de sistemas distribuidos:

"Un conjunto de computadoras independientes interconectados a través de una red de comunicación y equipado con software distribuido que le permitan coordinar sus actividades y compartir recursos de hardware, software o datos" [40].

"Un conjunto de computadoras interactuando entre sí con el objetivo de dar la apariencia de un único sistema ante los usuarios del mismo" [41].

De estas definiciones es posible determinar dos aspectos importantes a reflexionar, el primero de ellos es la existencia de una infraestructura de conexión en el sistema, específicamente una red de comunicación de datos, la cual da el soporte de hardware necesario para construir un sistema distribuido.

Siguiendo la reflexión de este primer aspecto, y sin demeritar su importancia en el funcionamiento a bajo nivel de un sistema distribuido, esta infraestructura de conexión es un elemento externo al comportamiento lógico global de los sistemas distribuidos, y este comportamiento lógico global es justamente el elemento fundamental que se propone considerar en esta tesis, dejando a un lado la infraestructura de conexión del sistema distribuido. Esto bajo el argumento de que para la especificación de los sistemas distribuidos no impacta en forma significativa esta infraestructura de conexión en el comportamiento lógico global del sistema. Esta es una de las diferencias que presenta este trabajo con respecto a PEDS, sin embargo en la propuesta que se realiza sí se considera la especificación del comportamiento lógico del medio de comunicación con el cual interactuará el sistema distribuido.

El segundo aspecto a reflexionar de las definiciones anteriores, es el soporte lógico necesario para proporcionar el funcionamiento distribuido del sistema. Este soporte lógico es al que la tesis enfoca su propuesta y está compuesta por diferentes símbolos gráficos, sus reglas de construcción (sintaxis) y sus significados (semántica), los cuales se aplicarán en distintos niveles según el aspecto concreto sobre el cual sean requeridos en la especificación del sistema distribuido. Estos símbolos gráficos, su sintaxis y su semántica, constituyen la propuesta que se realiza en la presente tesis del Lenguaje Gráfico de Especificación de Sistemas Distribuidos, el cual se ha nombrado LeGESD, y que cuenta con una estructura bien definida que da sustento a la especificación gráfica (visual) de sistemas distribuidos.

En este punto es conveniente definir el concepto de sistema distribuido que en la tesis se utilizará como marco de trabajo. De esta forma:

Definición 5.1. *Un Sistema Distribuido $D = (V, E)$ está compuesto por un conjunto V de procesos que interactúan entre sí mediante un conjunto E de enlaces para lograr un fin común.*

- *Cada proceso se especifica por medio de un autómata que describe la secuencia de pasos ejecutados por cada proceso.*
- *Los autómatas son aumentados con dos operaciones que permiten enviar y recibir mensajes por medio de un canal de comunicación.*
- *Dos procesos $u, v \in V$ pueden interactuar por medio de mensajes, si y sólo si existe un canal de comunicación $(u, v) \in E$ entre ellos.*

Esta definición identifica y separa a un sistema distribuido en entidades lógicas integradas por dos capas. La primera de ellas tiene como elemento el comportamiento lógico propio del sistema distribuido reflejado en los procesos que lo conformarán, y la segunda capa tiene como elemento a la plataforma tanto de comunicación (mecanismos de comunicación) como de ejecución (sistema operativo y lenguaje de programación) con las que el sistema interactuará durante su ejecución.

Por lo tanto, el Lenguaje Gráfico de Especificación para Sistemas Distribuidos (LeGESD) es un lenguaje visual formal para la especificación de sistemas distribuidos, entendiéndose a éstos bajo la Definición 5.1.

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 y una sintaxis que definen la comunicación y aspectos dinámicos del comportamiento del sistema. Adicionalmente, la semántica de LeGESD se basa en una especificación algebraica con semántica operacional basada en el álgebra de procesos [30], a la cual se ha 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, ayuda a la etapa de verificación del modelo diseñado. En el capítulo 6 se describirá a detalle a ADSD.

5.2. Modelos de distribución

En los sistemas distribuidos un aspecto importante a considerar es el tipo de modelo de distribución que utilizarán los programas distribuidos. Entre los tipos de modelos de distribución más comunes se encuentran los siguientes:

- Paso de mensajes: En este modelo se establece como base de comunicación entre los programas distribuidos la creación de mensajes que serán intercambiados entre el emisor y el receptor.
- Invocación remota de funciones: En este modelo se establece como base de comunicación entre los programas distribuidos la llamada remota de funciones, procedimientos o métodos que se encuentran implementados en un paradigma cliente-servidor, principalmente.
- Memoria compartida distribuida: En este modelo se establece como base de comunicación entre los programas distribuidos el acceso a una o varias regiones de memoria compartida existente en una o varias computadoras.

Considerando que en los sistemas distribuidos actualmente los modelos de distribución más utilizados son: el paso de mensajes y la invocación remota de funciones, en el presente trabajo éstos serán los modelos que se considerarán.

5.2.1. Comunicación en los sistemas distribuidos

Existen diferentes mecanismos de comunicación en los sistemas distribuidos, éstos dependen del tipo de modelo de distribución. Para el presente trabajo los mecanismos de comunicación considerados de acuerdo a los modelos de distribución son:

- Comunicación por flujo de bytes: Es el mecanismo más básico en los sistemas distribuidos; se basa en la transmisión de unidades de información básicas llamadas bytes entre un emisor y un receptor.

- Comunicación por mensajes: Este mecanismo consiste en el intercambio de unidades de información llamadas mensajes entre un emisor y un receptor. El intercambio de mensajes puede ser de manera directa o indirecta (a través de intermediarios), pudiéndose utilizar una red de comunicación de datos. El intercambio de mensajes puede estar implementado a través de protocolos de comunicación que definen la estructura de los mensajes. Esta estructura usualmente consiste en dos partes: un encabezado (que normalmente contiene información sobre el protocolo utilizado) y un contenido (que normalmente contiene la información transmitida por el mensaje). Este mecanismo se basa en el mecanismo de comunicación por flujo de bytes.

- Comunicación por invocación remota de funciones: Este mecanismo permite la llamada de funciones, procedimientos o métodos remotos implementados como servicios dentro de un programa servidor. Su funcionamiento está basado en el intercambio recíproco de mensajes entre quien llama a la función, procedimiento o método y quien atiende a esa función, procedimiento o método, obteniéndose un resultado de la llamada realizada. Este método se basa en el mecanismo de comunicación por mensajes.

5.3. Especificación de sistemas distribuidos mediante LeGESD

El modelo de especificación de sistemas distribuidos que se propone en este trabajo considera dos aspectos relevantes:

- Abstracción del sistema: En este aspecto se considera abstraer el comportamiento del sistema distribuido de su implementación, enfatizando en determinar los detalles generales relevantes del sistema; además se visualiza al sistema distribuido como una serie de abstracciones de su comportamiento.

- Especificación del sistema: En este aspecto se considera la construcción del modelo del sistema distribuido resultante de las abstracciones observadas del comportamiento del sistema. Estas abstracciones observadas del comportamiento serán directamente plasmadas en el lenguaje gráfico de especificación (LeGESD) propuesto en la presente tesis.

Del modelo propuesto anteriormente, es importante indicar los criterios principales de especificación que son considerados en LeGESD.

5.3.1. Criterios de especificación considerados en LeGESD

Los criterios de especificación que se proponen considerar en LeGESD para su uso como lenguaje de especificación de sistemas distribuidos son:

- La adecuada descripción del comportamiento de los programas distribuidos que integran al sistema. Este criterio se cumple a través de la definición en LeGESD de una simbología gráfica y reglas de construcción (sintaxis del lenguaje) que permite describir con un nivel de detalle suficiente, el comportamiento del sistema distribuido.

- El nivel de abstracción del lenguaje. Este criterio se cumple mediante las estructuras gráficas que LeGESD permite construir, de acuerdo a las distintas abstracciones observadas del comportamiento del sistema distribuido a especificar, teniéndose dos posibles niveles de abstracción: nivel de comunicación y nivel de comportamiento interno.

- Una estructura jerárquica. Este criterio se cumple a través de la característica de especificación jerárquica con la que cuenta LeGESD. Esta especificación jerárquica permite estructurar la especificación del sistema distribuido desde el comportamiento más general hasta llegar a la especificación del comportamiento más particular.

- El nivel de formalismo de la especificación. Este criterio se cumple mediante la incorporación del álgebra de procesos a LeGESD. Esta álgebra de procesos define la semántica del lenguaje, la cual se explicará en el siguiente capítulo.

5.3.2. Especificación de la comunicación en sistemas distribuidos considerada en LeGESD

En el apartado 5.2.1 se revisaron los diferentes mecanismos de comunicación en los sistemas distribuidos que se consideran en la presente tesis. Para estos mecanismos se ha determinado una especificación de la comunicación en LeGESD consistente en:

- Una inicialización de la comunicación. Esta inicialización es común a todos los mecanismos de comunicación, y consiste básicamente de tres pasos: una apertura del mecanismo de comunicación, un enlace del mecanismo de comunicación y el comienzo de la comunicación. En la apertura del mecanismo de comunicación se establecen aspectos relacionados con la identificación del medio a utilizar para la comunicación, la ubicación del destino de la comunicación, el puerto lógico utilizado en el destino de la comunicación, el tipo de comunicación a utilizar, el modo de conexión a establecer y el mecanismo de comunicación que se empleará. En el enlace del mecanismo de comunicación se establecen aspectos relacionados con la referencia del medio a utilizar para la comunicación y condiciones bajo las cuales se llevará a cabo el enlace con el medio. En el comienzo de la comunicación se establecen aspectos relacionados con la referencia del medio a utilizar para la comunicación (la cual debe corresponder a la referencia establecida en el enlace del mecanismo de comunicación), y con los programas distribuidos que harán uso del medio de comunicación para su ejecución.

- La ejecución de la comunicación. Esta ejecución es particular a cada mecanismo de comunicación, y consiste básicamente de dos posibles mecanismos: comunicación por mensajes y comunicación por invocación remota de funciones.

En toda comunicación requerida por los sistemas distribuidos, existen tres aspectos a considerar durante su especificación, estos son:

- El mecanismo de comunicación.

- La estructura del mensaje a transmitir.

- Las restricciones para la comunicación.

LeGESD en su especificación, como se ha mencionado anteriormente, considera los mecanismos de comunicación por mensajes y por invocación remota de funciones. La comunicación por flujo de bytes queda inmersa en la comunicación por mensajes, donde un mensaje se trata como un bloque de bytes.

LeGESD propone una estructura para los mensajes a transmitir consistente de dos partes:

- Un encabezado.
- Un contenido.

En el encabezado se especifica:

- El origen del mensaje.
- El destino del mensaje.
- La referencia del medio de comunicación por el cual se transmitirá el mensaje.

Los valores tanto del origen como del destino del mensaje dependerán del esquema de direccionamiento utilizado por el protocolo de comunicación que se especifique en la apertura del mecanismo de comunicación. El valor de la referencia del medio de comunicación es el que se especificó en el enlace del mecanismo de comunicación.

En el contenido se especifica:

- El tipo de dato que se transmite.
- Los datos a trasmitir.
- El tamaño total de datos a transmitir.

El valor del tipo de dato se establece con algún tipo de dato de los especificados por el pseudocódigo LeGESD (el cual se describe en la sección 5.4.2), el valor de los datos a transmitir depende del tipo de dato establecido, y el tamaño total de datos a transmitir se determina por el número de datos del tipo de dato especificado.

LeGESD establece las siguientes restricciones para la comunicación en sistemas distribuidos que podrá ser especificada con el lenguaje:

- El modelo de comunicación a especificar es exclusivamente del tipo productor-consumidor, es decir siempre que se produzca un mensaje, éste deberá ser consumido. Concretamente, el modelo cliente-servidor se ajusta a este modelo de comunicación.
- La comunicación es de tipo pasivo, es decir, los interesados en comunicarse son los que realizarán la petición para iniciar la comunicación.
- El tipo de comunicación puede ser síncrona o asíncrona.
- La confiabilidad de la comunicación se garantiza en el tipo de comunicación síncrona, no así en el tipo de comunicación asíncrona.

- La comunicación se realiza de manera atómica, es decir la transmisión del mensaje debe realizarse de manera completa sin interrupciones o no se realiza.
- La comunicación está dirigida por las referencias a los medios de comunicación definidas en la especificación.
- El modo de conexión para la comunicación puede ser orientado a conexión y no orientado a conexión.

5.4. Estructura de la especificación de un sistema distribuido en LeGESD (notación gráfica y sintaxis del lenguaje)

La Definición 5.1 establece a un sistema distribuido como un conjunto de entidades lógicas, pero qué se entiende como entidad lógica. La Definición 5.2 conceptualiza lo que se entenderá como entidad lógica.

Definición 5.2. *Una entidad lógica es un componente de software que contiene aspectos dinámicos del comportamiento del sistema y con capacidades de comunicación con otros componentes.*

A partir de esta definición, la especificación de un sistema distribuido se asume como la especificación de cada una de las entidades lógicas que integran al sistema y que interactúan en él, asegurando la correcta interacción entre los programas distribuidos y el ambiente de ejecución. El ambiente de ejecución puede contener una gran variedad de ambientes, los cuales serán compatibles con los programas distribuidos especificados, garantizando con ello la independencia entre estas dos capas dentro de las entidades lógicas.

La estructura que propone LeGESD para la especificación de un sistema distribuido, consiste en la idea de que la arquitectura global de las entidades lógicas que componen al sistema distribuido en su nivel más alto de abstracción está integrada por una:

- Vista de sistema.
- Vista de implementación.

Definición 5.3. *La vista de sistema representa la estructura estática de un sistema distribuido que incluye los elementos estáticos del sistema y sus relaciones estáticas.*

Definición 5.4. *La vista de implementación describe las restricciones en la implementación del sistema distribuido (ambiente de ejecución seleccionado: sistema operativo, lenguaje de programación y tipo de comunicación).*

La vista de implementación consiste en anotaciones textuales dentro de la vista de sistema, mientras que la vista de sistema está integrada por dos tipos de componentes gráficos:

- Componentes de comportamiento, nombrados trabajos LeGESD.
- Componentes de comunicación, nombrados medios LeGESD.

5.4.1. Definición formal de trabajos y medios LeGESD (notación gráfica)

El lenguaje LeGESD se basa en la visión de que un sistema distribuido consiste en dos tipos de componentes en su vista de sistema:

Definición 5.5. *Un componente de comportamiento, llamado trabajo LeGESD o trabajo, es un elemento construido a partir de un conjunto finito de estados que especifica la ejecución del comportamiento del programa distribuido.*

Definición 5.6. *Un componente de comunicación, llamado medio LeGESD o medio, es un elemento construido a partir de un conjunto finito de estados que especifica la comunicación dentro del ambiente de ejecución, y que se sincroniza a partir de la comunicación por mensajes o invocación remota de funciones.*

La descripción de un trabajo es representada por diagramas de comportamiento, mientras que la descripción del medio es representada por diagramas de comunicación. Ambos tipos de diagramas usan un conjunto de elementos gráficos.

Definición 5.7. *Un diagrama de comportamiento es un conjunto de elementos gráficos relacionados en un grafo dirigido utilizado para representar un componente de comportamiento.*

Definición 5.8. *Un diagrama de comunicación es un conjunto de elementos gráficos relacionados en un grafo dirigido utilizado para representar un componente de comunicación.*

Un diagrama de comunicación está integrado por dos diagramas, el diagrama de comunicación del medio de inicialización y el diagrama de comunicación del medio de ejecución.

Definición 5.9. *Un medio de inicialización es un conjunto de elementos gráficos relacionados en un grafo dirigido utilizado para representar la inicialización de la comunicación para un diagrama de comunicación.*

Definición 5.10. *Un medio de ejecución es un conjunto de elementos gráficos relacionados en un grafo dirigido utilizado para representar la ejecución de la comunicación para un diagrama de comunicación.*

LeGESD es un lenguaje gráfico orientado a la especificación de sistemas reactivos, como lo son los sistemas distribuidos. Los sistemas reactivos difieren de los sistemas clásicos en que los primeros interactúan con su exterior para determinar su comportamiento, por lo que requieren un tratamiento diferente para su especificación gráfica. Un elemento esencial en la especificación gráfica de sistemas reactivos es la necesidad de una clara descripción de su comportamiento mediante diversos niveles de diagramas [55] que estén compuestos de símbolos gráficos que cumplan dos características relevantes [56], [57]:

1. Simplicidad en su construcción. Esta simplicidad se puede conseguir utilizando figuras geométricas tales como líneas, cuadrados, rectángulos, círculos, elipses, triángulos, así como combinación de estas figuras.

2. Sencillez en recodar su significado. Esta sencillez se puede lograr si cada símbolo tiene alguna particularidad gráfica que lo identifique con su significado, de forma tal que una persona con familiaridad con el símbolo, al observarlo recuerde y asocie rápidamente su significado.

En LeGESD se definen dos conjuntos de símbolos gráficos principales: estados LeGESD (Figura 5.1a, 5.1b y 5.1d) y enlaces (Figura 5.1c) tanto para trabajos como para medios.

Definición 5.11. *Un estado LeGESD (o sólo estado) es un símbolo gráfico que representa la condición actual de ejecución de un diagrama de comportamiento o de comunicación.*

Definición 5.12. *Un enlace LeGESD (o sólo enlace) es un símbolo gráfico que representa la conectividad entre dos estados dentro de un diagrama de comportamiento o de comunicación.*

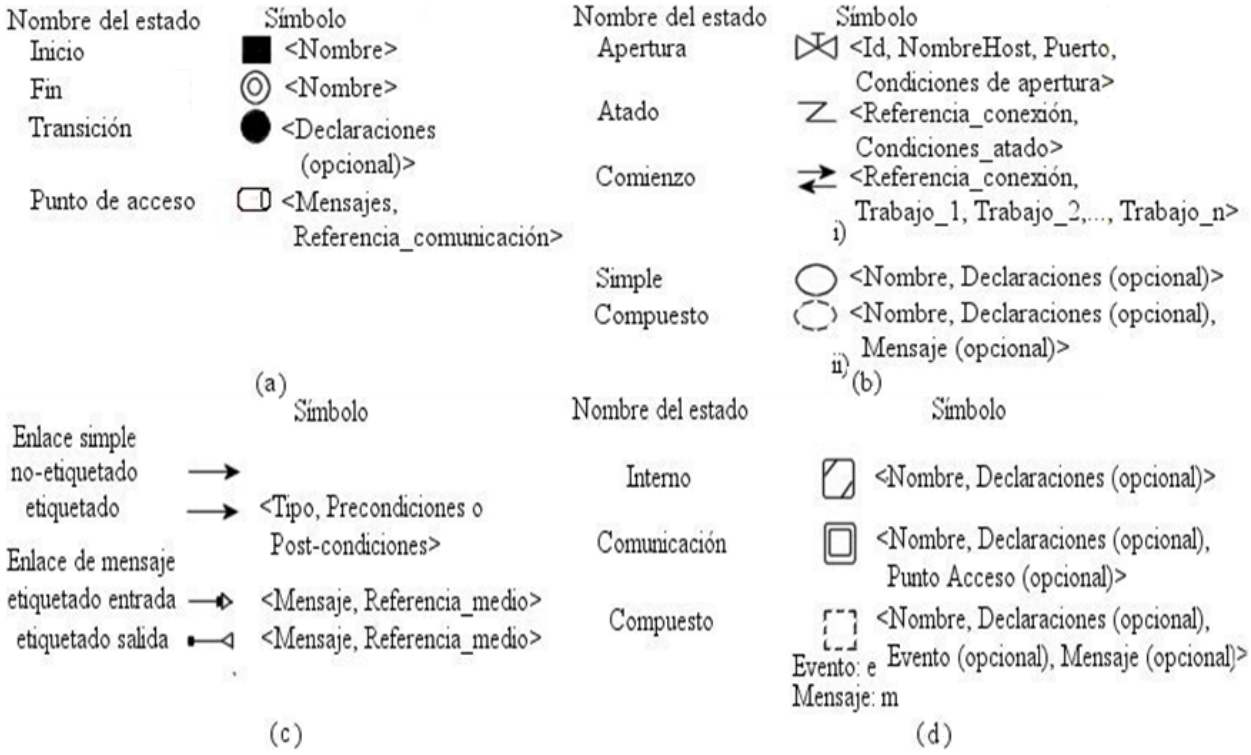


Figura 5.1: (a). Conjunto de símbolos gráficos comunes a los estados de los trabajos y medios LeGESD, (b). i) Conjunto de símbolos de inicialización y ii) conjunto de símbolos de ejecución de los estados de los medios, (c). Conjunto de símbolos de los enlaces de los trabajos y los medios, (d). Conjunto de símbolos de los estados de los trabajos LeGESD.

Los enlaces a su vez, se clasifican en dos tipos: enlace simple y enlace de mensaje.

Definición 5.13. *Un enlace simple es aquel que conecta un estado Transición con cualquier otro estado (o viceversa), que no requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación.*

Definición 5.14. *Un enlace de mensaje es aquel que conecta un estado Punto de acceso con un estado Transición (o viceversa), que requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación.*

De acuerdo a las Definiciones 5.11 y 5.12, los estados y los enlaces son símbolos gráficos que representarán la condición actual de ejecución del sistema distribuido a través de sus diagramas de comportamiento y comunicación. LeGESD considera a estos símbolos gráficos como los elementos sintácticos fundamentales del lenguaje visual propuesto. De esta manera es conveniente en este punto presentar la definición y especificación de cada uno de los elementos sintácticos de LeGESD.

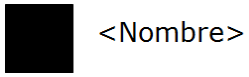


Figura 5.2: Estado Inicio.

Estado *Inicio*: Estado inicial de un diagrama de comportamiento o de comunicación.
Especificación: El estado *Inicio* tiene asociada una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado. La asociación de la etiqueta es obligatoria.
La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, no puede llevar espacios, y debe ser de valor único. La Figura 5.2 muestra el símbolo gráfico del estado *Inicio*.

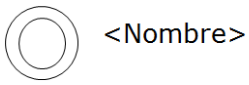


Figura 5.3: Estado Fin.

Estado *Fin*: Estado final de un diagrama de comportamiento o de comunicación.
Especificación: El estado *Fin* tiene asociada una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado. La asociación de la etiqueta es obligatoria.
La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios. La Figura 5.3 muestra el símbolo gráfico del estado *Fin*.

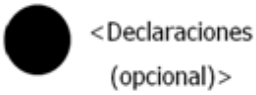


Figura 5.4: Estado Transición.

Estado *Transición*: Estado que controla la transición entre dos posibles estados de un diagrama de comportamiento o de comunicación.
Especificación: El estado *Transición* tiene asociadas declaraciones de control que se validan para permitir o negar la transición de un estado a otro. La asociación de declaraciones de control es opcional.
Las declaraciones de control son variables (si existen) y sus valores correspondientes que serán utilizadas en conjunto con la precondición y/o post-condición para validar la transición. Estas variables son declaradas como variables LeGESD haciendo uso del pseudocódigo LeGESD. La Figura 5.4 muestra el símbolo gráfico del estado *Transición*.

Figura 5.5: Estado Punto de acceso.

Estado *Punto de acceso*: Estado que controla el intercambio de mensajes entre los diagrama de comportamiento o de comunicación.
Especificación: El estado *Punto de acceso* tiene asociados los mensajes que serán comunicados de manera distribuida a los correspondientes estados Punto de acceso de los diagramas de comportamiento y de comunicación, así como también tiene asociada la referencia de comunicación del estado Punto de acceso que complementa la conexión distribuida. La asociación tanto de los mensajes como de la referencia de comunicación es obligatoria.
Los mensajes contienen los datos a transmitir por el medio de comunicación, su estructura consiste en: un encabezado y un contenido.

El encabezado se define con los siguientes elementos:

- Origen. Dirección de red del origen dependiente del esquema de direccionamiento utilizado según el protocolo empleado en el medio de comunicación. Es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos (su formato depende del formato de la dirección de red).
- Destino. Dirección de red del destino dependiente del esquema de direccionamiento utilizado según el protocolo empleado en el medio de comunicación. Es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos (su formato depende del formato de la dirección de red).

El contenido se define con una variable que mantiene el contenido del mensaje. Esta variable es del tipo msj, declarada con el pseudocódigo LeGESD, y está formada por:

- Un tipo del dato (asociado a un tipo de variable LeGESD declarada con el pseudocódigo LeGESD).
- Un valor del dato (asociado al tipo del dato).
- Un tamaño del dato (asociado a un valor numérico).

La referencia de comunicación contiene la referencia del estado Punto de acceso al cual se transmitirá el mensaje, es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios. Se especifica como una lista con todas las referencias de los estados Puntos de acceso que intervendrá en la transmisión de mensajes, cada referencia en la lista se separa con comas. La Figura 5.5 muestra el símbolo gráfico del estado *Punto de acceso*.

Los anteriores cuatro estados representan el conjunto de elementos sintácticos de LeGESD comunes tanto a los trabajos como a los medios, es decir, tanto los diagramas de comportamiento como los diagramas de comunicación podrán utilizar estos símbolos durante su construcción.

Figura 5.6: Estado Interno.

Estado *Interno*: Estado particular de un *trabajo* que representa el nivel mínimo de composición jerárquica de la condición actual de ejecución (acciones a realizar) de un diagrama de comportamiento.
Especificación: El estado *Interno* tiene asociada tanto una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado como declaraciones de acciones que deberán realizarse en el estado. La asociación de la etiqueta es obligatoria mientras que la asociación de declaraciones de acciones a realizar es opcional.

La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios.

Las declaraciones de acciones definen una serie de operaciones a realizar dentro del estado, y se escriben en pseudocódigo LeGESD. La Figura 5.6 muestra el símbolo gráfico del estado *Interno*.

Figura 5.7: Estado Comunicación.

Estado *Comunicación*: Estado particular de un trabajo que representa un nivel de composición jerárquica de la condición actual de ejecución de un diagrama de comportamiento que involucra acciones de comunicación distribuida.
Especificación: El estado *Comunicación* tiene asociada: una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado, declaraciones de acciones de comunicación que deberán realizarse en el estado, y la referencia al estado Punto de acceso a través del cual se realizará la conexión remota. La asociación de la etiqueta es obligatoria mientras que la asociación de declaraciones de acciones de comunicación así como de la referencia al estado Punto de acceso es opcional.

La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios.

Las declaraciones de acciones de comunicación establecen el tipo de mecanismo de comunicación a utilizar, se especifican con la constante LeGESD CM para el tipo de mecanismo de comunicación por mensajes, o con la constante LeGESD CIF para el tipo de mecanismo de comunicación por innovación remota de funciones (estas constantes están definidas en el pseudocódigo LeGESD).

La referencia (o referencias) al estado Punto de acceso es el nombre de la etiqueta que sirve como referencia del estado Punto de acceso al que se asocia el estado Comunicación; es una etiqueta que

tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios. La Figura 5.7 muestra el símbolo gráfico del estado *Comunicación*.

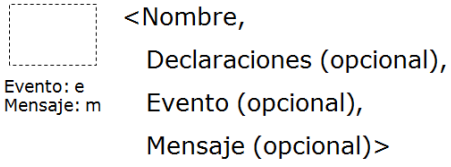


Figura 5.8: Estado Compuesto.

Estado *Compuesto*: Estado particular de un trabajo que representa un nivel de composición jerárquica de la condición actual de ejecución de un diagrama de comportamiento. Dentro del símbolo gráfico pueden incluirse otros estados para representar un nivel de composición jerárquica.
Especificación: El estado *Compuesto* tiene asociada: una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado, declaraciones de acciones que deberán realizarse en el estado, algún evento interno al estado a considerar, y algún mensaje interno a manejar en el estado. La asociación de la etiqueta es obligatoria mientras que la asociación de declaraciones de acciones, el evento y el mensaje interno, es opcional.

La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios.

Las declaraciones de acciones son variables (si existen) y sus valores correspondientes a utilizar en el nivel inferior de la composición jerárquica de la especificación. Estas variables se declaran como tipos de variables LeGESD (definidas en el pseudocódigo LeGESD).

Los eventos son acciones que se presentan asociadas al estado *Compuesto*, se definen con una variable que mantiene información relacionada con el evento, esta variable es del tipo evt, declarada con el pseudocódigo LeGESD, que contiene:

- Tipo del evento (asociado a una constante LeGESD RATÓN, TECLADO, SEMÁFORO, TUBERÍA, MEMCOMPARTIDA, ARCHIVO o BASEDATOS. Estas constantes están definidas en el pseudocódigo LeGESD).
- Los datos del evento (asociados a una variable que mantiene información concreta de acuerdo al tipo de evento generado). Esta variable, declarada con el pseudocódigo LeGESD, puede ser del tipo:
  - eraton. Contiene como miembros internos: un identificador numérico y el valor del evento.
  - eteclado. Contiene como miembros internos: un identificador numérico y el valor del evento.
  - esemaforo. Contiene como miembros internos: una clave numérica y un identificador numérico.

- etuberia. Contiene como miembros internos: un identificador numérico, un descriptor numérico para lectura y un descriptor numérico para escritura.
- ememcompartida. Contiene como miembros internos: una clave numérica y un identificador numérico.
- earchivo. Contiene como miembros internos: un identificador numérico y un nombre alfanumérico de 50 caracteres.
- ebasedatos. Contiene como miembros internos: un identificador numérico y un nombre alfanumérico de 50 caracteres.

Los mensajes contienen los datos a transmitir por la tubería, su estructura consiste de un encabezado y un contenido. El encabezado se define con los siguientes elementos:

- Origen. Asociado con el descriptor numérico de escritura del tipo de variable etuberia.
- Destino. Asociado con el descriptor numérico de lectura del tipo de variable etuberia.

El contenido se define con una variable que mantiene el contenido del mensaje, esta variable es del tipo msj, declarada con el pseudocódigo LeGESD, que contiene:

- Un tipo del dato (asociado a un tipo de variable LeGESD declarada con el pseudocódigo LeGESD).
- Un valor del dato (asociado al tipo del dato).
- Un tamaño del dato (asociado a un valor numérico).

La Figura 5.8 muestra el símbolo gráfico del estado *Compuesto*.

Los anteriores tres estados representan el conjunto de elementos sintácticos de LeGESD particulares de los trabajos, es decir únicamente los diagramas de comportamiento podrán utilizar estos símbolos durante su construcción.

Figura 5.9: Estado Apertura.

Estado *Apertura*: Estado particular de un medio que representa la identificación del medio de comunicación a utilizar en un diagrama de comunicación.
Especificación: El estado *Apertura* tiene asociado: un identificador que deberá corresponder con un valor que se le proporcione al estado, la dirección de la computadora (host) que fungirá como destino (servidor) en la conexión distribuida, el número de puerto lógico utilizado en la computadora destino para propósitos de comunicación de los programas distribuidos, y las condiciones de inicio en la apertura del medio de comunicación. La asociación del identificador, la dirección del host, el puerto y las condiciones de apertura, es obligatoria.

El identificador se declara como un valor numérico entre 1 y 256.

La dirección del host se declara como la dirección de red del destino (servidor) dependiente del esquema de direccionamiento utilizado según el protocolo empleado en el medio de comunicación, es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos (su formato depende del formato de la dirección de red).

El puerto se declara como un valor numérico entre 1 y 65536 inclusive.

Las condiciones de apertura declaran:

- El tipo de comunicación, a través de las constantes LeGESD SINC para especificar una comunicación síncrona o ASINC para especificar una comunicación asíncrona.
- El modo de conexión, a través de las constantes LeGESD OC para especificar una comunicación orientada a conexión o NOC para especificar una comunicación no orientada a conexión.

Estas constantes están definidas en el pseudocódigo LeGESD. La Figura 5.9 muestra el símbolo gráfico del estado *Apertura*.

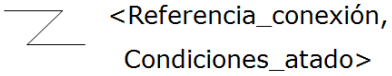


Figura 5.10: Estado Atado.

Estado *Atado*: Estado particular de un medio que representa el enlace con el medio de comunicación a utilizar en un diagrama de comunicación.
Especificación: El estado *Atado* tiene asociada tanto la referencia de conexión que identifica la conexión distribuida a construir como las condiciones bajo las cuales se llevará a cabo el enlace con el medio de comunicación. La asociación de la referencia de conexión como las condiciones del enlace es obligatoria.

La referencia de conexión contiene la referencia del estado Punto de acceso que intervendrá en la transmisión del mensaje, es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios, debe ser idéntica a la referencia del estado Punto de acceso que intervendrá en la transmisión del mensaje. Se especifica como una lista con todas las referencias de los estados Puntos de acceso que intervendrán en la transmisión de mensajes. Cada referencia en la lista se separa con comas.

Las condiciones del atado se declaran a través de una variable tipo atd, definida en el pseudocódigo LeGESD, que contiene: dirección de red destino (asociada con la dirección establecida en el estado Apertura) y puerto (asociado al puerto establecido en el estado Apertura). La Figura 5.10 muestra el símbolo gráfico del estado *Atado*.

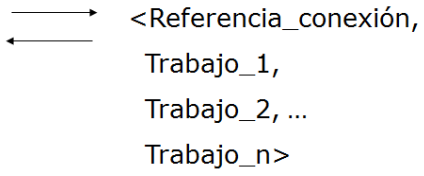


Figura 5.11: Estado Comienzo.

Estado *Comienzo*: Estado particular de un medio que genera la relación de los distintos trabajos que se conectarán en el diagrama de comunicación.
Especificación: El estado *Comienzo* tiene asociada tanto la referencia de conexión que identifica la conexión distribuida generada en el estado Atado como la lista de todos los trabajos que se conectarán a través del medio para llevar a cabo la comunicación distribuida. La asociación de la referencia de conexión como la lista de trabajos es obligatoria.

La referencia de conexión contiene la referencia del estado Punto de acceso que intervendrá en la transmisión del mensaje. Es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios, debe ser idéntica a la referencia del estado Punto de acceso que intervendrá en la transmisión del mensaje. Se especifica como una lista con todas las referencias de los estados Puntos de acceso que intervendrán en la transmisión de mensajes. Cada referencia en la lista se separa con comas. Esta lista debe ser la misma a la especificada en el estado Atado.

La lista de trabajos se especifica con las etiquetas de los nombres de los trabajos que se comunicarán a través del medio de comunicación. Cada nombre de trabajo en la lista se separa con comas. La Figura 5.11 muestra el símbolo gráfico del estado *Comienzo*.

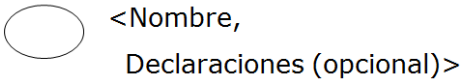


Figura 5.12: Estado Simple.

Estado *Simple*: Estado particular de un medio que representa el mecanismo de comunicación por mensajes de un diagrama de comunicación. Se considera idéntico al estado *Comunicación*.
Especificación: El estado *Simple* tiene asociada tanto una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado como declaraciones de acciones que deberán realizarse en el estado. La asociación de la etiqueta es obligatoria mientras que la asociación de declaraciones de acciones es opcional.

La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios.

Las declaraciones de acciones definen una serie de operaciones a realizar dentro del estado, y se escriben en pseudocódigo LeGESD. La Figura 5.12 muestra el símbolo gráfico del estado *Simple*.

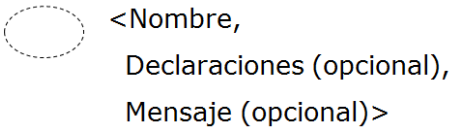


Figura 5.13: Estado Compuesto.

Estado *Compuesto*: Estado particular de un medio que representa el mecanismo de comunicación por innovación remota de funciones de un diagrama de comunicación. Se considera idéntico al estado *Comunicación*.
Especificación: El estado *Compuesto* tiene asociada: una etiqueta que deberá corresponder con un nombre simbólico que se le proporcione al estado, declaraciones para la innovación remota de la función que deberá realizarse en el estado, y mensajes internos a manejar en el estado. La asociación de la etiqueta es obligatoria mientras que la asociación de declaraciones de acciones de comunicación y los mensajes internos, es opcional.

La etiqueta tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, no puede llevar espacios, y debe ser de valor único.

Las declaraciones para la invocación remota de la función se especifican en pseudocódigo LeGESD, declarando el prototipo de la función a invocar remotamente.

Los mensajes internos contienen datos adicionales a transmitir por el medio de comunicación, su estructura consiste de: un encabezado y un contenido. El encabezado se define con los siguientes elementos:

- Origen. Dirección de red del origen dependiente del esquema de direccionamiento utilizado según el protocolo empleado en el medio de comunicación. Es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos (su formato depende del formato de la dirección de red).
- Destino. Dirección de red del destino dependiente del esquema de direccionamiento utilizado según el protocolo empleado en el medio de comunicación. Es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos (su formato depende del formato de la dirección de red).

El contenido se define con una variable que mantiene el contenido del mensaje, esta variable es del tipo msj, declarada con el pseudocódigo LeGESD, y está formada por:

- Un tipo del dato (asociado a un tipo de variable LeGESD declarada con el pseudocódigo LeGESD).
- Un valor del dato (asociado al tipo del dato).
- Un tamaño del dato (asociado a un valor numérico).

La Figura 5.13 muestra el símbolo gráfico del estado *Compuesto*.

Los anteriores cinco estados representan el conjunto de elementos sintácticos de LeGESD particulares de los medios, es decir únicamente los diagramas de comunicación podrán utilizar estos

símbolos durante su construcción. A su vez, los estados Apertura, Atado y Comienzo representan los elementos sintácticos del medio de inicialización (inicialización de la comunicación), mientras que los estados Simple y Compuesto representan los elementos sintácticos del medio de ejecución (ejecución de la comunicación).

Figura 5.14: Enlace simple no-etiquetado.

Enlace simple *no-etiquetado*: Enlace que conecta un estado Transición con cualquier otro estado (o viceversa), sin especificar la existencia de una precondición o post-condición dentro de un diagrama de comportamiento o de comunicación.
Especificación: El enlace simple *no-etiquetado* se asocia únicamente con estados que no requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación. La Figura 5.14 muestra el símbolo gráfico del enlace simple *no-etiquetado*.

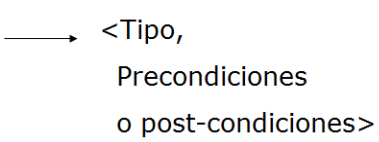


Figura 5.15: Enlace simple etiquetado.

Enlace simple *etiquetado*: Enlace que conecta un estado Transición con cualquier otro estado (o viceversa), especificando la existencia de una precondición o post-condición dentro de un diagrama de comportamiento o de comunicación.
Especificación: El enlace simple *etiquetado* se utiliza únicamente con estados que no requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación. El enlace simple *etiquetado* tiene asociado tanto el tipo de evento que se genera (1 para precondición, 2 para post-condición) como la precondición o post-condición a especificar (separadas por comas, y que pueden contener declaración, asignación y comparación de variables y constantes). La asociación del tipo de evento así como la especificación (declaración e inicialización) de la precondición o post-condición es obligatoria.

Una precondición es una condición que deberá ser verdadera antes de realizarse un cambio de estado, una post-condición es una condición que se generará como parte del cambio de estado, y deberá ser también verdadera. La verificación de las precondiciones así como la generación de las post-condiciones se realizará en el estado Transición. Las precondiciones y post-condiciones se especifican en pseudocódigo LeGESD. La Figura 5.15 muestra el símbolo gráfico del enlace simple *etiquetado*.

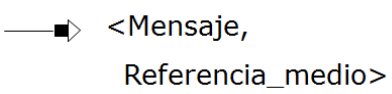


Figura 5.16: Enlace de mensaje etiquetado-entrada.

Enlace de mensaje *etiquetado-entrada*: Enlace que conecta un estado Punto de acceso con un estado Comunicación o Simple o Compuesto del medio (mediante un estado Transición), especificando la existencia de mensajes de entrada dentro de un diagrama de comportamiento o de comunicación.

Especificación: El enlace de mensaje *etiquetado-entrada* se utiliza únicamente con estados que requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación. El enlace de mensaje *etiquetado-entrada* tiene asociado tanto el mensaje de entrada (a recibir) proveniente del medio de comunicación como la referencia del medio de comunicación de donde proviene el mensaje. La asociación del mensaje de entrada así como la referencia del medio de comunicación es obligatoria.

La estructura del mensaje de entrada consiste de: un encabezado y un contenido. El encabezado se define con los siguientes elementos:

- Origen. Referencia del estado Punto de acceso origen.
- Destino. Referencia del estado Punto de acceso destino.

El contenido se define con una variable que mantiene el contenido del mensaje, esta variable es del tipo msj, declarada con el pseudocódigo LeGESD, y está formada de:

- Un tipo del dato (asociado a un tipo de variable LeGESD declarada con el pseudocódigo LeGESD).
- Un valor del dato (asociado al tipo del dato).
- Un tamaño del dato (asociado a un valor numérico).

La referencia de comunicación contiene la referencia del estado Punto de acceso del cual se espera sea transmitido el mensaje, es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios. La Figura 5.16 muestra el símbolo gráfico del enlace de mensaje *etiquetado-entrada*.

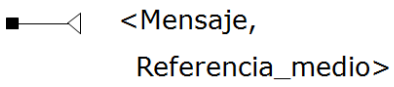


Figura 5.17: Enlace de mensaje etiquetado-salida.

Enlace de mensaje *etiquetado-salida*: Enlace que conecta un estado Comunicación o Simple o Compuesto del medio (mediante un estado Transición) a un estado Punto de acceso especificando los mensajes de salida dentro de un diagrama de comportamiento o de comunicación.
Especificación: El enlace de mensaje *etiquetado-salida* se utiliza únicamente con estados que requieren un intercambio de mensajes dentro de un diagrama de comportamiento o de comunicación. El enlace de mensaje *etiquetado-salida* tiene asociado tanto el mensaje de salida (a enviar) hacia el medio de comunicación como la referencia del medio de comunicación por donde se enviará el mensaje. La asociación del mensaje de salida así como la referencia del medio de comunicación es obligatoria.

La estructura del mensaje de salida consiste de: un encabezado y un contenido. El encabezado se define con los siguientes elementos:

- Origen. Referencia del estado Punto de acceso origen.
- Destino. Referencia del estado Punto de acceso destino.

El contenido se define con una variable que mantiene el contenido del mensaje, esta variable es del tipo msj, declarada con el pseudocódigo LeGESD, y está formada de:

- Un tipo del dato (asociado a un tipo de variable LeGESD declarada con el pseudocódigo LeGESD).
- Un valor del dato (asociado al tipo del dato).
- Un tamaño del dato (asociado a un valor numérico).

La referencia de comunicación contiene la referencia del estado Punto de acceso al cual se espera transmitir el mensaje, es una etiqueta que tiene como longitud máxima 50 caracteres alfanuméricos, no puede iniciar con algún número o carácter especial, y no puede llevar espacios. La Figura 5.17 muestra el símbolo gráfico del enlace de mensaje *etiquetado-salida*.

Los enlaces anteriores son comunes tanto para los trabajos como para los medios, es decir tanto los diagramas de comportamiento como los diagramas de comunicación podrán utilizar estos símbolos durante su construcción.

Una vez que ha sido presentada la simbología del lenguaje y se han definido lo que son trabajos y medios, ahora es posible definir de manera más precisa tanto a los trabajos como a los medios que son los componentes principales del lenguaje gráfico LeGESD, y que representan los elementos esenciales a utilizar durante la especificación del sistema distribuido.

Definición 5.15. *Un componente de comportamiento, llamado trabajo LeGESD o trabajo, 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 un conjunto de post-condiciones, M es un conjunto de mensajes, y R es un*

conjunto 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 $L \subseteq (S \times l \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (\aleph \times Pr) \cup (\aleph \times Po) \cup (M \times R)$, donde ε denota a la etiqueta vacía.

Debido a que los trabajos LeGESD tienen que cumplir con el criterio de especificación de ser jerárquicos, el conjunto de estados *S* de un diagrama de comportamiento 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,

Definición 5.16. *La función jerárquica ψ se define como $\psi(s) = \{J_1, \ldots, J_k\}$ si los trabajos LeGESD $J_1, \ldots, J_k$ están contenidos dentro del estado s. Dada la función jerárquica ψ, si $\psi(s) = 0$ entonces un estado s es del tipo de uno de los siguientes estados posibles: Inicio, Interno, Punto de acceso, Final, o Transición; si $\psi(s) = \{J_1, \ldots, J_k\}$ entonces un estado s es un estado Compuesto o un estado Comunicación.*

De forma similar, el conjunto de referencias de medios *R* está definido por una *función compuesta* Λ, la cual está definida (para cada medio) por los dos tipos de medios LeGESD que contiene. Esto es,

Definición 5.17. *La función compuesta Λ se define como $\Lambda(r) = \{I_m, E_m\}$ si el medio de inicialización I_m y el medio de ejecución E_m están contenidos en el medio r.*

Definición 5.18. *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 un conjunto de post-condiciones, y No es un conjunto 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 $L \subseteq (S \times l \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (\aleph \times Pr) \cup (\aleph \times Po) \cup (No)$ 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.*

Definición 5.19. *Un medio de ejecución 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 $L \subseteq (S \times l \times S)$ está definido sobre el conjunto de etiquetas $L \subseteq (S \times l \times S)$ está definido sobre el conjunto de etiquetas $l \subseteq \{\varepsilon\} \cup (\aleph \times Pr) \cup (\aleph \times Po) \cup (M \times No)$ donde ε denota a la etiqueta vacía.*

Debido a que E_m tiene que cumplir con el criterio de especificación de ser jerárquico, el conjunto de estados *S* está definido a través de una *función jerárquica* Ω, la cual está definida (para cada estado) por el conjunto de medios de ejecución que contiene. Esto es,

Definición 5.20. *La función jerárquica Ω se define como $\Omega(s) = \{E_{m1}, \ldots, E_{mk}\}$ si los medios de ejecución $E_{m1}, \ldots, E_{mk}$ están contenidos dentro del estado s. Dada la función jerárquica Ω, si $\Omega(s) = 0$ entonces un estado s es uno de los siguientes estados posibles: Inicio, Simple, Punto de acceso, Final, o Transición; si $\Omega(s) = \{E_{m1}, \ldots, E_{mk}\}$ entonces un estado s es un estado Compuesto.*

5.4.2. Pseudocódigo LeGESD

El pseudocódigo LeGESD está orientado a facilitar la inclusión de variables, constantes, y operaciones que complementen la descripción de los elementos sintácticos del lenguaje visual utilizados para la especificación de un sistema distribuido. El pseudocódigo LeGESD está compuesto por los siguientes elementos:

- Palabras reservadas.
- Tipos de datos.
- Constantes.
- Operadores y expresiones.
- Declaración de variables.

Las palabras reservadas que utiliza el pseudocódigo LeGESD se resumen en la Tabla 5.1.

Los tipos de datos definidos en el pseudocódigo LeGESD son idénticos a los definidos en el lenguaje C [58], es decir se tienen los siguientes tipos:

- Tipos fundamentales. int, char, float, double, void, modificadores como const, short y long, y apuntadores (*).
- Tipos derivados. arreglo ([ ]) y struct.

Adicionalmente, el pseudocódigo LeGESD define los siguientes tipos de datos:

- Tipo mensaje (msj). Es un tipo de dato derivado utilizado para declarar una variable mensaje, contiene como miembros:
  - void tDato (Tipo de dato, void = int, char, float, double, arreglo, struct, y *).
  - void vDato (Valor del dato, void = según el tipo del dato especificado).
  - int tmDato (Tamaño del dato).
- Tipo atado (atd). Es un tipo de dato derivado utilizado para declarar una variable de atado de la comunicación distribuida, contiene como miembros:
  - char *dRed (Dirección de red).
  - int nPuerto (Número del puerto).
- Tipo evento (evt). Es un tipo de dato derivado utilizado para declarar una variable evento, contiene como miembros:
  - int tEvento (Tipo de evento).
  - void dEvento (Datos del evento, void = eraton, eteclado, esemaforo, etuberia, ememcompartida, earchivo o ebasedatos).

- Tipo evento de ratón (eraton). Es un tipo de dato derivado utilizado para declarar una variable evento de ratón, contiene como miembros:

  - int iEvento (Identificador del evento).
  - void vEvento (Valor del evento, void = int, char, arreglo o struct).

- Tipo evento de teclado (eteclado). Es un tipo de dato derivado utilizado para declarar una variable evento de teclado, contiene como miembros:

  - int iEvento (Identificador del evento).
  - void vEvento (Valor del evento, void = int, char, arreglo o struct).

- Tipo evento de semáforo (esemaforo). Es un tipo de dato derivado utilizado para declarar una variable evento de semáforo, contiene como miembros:

  - int iEvento (Identificador del evento).
  - int cSemaforo (Clave del semáforo).

- Tipo evento de tubería (etuberia). Es un tipo de dato derivado utilizado para declarar una variable evento de tubería, contiene como miembros:

  - int iEvento (Identificador del evento).
  - int iLectura (Descriptor lectura de la tubería).
  - int iEscritura (Descriptor escritura en la tubería).

- Tipo evento de memoria compartida (ememcompartida). Es un tipo de dato derivado utilizado para declarar una variable evento de memoria compartida, contiene como miembros:

  - int iEvento (Identificador del evento).
  - int cMemoria (Clave de la memoria compartida).

- Tipo evento de archivo (earchivo). Es un tipo de dato derivado utilizado para declarar una variable evento de archivo, contiene como miembros:

  - int iEvento (Identificador del evento).
  - char nArchivo[50] (Nombre del archivo).

- Tipo evento de base de datos (ebasedatos). Es un tipo de dato derivado utilizado para declarar una variable evento de base de datos, contiene como miembros:

  - int iEvento (Identificador del evento).
  - char nBaseDatos[50] (Nombre de la base de datos).

Las constantes definidas en el pseudocódigo LeGESD son las siguientes:

- ARCHIVO. Identifica un tipo de evento de archivo.
- ASINC. Identifica un tipo de comunicación asíncrona.
- BASEDATOS. Identifica un tipo de evento de base de datos.
- CIF. Identifica un mecanismo de comunicación por invocación remota de función.
- CM. Identifica un mecanismo de comunicación por mensajes.
- MEMCOMPARTIDA. Identifica un tipo de evento de memoria compartida.
- NOC. Identifica un modo de conexión no orientado a conexión.
- OC. Identifica un modo de conexión orientado a conexión.
- RATON. Identifica un tipo de evento de ratón.
- SEMAFORO. Identifica un tipo de evento de semáforo.
- SINC. Identifica un tipo de comunicación síncrona.
- TECLADO. Identifica un tipo de evento de teclado.
- TUBERIA. Identifica un tipo de evento de tubería.

Palabra reservada	Definición	Sintaxis
Ciclo	Declaración de un ciclo a ejecutar.	Ciclo *condición : expresiones;* |
Si	Declaración de una bifurcación a ejecutar.	Si *condición : expresiones;* |
Tipo Const()	Declaración de variables en la vista de implementación que se consideran invariantes a lo largo de la ejecución del sistema.	Tipo Const(*variable*)
Valor	Asignación de un valor a una variable Tipo Const().	Tipo Const(*variable*) Valor *valor*;
Nom_Modelo	Declaración en la vista de implementación para nombrar el sistema distribuido a especificar.	Nom_Modelo: *nombre*;
Autor	Declaración en la vista de implementación para nombrar al autor de la especificación.	Autor: *nombre*;
Versión	Declaración en la vista de implementación para indicar el número de versión de la especificación.	Versión: *número*;
Instancias_Clases:Inicio...	Declaración en la vista de implementación para indicar el inicio de la definición de los componentes globales que integrarán al sistema distribuido a especificar.	Instancias_Clases:Inicio... *componentes*;
Instancias_Clases:Fin...	Declaración en la vista de implementación para indicar el final de la definición de los componentes globales que integrarán al sistema distribuido a especificar.	Instancias_Clases:Inicio... *componentes*; Instancias_Clases:Fin...
Usa_IDC()	Declaración en la vista de implementación dentro de Instancias_Clases:Inicio... e Instancias_Clases:Fin... para definir el identificador de un componente global que integrará al sistema distribuido a especificar.	*componente* Usa_IDC(*identificador*);
Ambiente_Ejec:Inicio...	Declaración en la vista de implementación para indicar el inicio de la definición de los elementos del ambiente de ejecución que integrarán al sistema distribuido a especificar.	Ambiente_Ejec:Inicio... *elementos*;
Ambiente_Ejec:Fin...	Declaración en la vista de implementación para indicar el final de la definición de los elementos del ambiente de ejecución que integrarán al sistema distribuido a especificar.	Ambiente_Ejec:Inicio... *elementos*; Ambiente_Ejec:Fin...
Sistema_Oper	Declaración en la vista de implementación dentro de Ambiente_Ejec:Inicio... y Ambiente_Ejec:Fin... para definir el sistema operativo a utilizar en la implementación del sistema distribuido a especificar.	Sistema_Oper: *nombre*;
Lenguaje_Prog	Declaración en la vista de implementación dentro de Ambiente_Ejec:Inicio... y Ambiente_Ejec:Fin... para definir el lenguaje de programación a utilizar en la implementación del sistema distribuido a especificar.	Lenguaje_Prog: *nombre*;
Tipo_Comunicación	Declaración en la vista de implementación dentro de Ambiente_Ejec:Inicio... y Ambiente_Ejec:Fin... para definir el tipo de comunicación a utilizar en la implementación del sistema distribuido a especificar.	Tipo_Comunicación: *nombre*;

Tabla 5.1: Palabras reservadas del pseudocódigo LeGESD.

Las expresiones y los operadores definidos en el pseudocódigo LeGESD son idénticos a los definidos en el lenguaje C [58], es decir se tiene lo siguiente:

Una expresión es una combinación de operadores y operandos de cuya evaluación se obtiene un valor. Los operandos pueden ser nombres que denoten variables o constantes, funciones, literales de cualquier tipo adecuado de acuerdo con los operadores u otras expresiones más simples. La evaluación de una expresión da lugar a un valor de algún tipo, una expresión se dice que es del tipo de su resultado.

Un operador es un elemento o carácter encargado de manipular datos que pueden ser números, caracteres, variables o constantes. Existen diversos tipos de operadores en el pseudocódigo LeGESD, los cuales son:

- Operadores aritméticos. Los operadores aritméticos realizan operaciones aritméticas con los operandos que se asocian a los operadores. La Tabla 5.2 muestra los operadores aritméticos utilizados en el pseudocódigo LeGESD.

Operación	Acción
++	Post-incremento o pre-incremento.
–	Post-decremento o pre-decremento.
*	Multiplicación.
/	División.
%	Módulo.
+	Suma.
-	Resta.

Tabla 5.2: Operadores aritméticos utilizados en el pseudocódigo LeGESD.

- Operadores relacionales. Los operadores relacionales comparan sus operandos y devuelven el valor 1 si la relación es cierta, y 0 si no lo es. La Tabla 5.3 muestra los operadores relacionales utilizados en el pseudocódigo LeGESD.

Operador	Propósito
<	Menor que.
<=	Menor o igual que.
>	Mayor que.
>=	Mayor o igual que.
==	Igual.
!=	No igual.

Tabla 5.3: Operadores relacionales utilizados en el pseudocódigo LeGESD.

- Operadores lógicos. Los operadores lógicos comparan sus operandos a valores de falso o

verdadero. La Tabla 5.4 muestra los operadores lógicos utilizados en el pseudocódigo LeGESD.

Operador	Acción
!	Negación lógica.
&&	Y lógico.
\|\| | O lógico. |

Tabla 5.4: Operadores lógicos utilizados en el pseudocódigo LeGESD.

- Operador de asignación. Un operador de asignación altera el valor de una variable sin alterar su tipo. El operador de asignación (=), copia el valor del operando de la derecha en el operando de la izquierda, aplicando las conversiones de tipo cuando es necesario. Es posible utilizar ”casting” para la conversión de tipos de datos de las variables.

La precedencia de los operadores anteriormente descritos en la evaluación de expresiones siguen las mismas reglas que en el lenguaje C.

La declaración de variables en el pseudocódigo LeGESD es idéntica a la declaración que se realiza en el lenguaje C [58], es decir se tiene la siguiente sintaxis:

Tipo Identificador = Valor; donde,

- Identificador: es el nombre para referenciar a la variable.
- Tipo: es el tipo de dato al que la variable pertenece.
- Valor: es el contenido dentro del rango de valores permitidos por el tipo de dato definido.

5.4.3. Definición de la sintaxis de LeGESD

La sintaxis de LeGESD está definida por un conjunto de reglas de construcción, que deberán seguirse para conectar los elementos sintácticos que fueron definidos en el apartado anterior y que intervendrán en la generación de un trabajo o medio LeGESD. Estas reglas de construcción son las siguientes:

- **Regla 1**. Todo trabajo o medio debe comenzar en un estado Inicio y terminar en un estado Fin.

  En la Figura 5.18 se observan ejemplos de la forma correcta en la aplicación de la regla 1, así como la forma incorrecta de su aplicación.

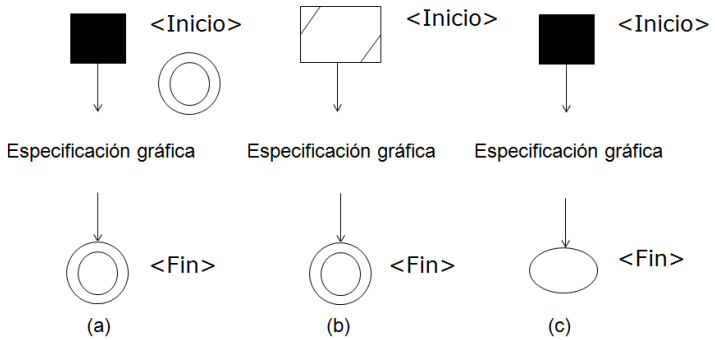


Figura 5.18: (a). Aplicación correcta de la regla 1. (b) y (c). Aplicación incorrecta de la regla 1.

La Figura 5.18(a) muestra la aplicación correcta de la regla 1, en donde existe un estado Inicio iniciando la especificación gráfica y un estado Fin terminando la especificación gráfica. La Figura 5.18(b) muestra una aplicación incorrecta de la regla 1, en donde se está utilizando un estado Interno como inicio de la especificación gráfica aún cuando se utiliza un estado Fin para terminar la especificación. La Figura 5.18(c) muestra otra aplicación incorrecta de la regla 1, en donde se está utilizando un estado Simple como terminación de la especificación gráfica aún cuando se utiliza un estado Inicio para iniciar la especificación.

- **Regla 2**. En un trabajo o medio es posible declarar un sólo estado Inicio y un sólo estado Fin.

  En la Figura 5.19 se observan ejemplos de la forma correcta en la aplicación de la regla 2, así como la forma incorrecta de su aplicación.

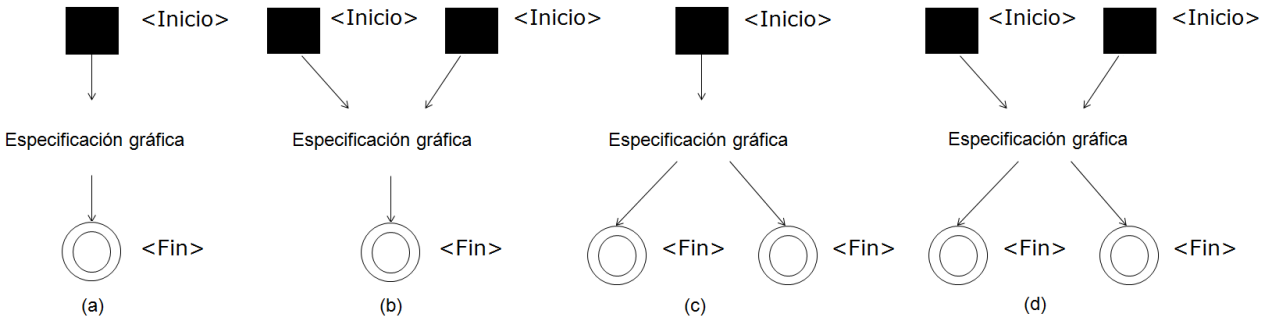


Figura 5.19: (a). Aplicación correcta de la regla 2.
(b), (c) y (d). Aplicación incorrecta de la regla 2.

La Figura 5.19(a) muestra la aplicación correcta de la regla 2, en donde únicamente existen un estado Inicio y un estado Fin delimitando la especificación gráfica. La Figura 5.19(b) muestra una aplicación incorrecta de la regla 2, en donde se están utilizando dos estados Inicio para iniciar la especificación gráfica aún cuando se utiliza un sólo estado Fin para terminar la especificación. La Figura 5.19(c) muestra otra aplicación incorrecta de la regla 2, en donde se están utilizando dos estados Fin como terminación de la especificación gráfica aún cuando se utiliza un sólo estado Inicio para iniciar la especificación. La Figura 5.19(d)

muestra una última aplicación incorrecta de la regla 2, en donde se están utilizando tanto dos estados Inicio para iniciar la especificación gráfica como dos estados Fin como terminación de la especificación gráfica.

- **Regla 3**.Dos estados cualesquiera, excepto el estado Transición, deben estar conectados a través de un estado Transición utilizando enlaces simples o de mensajes.

  En la Figura 5.20 se observan ejemplos de la forma correcta en la aplicación de la regla 3, así como la forma incorrecta de su aplicación.

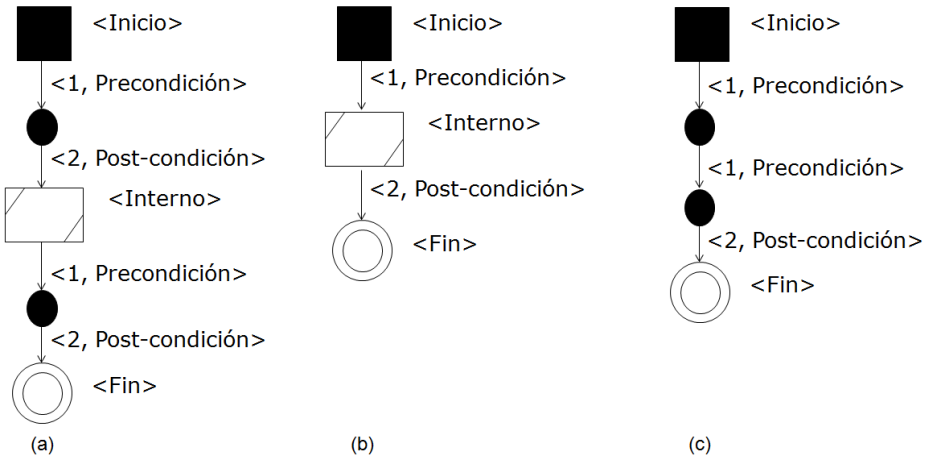


Figura 5.20: (a). Aplicación correcta de la regla 3. (b), y (c). Aplicación incorrecta de la regla 3.

La Figura 5.20(a) muestra la aplicación correcta de la regla 3, en donde los estados Inicio e Interno están conectados a través de un estado Transición usando enlaces simples etiquetados, y los estados Interno y Fin están conectados a través de otro estado Transición usando enlaces simples etiquetados. La Figura 5.20(b) muestra una aplicación incorrecta de la regla 3, en donde los estados Inicio, Interno y Fin están conectados con enlaces simples etiquetados pero sin utilizar estados Transición. La Figura 5.20(c) muestra otra aplicación incorrecta de la regla 3, en donde se están utilizando dos estados Transición conectados entre ellos.

- **Regla 4**. Un estado cualquiera siempre tiene al menos un enlace simple etiquetado entrando hacia él y uno saliendo de él, excepto el estado Inicio que puede tener sólo un enlace simple etiquetado saliendo y el estado Fin que no tiene enlaces simples etiquetados saliendo de él.

  En la Figura 5.21 se observan ejemplos de la forma correcta en la aplicación de la regla 4, así como la forma incorrecta de su aplicación.

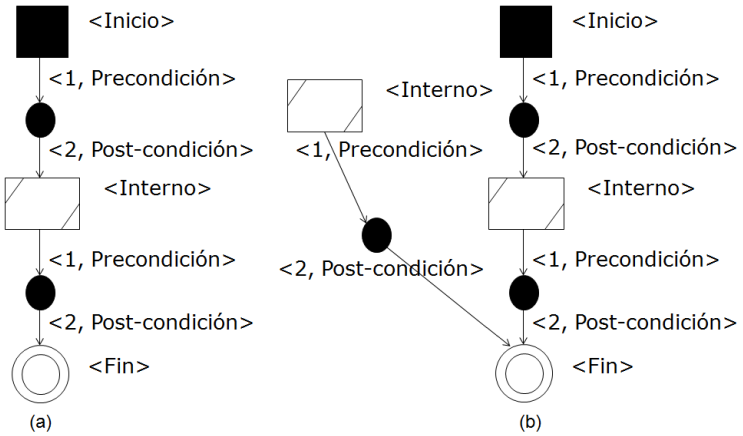


Figura 5.21: (a). Aplicación correcta de la regla 4. (b). Aplicación incorrecta de la regla 4.

La Figura 5.21(a)muestra la aplicación correcta de la regla 4, en donde los estados Inicio, Interno, Transición y Fin están conectados con al menos un enlace simple etiquetado entrando hacia ellos y saliendo de ellos, excepto en el estado Fin donde sólo se tiene un enlace simple etiquetado entrando hacia él, y en el estado Inicio donde sólo se tiene un enlace simple etiquetado saliendo de él. La Figura 5.21(b) muestra una aplicación incorrecta de la regla 4, en donde uno de los estados internos (el de la izquierda) sólo tiene un enlace simple saliendo de él sin que exista un enlace entrando hacia él, lo que implica que ese estado no puede alcanzarse.

- **Regla 5**. La precondición está siempre declarada antes de un estado Transición, y la post-condición está siempre declarada después de un estado Transición.

  En la Figura 5.22 se observan ejemplos de la forma correcta en la aplicación de la regla 5, así como la forma incorrecta de su aplicación.

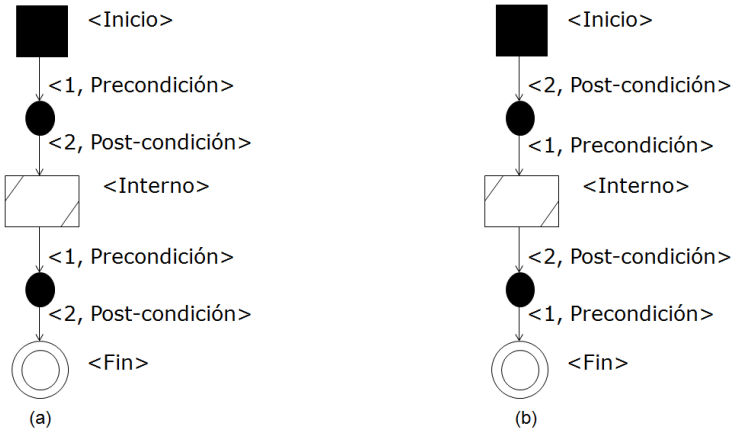


Figura 5.22: (a). Aplicación correcta de la regla 5. (b). Aplicación incorrecta de la regla 5.

La Figura 5.22(a) muestra la aplicación correcta de la regla 5, en donde los estados Transición tienen declaradas las precondiciones antes de entrar a los estados y las post-condiciones

después de salir de los estados. La Figura 5.22(b) muestra una aplicación incorrecta de la regla 5, en donde los estados Transición tienen declaradas las precondiciones después de salir de los estados y las post-condiciones antes de entrar a los estados.

- **Regla 6**. Un estado Inicio, Fin, Interno, Comunicación, Simple, Compuesto, Apertura, Atado y Comienzo únicamente está asociado con uno o más enlaces simples etiquetados.

  En la Figura 5.23 se observan ejemplos de la forma correcta en la aplicación de la regla 6, así como la forma incorrecta de su aplicación.

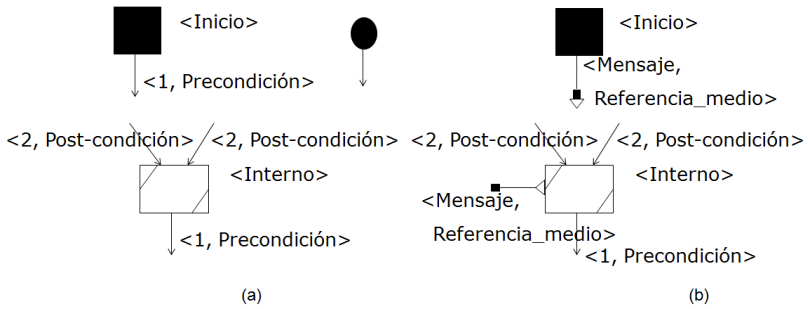


Figura 5.23: (a). Aplicación correcta de la regla 6. (b). Aplicación incorrecta de la regla 6.

La Figura 5.23(a) muestra la aplicación correcta de la regla 6, en donde los estados Inicio, Transición e Interno están asociados a uno o más enlaces simples etiquetados. La Figura 5.23(b) muestra una aplicación incorrecta de la regla 6, en donde los estados Inicio e Interno están asociados a enlaces de mensaje (etiquetados-entrada o etiquetados-salida).

- **Regla 7**. Un estado Punto de acceso está asociado con uno o más enlaces simples etiquetados y con un único enlace de mensaje (etiquetado-entrada y/o etiquetado-salida).

  En la Figura 5.24 se observan ejemplos de la forma correcta en la aplicación de la regla 7, así como la forma incorrecta de su aplicación.

  La Figura 5.24(a) muestra la aplicación correcta de la regla 7, en donde los estados Punto de acceso están asociados a uno o más enlaces simples etiquetados, y a un sólo enlace de mensaje etiquetado-entrada o etiquetado-salida, o están asociados a un sólo enlace de mensaje etiquetado-entrada y etiquetado-salida. La Figura 5.24(b) muestra una aplicación incorrecta de la regla 7, en donde el estado Punto de acceso está asociado únicamente a un enlace de mensaje etiquetado-entrada pero sin tener algún enlace simple etiquetado.

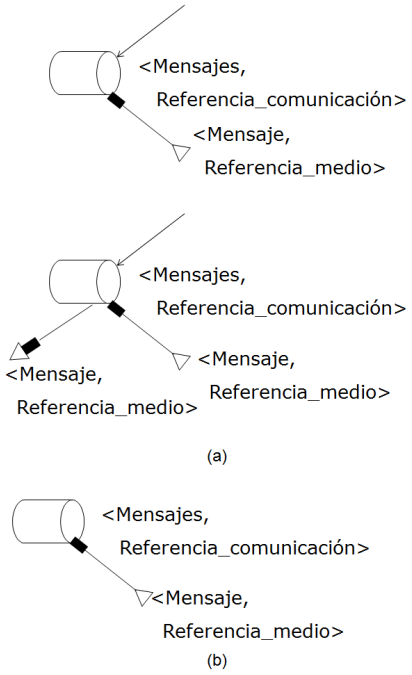


Figura 5.24: (a). Aplicación correcta de la regla 7. (b). Aplicación incorrecta de la regla 7.

- **Regla 8**. Un estado Punto de acceso y un estado Transición están conectados entre ellos con un enlace de mensaje etiquetado-entrada y/o etiquetado-salida, debiendo estar asociados a un estado Comunicación o Simple o Compuesto del medio.

  En la Figura 5.25 se observan ejemplos de la forma correcta en la aplicación de la regla 8, así como la forma incorrecta de su aplicación.

  La Figura 5.25(a) muestra la aplicación correcta de la regla 8, en donde los estados Punto de acceso y Transición están conectados a través de un enlace de mensaje etiquetado-salida. La Figura 5.25(b) muestra una aplicación incorrecta de la regla 8, en donde los estados Comunicación y Transición están conectados a través de un enlace de mensaje etiquetado-salida.

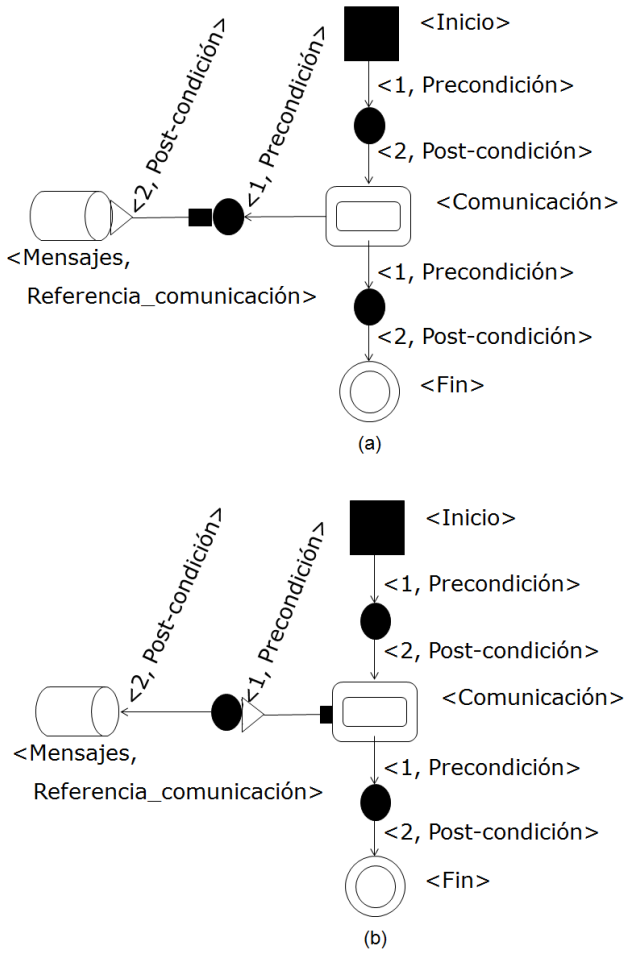


Figura 5.25: (a). Aplicación correcta de la regla 8. (b). Aplicación incorrecta de la regla 8.

- **Regla 9**. Un estado Comunicación conectado a un estado Punto de acceso, mediante un estado Transición, debe especificar tanto sus declaraciones como la etiqueta del nombre del estado Punto de acceso al cual se asocia. En otro caso las declaraciones y la etiqueta no se deben especificar.

  En la Figura 5.26 se observan ejemplos de la forma correcta en la aplicación de la regla 9, así como la forma incorrecta de su aplicación.

  La Figura 5.26(a) muestra la aplicación correcta de la regla 9, en donde las declaraciones y la etiqueta del nombre del estado Punto de acceso asociado al estado Comunicación se han especificado. La Figura 5.26(b) muestra una aplicación incorrecta de la regla 9, en donde no se han especificado las declaraciones y la etiqueta del nombre del estado Punto de acceso asociado al estado Comunicación.

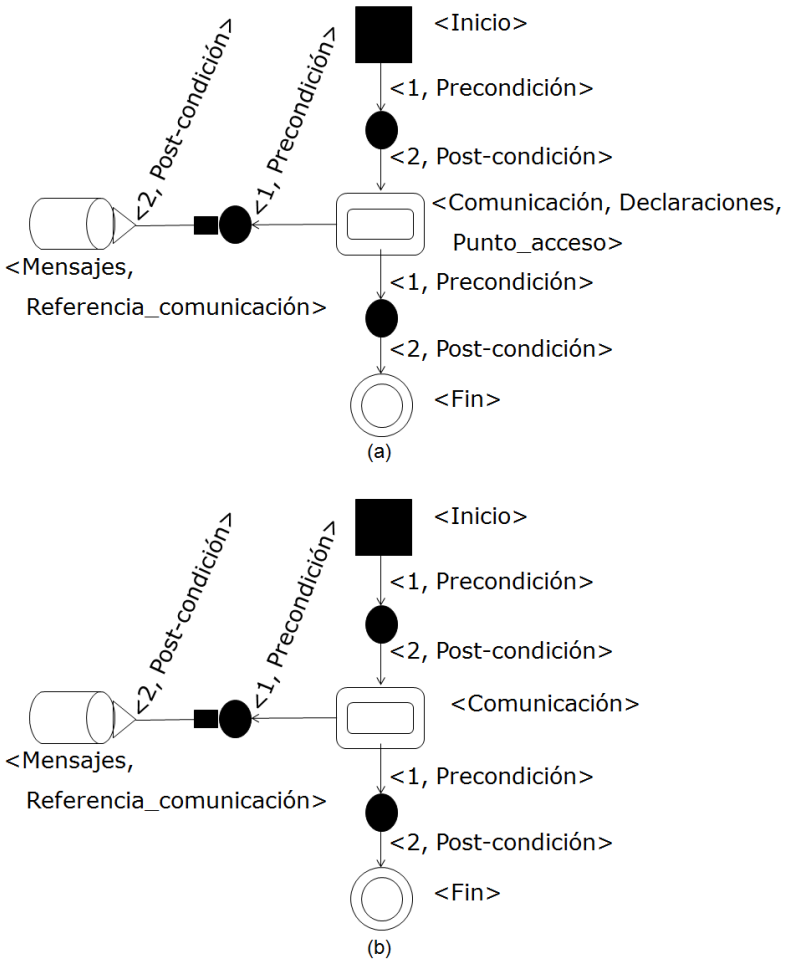


Figura 5.26: (a). Aplicación correcta de la regla 9. (b). Aplicación incorrecta de la regla 9.

- **Regla 10**. Un estado Comunicación que en sus declaraciones especifique la constante LeGESD CM siempre especificará su diagrama de comunicación del medio de ejecución correspondiente, y este diagrama debe estar compuesto de estados Simples.

  En la Figura 5.27 se observan ejemplos de la forma correcta en la aplicación de la regla 10, así como la forma incorrecta de su aplicación.

  La Figura 5.27(a) muestra la aplicación correcta de la regla 10, en donde el estado Comunicación en sus declaraciones especifica la constante LeGESD CM (diagrama a la izquierda), por lo que su diagrama de comunicación del medio de ejecución asociado está especificado haciendo uso del estado Simple (diagrama a la derecha). La Figura 5.27(b) muestra una aplicación incorrecta de la regla 10, en donde el estado Comunicación en sus declaraciones especifica la constante LeGESD CM (diagrama a la izquierda), pero su diagrama de comunicación del medio de ejecución asociado está especificado haciendo uso del estado Compuesto (diagrama a la derecha).

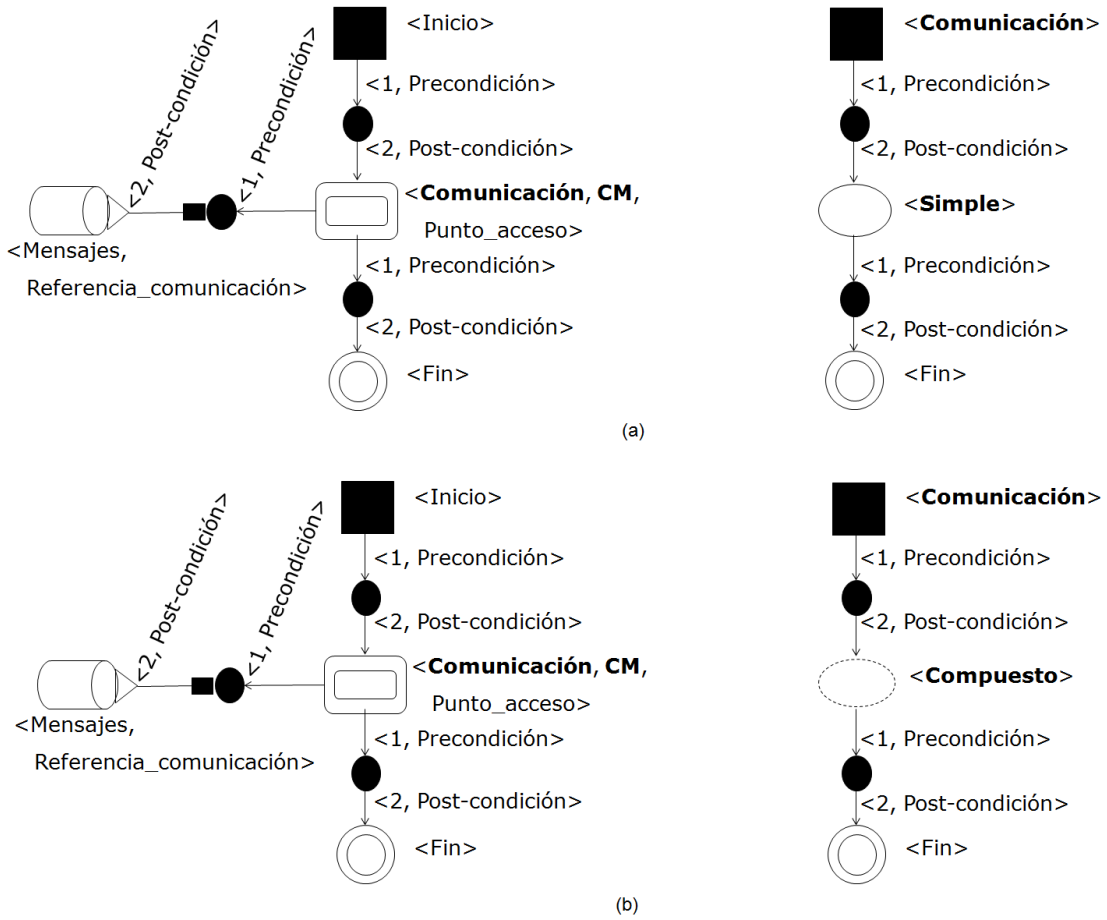


Figura 5.27: (a). Aplicación correcta de la regla 10. (b). Aplicación incorrecta de la regla 10.

- **Regla 11**. Un estado Comunicación que en sus declaraciones especifique la constante LeGESD CIF siempre especificará su diagrama de comunicación del medio de ejecución correspondiente, y este diagrama debe contener estados Compuestos.

  En la Figura 5.28 se observan ejemplos de la forma correcta en la aplicación de la regla 11, así como la forma incorrecta de su aplicación.

  La Figura 5.28(a) muestra la aplicación correcta de la regla 11, en donde el estado Comunicación en sus declaraciones especifica la constante LeGESD CIF (diagrama a la izquierda), por lo que su diagrama de comunicación del medio de ejecución asociado está especificado haciendo uso del estado Compuesto (diagrama a la derecha). La Figura 5.28(b) muestra una aplicación incorrecta de la regla 11, en donde el estado Comunicación en sus declaraciones especifica la constante LeGESD CIF (diagrama a la izquierda), pero su diagrama de comunicación del medio de ejecución asociado está especificado haciendo uso del estado Simple (diagrama a la derecha).

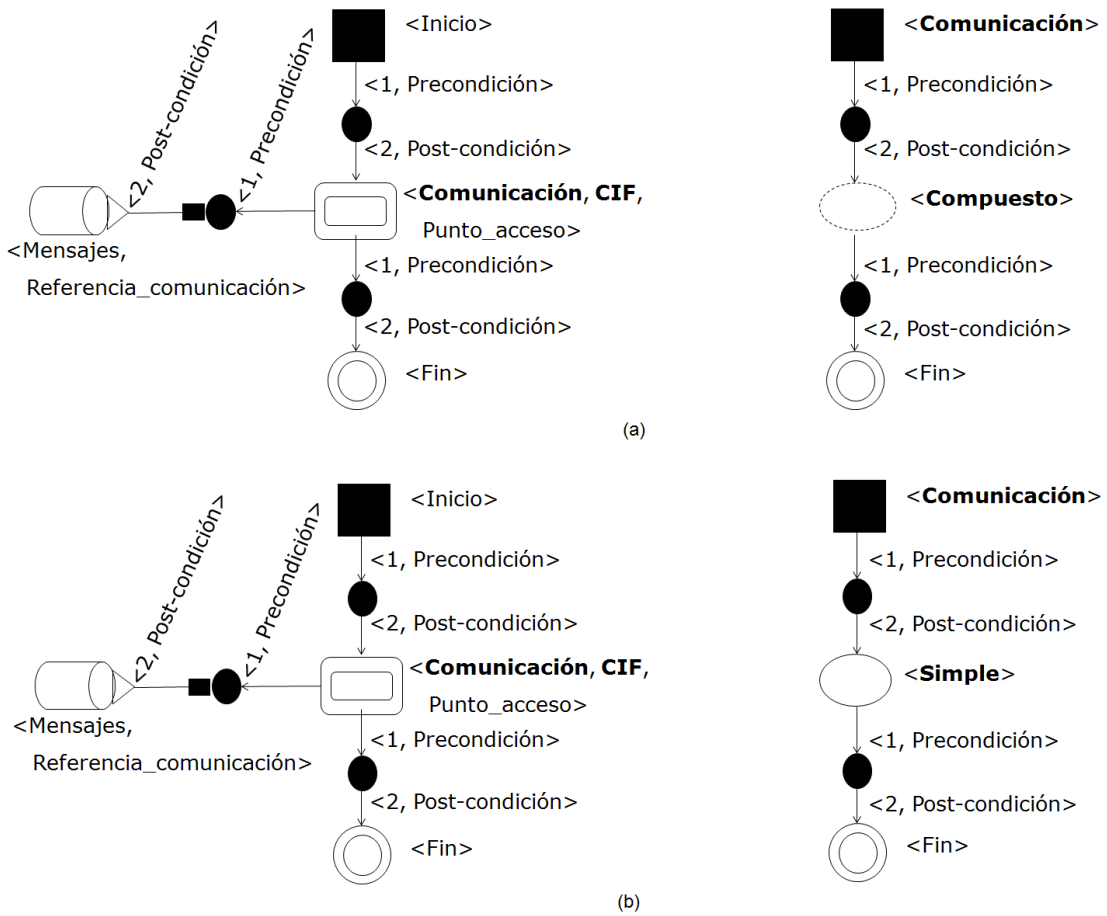


Figura 5.28: (a). Aplicación correcta de la regla 11. Figura 5.28 (b). Aplicación incorrecta de la regla 11.

- **Regla 12**. Un estado Compuesto o un estado Comunicación (sin conexión con un estado Punto de acceso) dentro de un Trabajo, siempre debe relacionarse con su diagrama de comportamiento en el nivel siguiente jerárquico de la especificación. Esta regla se aplica recursivamente a cada estado Compuesto o estado Comunicación que se encuentre en algún nivel jerárquico de la especificación, hasta no encontrarse con más estados Compuestos o Comunicación.

  La Figura 5.29 muestra la aplicación correcta de la regla 12, en donde los estados Comunicación y Compuesto del primer nivel jerárquico de la especificación (diagramas de comportamiento a la izquierda), son relacionados con sus diagramas de comportamiento en el siguiente nivel jerárquico de la especificación (diagramas centrales), y finalmente este nivel es relacionado con sus diagramas de comportamiento en el último nivel jerárquico de la especificación (diagramas a la derecha), no habiendo más estados Comunicación o Compuestos a especificar con sus diagramas de comportamiento. La forma incorrecta de la aplicación de la regla 12, se produce cuando no es especificado el diagrama de comportamiento correspondiente al estado Comunicación o Compuesto.

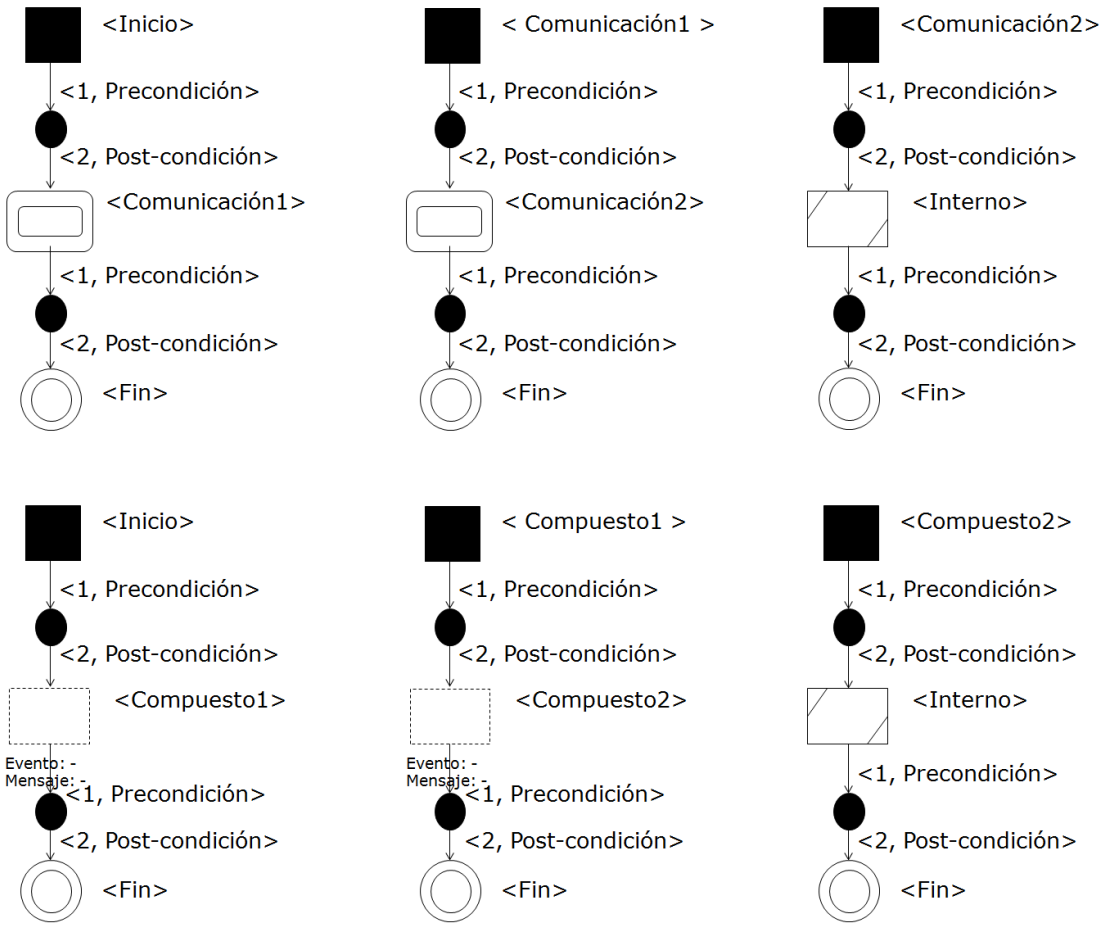


Figura 5.29: Aplicación correcta de la regla 12.

- **Regla 13**. El nombre de la etiqueta del estado Inicio, de un diagrama de comportamiento o de comunicación que corresponda a un estado Compuesto o Comunicación, debe contener el mismo nombre que la etiqueta asociada al estado Compuesto o Comunicación.

  La Figura 5.30 muestra la aplicación correcta de la regla 13, en donde los diagramas de comportamiento de la parte central y a la derecha de la figura, tienen en la etiqueta del nombre de sus estados Inicio tanto 'Comunicación' como 'Compuesto', los cuales corresponden respectivamente a los estados Comunicación y Compuesto cuyas etiquetas de nombre son 'Comunicación' y 'Compuesto'. La forma incorrecta de la aplicación de la regla 13, se produce cuando es especificado en la etiqueta del nombre de un estado Inicio, un nombre diferente al que se dio en el nivel superior jerárquico de la especificación, por ejemplo si en el diagrama de comportamiento central de la figura se especificara el nombre de la etiqueta como 'Comunicación1', entonces no correspondería con el nombre de la etiqueta dado al estado Comunicación en el diagrama de comportamiento de la izquierda.

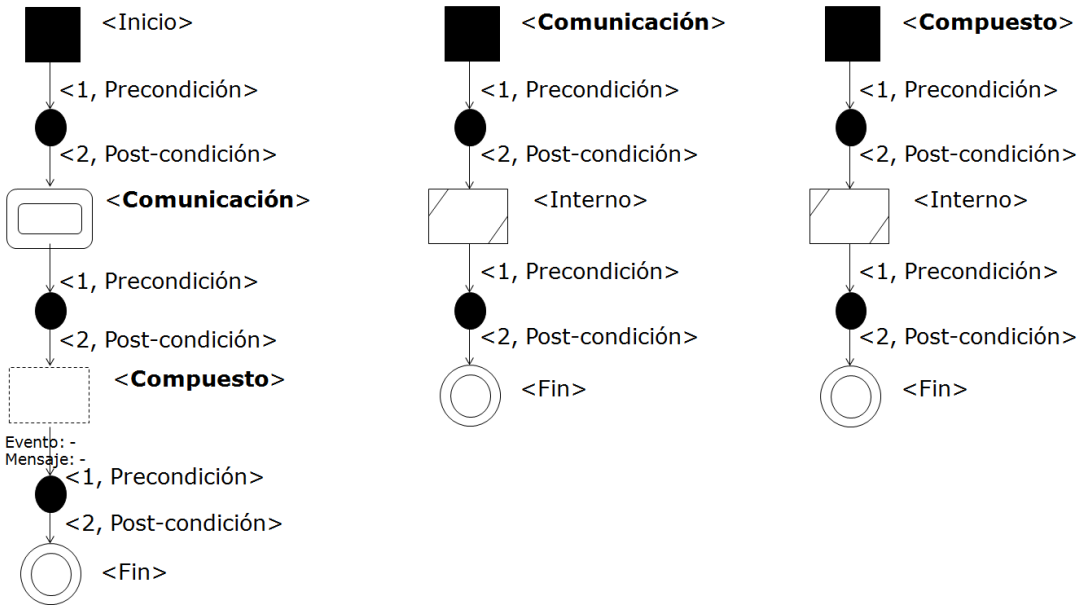


Figura 5.30: Aplicación correcta de la regla 13.

- **Regla 14**. Para un estado Punto de acceso con un enlace de mensaje etiquetado-salida, en un diagrama de comportamiento o de comunicación, debe existir un estado Punto de acceso con un enlace de mensaje etiquetado-entrada en otro diagrama de comportamiento o de comunicación, y viceversa.

  La Figura 5.31 muestra la aplicación correcta de la regla 14, en donde el diagrama de comportamiento de la parte superior de la figura tiene un estado Punto de acceso conectado a un enlace de mensaje etiquetado-salida, y el diagrama de comportamiento de la parte inferior tiene un estado Punto de acceso conectado a un enlace de mensaje etiquetado-entrada. La forma incorrecta de la aplicación de la regla 14, se produce cuando no es especificado en el diagrama de la parte inferior, un estado Punto de acceso conectado a un enlace de mensaje etiquetado-entrada, el cual corresponda al estado Punto de acceso conectado a un enlace de mensaje etiquetado-salida en el diagrama de la parte superior.

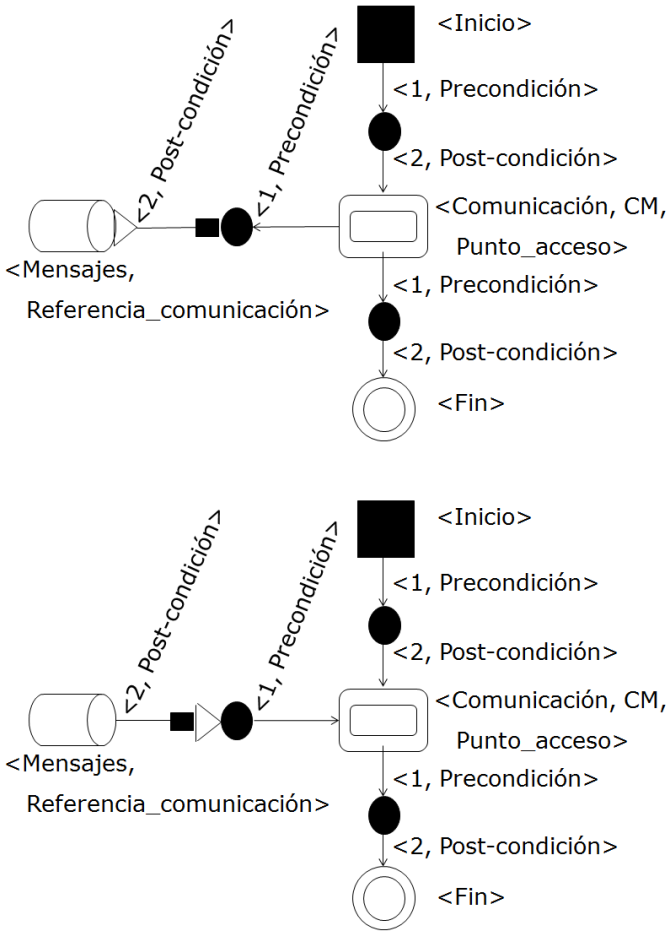


Figura 5.31: Aplicación correcta de la regla 14.

- **Regla 15**. La referencia de comunicación de un estado Punto de acceso con un enlace de mensaje etiquetado-salida, en un diagrama de comportamiento o de comunicación, debe ser la misma referencia en un estado Punto de acceso con un enlace de mensaje etiquetado-entrada en otro diagrama de comportamiento o de comunicación, y viceversa.

  En la Figura 5.32 se observan ejemplos de la forma correcta en la aplicación de la regla 15.

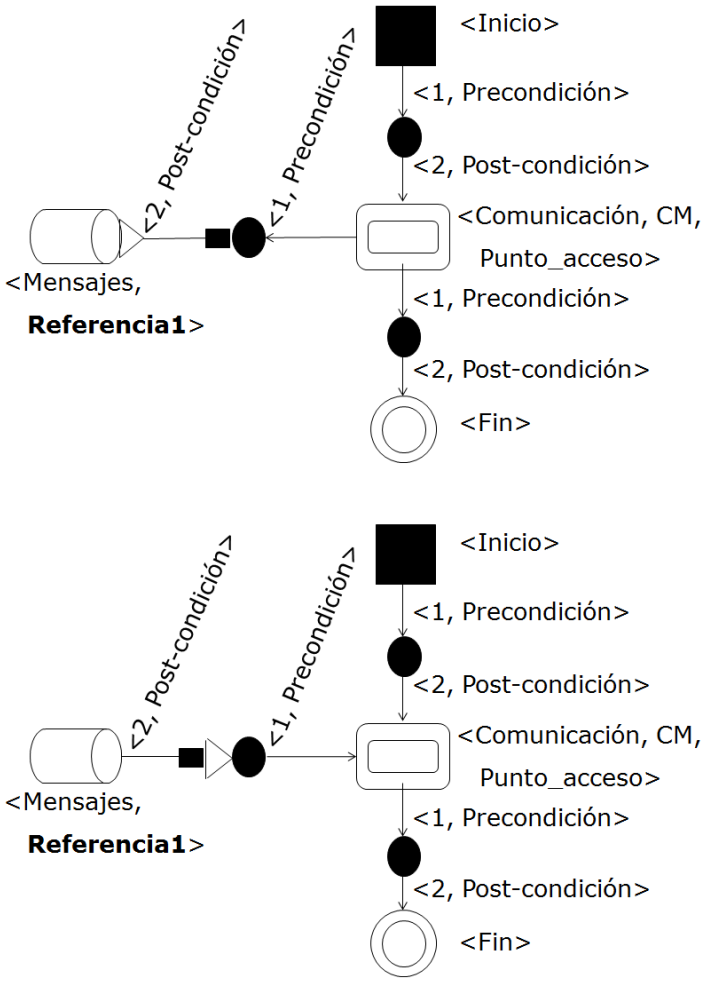


Figura 5.32: Aplicación correcta de la regla 15.

La Figura 5.32 muestra la aplicación correcta de la regla 15, en donde el diagrama de comportamiento de la parte superior de la figura tiene un estado Punto de acceso con una referencia nombrada como 'Referencia1', y el diagrama de comportamiento de la parte inferior tiene un estado Punto de acceso con una referencia nombrada como 'Referencia1', la cual corresponde con el estado Punto de acceso del diagrama de comportamiento de la parte superior. La forma incorrecta de la aplicación de la regla 15, se produce cuando no se especifica en el diagrama de la parte inferior, la misma referencia que se utilizó en el diagrama de comportamiento de la parte superior para el estado Punto de acceso.

Con esto concluye la presentación de los elementos sintácticos (notación gráfica), del pseudocódigo y de las reglas sintácticas de LeGESD. En el siguiente capítulo, se complementa el desarrollo de la propuesta de solución que propone la presente tesis, al abordar la semántica de LeGESD.

6 Análisis y Diseño de Sistemas Distribuidos

El Análisis y Diseño de Sistemas Distribuidos complementa a la propuesta de LeGESD, constituyéndose en el formalismo del lenguaje al ser el álgebra de procesos propuesta para sustentar la semántica de LeGESD. El objetivo del presente capítulo es presentar el álgebra de procesos de LeGESD, nombrada como Análisis y Diseño de Sistemas Distribuidos (ADSD).

6.1. Introducción

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 [19] que permite construir ecuaciones algebraicas a partir de la especificación gráfica realizada con el lenguaje LeGESD de un sistema distribuido dado. Partiendo de este grupo de ecuaciones algebraicas es posible realizar la verificación formal de la especificación.

El álgebra de procesos ofrece un marco de trabajo matemático bien elaborado para el análisis de la estructura y comportamiento de un sistema de manera composicional, es decir, a través de la descomposición del sistema en componentes. El álgebra de procesos incluye mecanismos de abstracción que permiten el manejo de los componentes de un sistema como cajas negras, encapsulando su estructura interna. Las diversas álgebras de procesos están típicamente integradas por una semántica operacional [35], [38] con una estructura formalmente definida que traduce los términos del álgebra de procesos en sistemas de transición etiquetados [36] y viceversa, todo esto con un enfoque composicional. ADSD proporciona este marco de trabajo para el análisis de la estructura y comportamiento de un sistema distribuido especificado con LeGESD de manera composicional.

6.2. Formalismo de ADSD

6.2.1. Definiciones básicas

ADSD está basado en el álgebra de procesos, por lo cual es importante dar la definición de álgebra de procesos. Para la presente tesis se define al álgebra de procesos como sigue:

Definición 6.1. *El álgebra de procesos es una técnica formal de descripción de sistemas computacionales reactivos que estén integrados con componentes de comunicación y concurrencia.*

ADSD comparte tres características importantes con otras álgebras de procesos ampliamente utilizadas, estas características son:

- Especificación composicional. ADSD proporciona un conjunto de operadores para la construcción de sistemas distribuidos extensos a partir de su descomposición en elementos más reducidos. Concretamente, ADSD proporciona cinco operadores, algunos de los cuales son los utilizados en el álgebra de procesos CCS.
- Semántica operacional. ADSD está conformada por un conjunto de reglas de transición que describe la ejecución paso a paso del sistema distribuido especificado con LeGESD. Haciendo uso de esta semántica operacional, el sistema representado como términos del álgebra es la especificación formal de la semántica de LeGESD.
- Análisis del comportamiento. ADSD establece un algoritmo de transformación y un teorema de equivalencia entre los elementos sintácticos de LeGESD y los términos algebraicos de la semántica operacional de ADSD.

Las ventajas que ADSD presenta son las siguientes:

- Permite segmentar la especificación de sistemas distribuidos extensos en componentes más reducidos, permitiendo un análisis modular del comportamiento del sistema completo.
- Permite verificar la especificación realizada en el lenguaje gráfico LeGESD a través de relaciones de equivalencia que generan conjuntos de ecuaciones algebraicas representativas de la semántica operacional de ADSD.

ADSD al ser un álgebra de procesos, su elemento básico de operación es el proceso, un proceso ADSD, pero ¿qué es un proceso ADSD?

Definición 6.2. *Un proceso ADSD es un estado computacional que puede interactuar con su ambiente a través de un conjunto de transiciones.*

Partiendo de esta definición, es posible decir que el sistema distribuido especificado en LeGESD está formado por múltiples procesos ADSD, y que en su conjunto determinan el comportamiento general del sistema. De esta manera, se puede afirmar que ADSD tiene dos propósitos principales:

- La especificación formal de la semántica de LeGESD. Consiste en representar el comportamiento del sistema a través de procesos ADSD incluidos en ecuaciones algebraicas que conforma la semántica operacional de ADSD.

- La verificación de la especificación. Consiste en comprobar propiedades de seguridad acerca del sistema especificado con LeGESD.

En toda álgebra de procesos debe especificarse cuál es el dominio de procesos que se empleará, un dominio de procesos se define como:

Definición 6.3. *Un dominio de procesos es el dominio matemático en el cual se representan los procesos ADSD como elementos de ese dominio.*

En este dominio de procesos, es posible mantener una equivalencia semántica. La equivalencia semántica se define como:

Definición 6.4. *Una equivalencia semántica es la especificación de la semántica operacional de ADSD como una equivalencia en el dominio de procesos.*

El dominio de procesos que utiliza ADSD es el dominio gráfico, en el cual el comportamiento de los procesos ADSD se representa a través de los trabajos o medios de LeGESD.

Otro aspecto a considerar en el álgebra de procesos son las relaciones de acción, una relación de acción se define como:

Definición 6.5. *Sean x y z dos procesos ADSD, relacionados mediante la expresión* $x \xrightarrow{a} z$. *Si el proceso x puede llegar al proceso z a través de la realización de la acción a, entonces el predicado binario* $\xrightarrow{a}$ *es conocido como una relación de transición.*

Por otra parte, una acción se define como:

Definición 6.6. *Una acción es cualquier actividad considerada como una entidad conceptual sobre un nivel de abstracción establecido.*

Estas definiciones se aplican al dominio de procesos que utiliza ADSD, en el cual las relaciones de acción están definidas a través de los enlaces simples y de los enlaces de mensaje de LeGESD. Las acciones corresponden a la precondición y post-condición asociadas al estado Transición.

En ADSD se delimita la clase de procesos a especificar a secuenciales, concretos y con ramificaciones finitas.

Definición 6.7. *Un proceso secuencial es un proceso capaz de realizar a lo más una acción a la vez.*

Definición 6.8. *Un proceso concreto es un proceso en el cual ninguna acción interna ocurre.*

Definición 6.9. *Un proceso con ramificaciones finitas es un proceso en el que en cada estado únicamente tiene un número finito de formas para cambiar de estado.*

6.2.2. LeGESD como un sistema de transición etiquetado

En el apartado anterior, se mencionó en la Definición 6.5 que los procesos son capaces de realizar acciones a partir de un conjunto de acciones dado; a este conjunto de acciones se le llamará *Acc*. Las acciones pueden ser instantáneas o con una duración y no es requisito que terminen de inmediato, sin embargo en un tiempo finito un número finito de acciones pueden ser realizadas.

Por otra parte, los procesos secuenciales se representan frecuentemente por sistemas de transición etiquetados, los cuales se encuentran en un dominio E. Sobre este dominio se escriben predicados binarios de la forma $\xrightarrow{a}$ definidos para cada acción $a \in Acc$. Los elementos de E representan procesos, y la expresión $x \xrightarrow{a} z$ significa que el proceso x realizará o validará la acción a para alcanzar el proceso z. En un sistema de transición etiquetado puede ocurrir que $x \xrightarrow{a} z$ y que $x \xrightarrow{b} w$ debido a las acciones diferentes a y b o a los diferentes procesos z y w, a esta situación se le nombra ramificación.

Ciertas acciones pueden ser de sincronización de los procesos con su ambiente de ejecución, o pueden ser de recepción de señales o mensajes enviados por el ambiente. En un sistema de transición etiquetado todas estas posibles acciones son incluidas, de forma que $\xrightarrow{a}$ significa que existe un ambiente de ejecución en el cual la acción a ocurre, independientemente de qué tipo de acción se haya producido.

Después de que se han establecido las características de los sistemas de transición etiquetados, se definirán los trabajos y medios de LeGESD como sistemas de transición etiquetados.

Definición 6.10. *Un trabajo o medio es un par $(E, \rightarrow)$ con E siendo una clase y $\rightarrow \subseteq E \times Acc \times E$, tal que para $x \in E$ y $a \in Acc$ la clase $\{z \in E \mid (x,a,z) \in \rightarrow\}$ es un conjunto.*

Partiendo de esta definición, dado un trabajo o medio $(E, \rightarrow)$ con valores $x, z, w, \ldots$ se denota $x \xrightarrow{a} z$ para $(x,a,z) \in \rightarrow$. Los predicados binarios $\xrightarrow{a}$ son nombrados relaciones de acción.

Las propiedades que cumplen los trabajos y medios LeGESD son las siguientes:

- $x \xrightarrow{\varepsilon} x$, para cualquier proceso x donde ε es una acción vacía.
- $(x,a,z) \in \rightarrow$ con $a \in Acc$ implica que $x \xrightarrow{a} z$ con $a \in Acc^+$, donde Acc^+ es el conjunto de secuencias finitas de acciones.
- $x \xrightarrow{\beta} z \xrightarrow{\mu} w$ implica $x \xrightarrow{\beta\mu} w$, donde $\beta\mu$ es la concatenación de $\beta \in Acc^+$ y de $\mu \in Acc^-$, siendo Acc^- el conjunto de secuencias finitas e infinitas de acciones de Acc.
- Un proceso ADSD $z \in E$ es alcanzable desde $x \in E$ si $x \xrightarrow{\beta} z$ para algún $\beta \in Acc^+$.
- Un proceso ADSD $x \in E$ es finito si el conjunto $\{(\beta, z) \in (Acc^+ \times E) \mid x \xrightarrow{\beta} z\}$ es finito.
- x está bien formado si no existe una secuencia infinita de la forma $x \xrightarrow{\beta_1} x_1 \xrightarrow{\beta_2} x_2 \xrightarrow{\beta_3} x_3 \ldots$.
- x tiene ramificaciones finitas si para cada z alcanzable desde x, el conjunto $\{(\beta, w) \in (Acc \times E) \mid z \xrightarrow{\beta} w\}$ es finito.

6.2.3. LeGESD como un grafo de procesos ADSD

La definición de LeGESD como un grafo de procesos ADSD es la siguiente:

Definición 6.11. *Un grafo de procesos ADSD en un alfabeto Acc es un grafo dirigido con un inicio y un fin cuyos enlaces están etiquetados por elementos de Acc. Formalmente, un grafo de proceso g es una tupla formada por $(ESTADOS(g), INICIO(g), ENLACES(g), FIN(g))$, donde*

- $ESTADOS(g)$ *es un conjunto en el cual sus elementos son nombrados los estados de g.*
- $INICIO(g) \in ESTADOS(g)$ *es un estado especial: el estado inicial de g.*
- $ENLACES(g) \subseteq ESTADOS(g) \times Acc \times ESTADOS(g)$ *es un conjunto de tuplas formadas por* (x,a,z) *con* $x,z \in ESTADOS(g)$ *y* $a \in Acc$*,es decir los enlaces de transición de g.*
- $FIN(g) \in ESTADOS(g)$ *es un estado especial, es decir el estado final de g.*

Si $r = (x,a,z) \in ENLACES(g)$, es posible afirmar que r se dirige de x hacia z. Una ruta finita R en un grafo de procesos ADSD es una secuencia alternada de estados y enlaces, iniciando y terminando con un estado, de esta manera cada enlace se dirige desde el estado anterior hacia el estado posterior a éste. Si $R = x_0(x_0,a_1,x_1)x_1(x_1,a_2,x_2)\ldots x_{n-1}(x_{n-1},a_n,x_n)x_n$ también denotado como $R = x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} x_n$, se puede afirmar que R se dirige de x_0 hacia x_n, es decir, R inicia en x_0 y termina en $fin(R) = x_n$.

Sea $RUTAS(g)$ el conjunto de rutas en g que inician desde el estado inicial. Si x y z son estados en un grafo de procesos ADSD entonces z puede ser alcanzado desde x si existe una ruta desde x hasta z. Un grafo de procesos ADSD se considera como conectado si todos sus estados pueden ser alcanzados a partir del estado inicial.

Sea G el dominio de grafos de procesos ADSD conectados sobre un alfabeto Acc dado.

Definición 6.12. *Para* $g \in G$ *y* $x \in ESTADOS(g)$*, sea g el grafo de procesos ADSD definido por*

- $ESTADOS(g_x) = \{z \in ESTADOS(g) \mid$ *existe una ruta dirigida desde x hacia* $z\}$.
- $INICIO(g_x) = x \in ESTADOS(g_x)$.
- $(z,a,w) \in ENLACES(g_x)$ *si* $z,w \in ESTADOS(g_x)$ *y* $(z,a,w) \in ENLACES(g)$.
- $FIN(g_x) = x \in ESTADOS(g_x)$.

De lo anterior se deduce que $g_x \in G$, además $g_{INICIO(g)} = g$. Para G las relaciones de acción $\xrightarrow{a}$ para $a \in Acc$ están definidas por $g \xrightarrow{a} h$ si $(INICIO(g),a,x) \in ENLACES(g)$ y $h = g_x$. Esto hace a G un sistema de transición etiquetado, como LeGESD.

Definición 6.13. *Sea* $(E,\rightarrow)$ *un trabajo o medio arbitrario de LeGESD y sea* $x \in E$*. El grafo de procesos ADSD* $G(x)$ *de x está definido como*

- $ESTADOS(G(x)) = \{z \in E \mid \exists \beta \in Acc^+ : x \xrightarrow{\beta} z\}$.
- $INICIO(G(x)) = x \in ESTADOS(G(x))$.
- $(z,a,w) \in ENLACES(G(x))$ *si* $z,w \in ESTADOS(G(x))$ *y* $z \xrightarrow{a} w$.
- $FIN(G(x)) = x \in ESTADOS(G(x))$.

De lo anterior se deduce que $G(x) \in G$, esto significa que G es una función de E hacia G, y se puede decir que:

$G : E \to G$ es inyectiva y satisface que para $a \in Acc : G(x) \xrightarrow{a} G(z) \Leftrightarrow x \xrightarrow{a} z$. Por otra parte, $G(x) \xrightarrow{a} h$ sólo si h tiene la forma $G(z)$ para algún $z \in E$ (con $x \xrightarrow{a} z$).

Lo anterior confirma que G es una inclusión de E en G, lo que implica que cualquier trabajo o medio de LeGESD sobre Acc puede ser representado como una subclase $G(E) = \{G(x) \in G \mid x \in E\}$ de G.

6.2.4. Álgebra de procesos ADSD

ADSD tiene como fundamento el álgebra de procesos CCS [19] con acciones presentándose en tiempos discretos. 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.

El álgebra de procesos ADSD se explica con base en las tres características que identifican a cualquier álgebra de procesos: especificación composicional, semántica operacional, y análisis de comportamiento.

Permitiendo a la gramática P variar en el dominio de términos (procesos) T, e en el conjunto de eventos etiquetados Acc, F en un conjunto de eventos etiquetados ($F \in e$), m en un conjunto de mensajes y X en el dominio de variables de eventos externos.

Definición 6.14. *La sintaxis (especificación composicional) de ADSD está dada por la gramática siguiente:*

$$P ::= P_1 \overset{\alpha}{\to}\overset{\beta}{} P_2 \mid P1 + P2 \mid P\Delta^b(P_1) \mid [P/F]m \mid recX.P \mid X \mid 0$$

La especificación composicional de ADSD de acuerdo a la Definición 6.14 consiste en los siguientes elementos.

- El operador transición $\to$ es utilizado para representar transiciones entre los procesos ADSD.
- El operador elección $+$ $(P_1 + P_2)$ es utilizado para representar dos posibilidades, alguno de los dos procesos ADSD P_1 ó P_2 puede ser elegido para ejecutarse. Esta elección depende del evento presente.
- El operador alcance Δ^b $(P\Delta^b(P_1))$ es utilizado para representar el enlace del proceso P mediante un evento de alcance b.
- El operador restricción $[/]$ $([P/F]m)$ es utilizado para limitar el comportamiento de P según F o m.
- El operador *rec* $(recX.P)$ es utilizado para representar recursión, permitiendo especificar comportamientos recursivos.
- El término X, sin algún enlace a 'rec', es una variable libre (variable externa e independiente al operador 'rec') que pertenece a un conjunto finito de variables de término libres.
- El término 0 es un indicador de paro.

La cantidad de operadores debe ser la suficiente para proporcionar una alta flexibilidad en la especificación del comportamiento de cada proceso que interviene en el sistema. Concretamente para los sistemas reactivos, como son los sistemas distribuidos, es imprescindible mantener operadores diferenciados para los comportamientos de interacción entre los procesos de diferentes sistemas. Es por ello que ADSD, define los operadores alcance (Δ^b) y restricción ($[/]$) para proporcionar esta flexibilidad en la especificación de sistemas distribuidos.

Definición 6.15. *La semántica operacional de ADSD es un conjunto de reglas de transiciones las cuales indican la manera en que un proceso x puede alcanzar el proceso z a través del evento a.*

La semántica operacional de ADSD de acuerdo a la Definición 6.15 consiste en las reglas de transición siguientes:

- **Regla 1**. El operador transición $\rightarrow (P_1 \overset{\alpha}{\rightarrow}{}^{\beta} P_2)$ denota una transición del proceso ADSD P_1 al proceso ADSD P_2 bajo los eventos α y β que P_1 debe ejecutar para alcanzar P_2. El evento α se relaciona con la precondición de LeGESD y el evento β se relaciona con la post-condición de LeGESD. Si α y β están vacíos entonces $P_1 \overset{\varepsilon}{\rightarrow} P_1$.
- **Regla 2**. El operador elección $+$ $(P_1 + P_2)$ denota una transición de un proceso ADSD P a alguno de los dos procesos ADSD P_1 o P_2, es decir, $P \overset{\alpha}{\rightarrow}{}^{\beta} P_1$ o $P \overset{\alpha}{\rightarrow}{}^{\beta} P_2$ donde la elección entre P_1 o P_2 depende del evento α y β que P debe ejecutar.
- **Regla 3**. El operador alcance Δ^b $(P\Delta^b(P_1))$ denota una transición en la cual el operador alcance puede terminar únicamente si el proceso ADSD P concluye exitosamente mediante la ejecución de un evento de alcance etiquetado como b, entonces el control procede al 'manejador de éxito' P_1, donde b es evento α y β que P debe ejecutar. De otro modo $P \overset{\varepsilon}{\rightarrow} P$, donde ε es un evento vacío.
- **Regla 4**. El operador restricción $[/]$ $([P/F]m)$ denota una transición de P a otro proceso ADSD x únicamente si eventos etiquetados en F sincronizan y forman parte del conjunto de eventos Acc o si mensajes m han sido recibidos.
- **Regla 5**. El operador *rec* $(recX.P)$ denota una transición de P así mismo controlada por la variable de evento externa X, es decir, $P \overset{X}{\rightarrow} P$ donde X es un evento α y β que P debe ejecutar.

Partiendo de estas reglas de transición, es posible construir ecuaciones de transición ADSD las cuales complementan la semántica operacional de ADSD. Una ecuación de transición ADSD se define como:

Definición 6.16. *Una ecuación de transición ADSD es una expresión resultante de una transición que contiene como términos algebraicos, procesos ADSD, asociados por operadores.*

Estas ecuaciones de transición se construyen a partir de la relación existente entre los trabajos y medios de LeGESD con ADSD, en su conjunto, el grupo de ecuaciones de transición que se generan conforman el comportamiento de los trabajos y medios especificados. Haciendo uso de estas ecuaciones de transición, es posible realizar la verificación de la especificación construida,

permitiendo validar las afirmaciones que se realicen sobre el comportamiento del sistema especificado.

Como se mencionó anteriormente, existe una relación entre los trabajos y medios de LeGESD con ADSD. Esta relación se crea a través de un algoritmo de transformación que conforma la última característica del álgebra de procesos ADSD, el análisis de comportamiento de ADSD, el cual se define como:

Definición 6.17. *El análisis de comportamiento de ADSD es la equivalencia entre los términos algebraicos de la semántica operacional de ADSD (procesos ADSD) y los elementos sintácticos de LeGESD (estados LeGESD).*

El análisis de comportamiento de ADSD complementa la semántica de LeGESD, este análisis de comportamiento consiste de un algoritmo de transformación de estados LeGESD a procesos ADSD y de un teorema de equivalencia gráfico-algebraica.

6.3. Análisis de comportamiento de ADSD

En este apartado se presenta el algoritmo de transformación de estados LeGESD a procesos ADSD y el teorema de equivalencia gráfico-algebraica, así como una serie de teoremas auxiliares que validan que el algoritmo es correcto, y que establecen que ambas especificaciones (gráfica y algebraica) de un sistema distribuido son equivalentes. Lo anterior significa que los resultados formales que se establecen sobre alguna de las especificaciones son aplicables a la otra. Esto es particularmente importante porque permite realizar demostraciones utilizando el álgebra de procesos, siendo estas demostraciones igualmente válidas a la especificación gráfica del sistema.

6.3.1. Transformación de estados LeGESD a procesos ADSD

La Tabla 6.1 muestra la representación textual asociada a cada una de las representaciones gráficas que serán utilizadas tanto en los teoremas como en el Algoritmo de transformación de estados LeGESD a procesos ADSD (TLA) que se desarrollarán en el presente apartado.

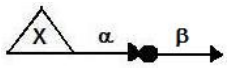
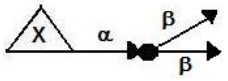
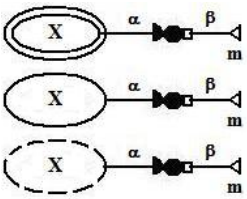
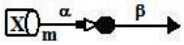
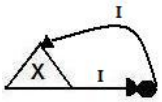
Representación gráfica	Representación textual
	$\Delta(X)_{\alpha,\beta}$
	$\Delta(X)_{\alpha}\Sigma_{\beta}^{\beta}$
	$\Delta(X)_{m,m}$
	$\Delta(X)^{m,m}$
	$\Delta(X)_{I,I}$

Tabla 6.1: Representación textual asociada a cada una de las representaciones gráficas.

Lo anterior tiene como único objetivo el que se pueda hacer una referencia precisa a los símbolos gráficos dentro de los teoremas y algoritmos.

6.3.1.1. Algoritmo de transformación de estados LeGESD a procesos ADSD (TLA)

En este apartado se presenta el Algoritmo de transformación de estados LeGESD a procesos ADSD (Algoritmo 1 TLA), el cual se basa en el algoritmo ”Depth-first search” con modificaciones realizadas para adecuarse a LeGESD. Este algoritmo tiene como propósito transformar los estados LeGESD de un diagrama de comportamiento o comunicación a sus correspondientes procesos ADSD, a través de la especificación de un conjunto de equivalencias locales entre los estados y los procesos. Posteriormente, el Algoritmo 1 será utilizado por el Teorema de equivalencia gráfico-algebraica que se propone más adelante.

Algoritmo 1: TLA Algoritmo de transformación de estados LeGESD a procesos ADSD

Entrada: Una especificación en LeGESD
Salida: Una especificación en ADSD

```
1  para cada diagrama de comportamiento o comunicación D ∈ LeGESD hacer
2      Comenzar a partir del estado inicial de D
3      para cada estado u ∈ E[D] hacer
4          Edo_Visitado[u] ← Falso
5          Edo_Predecesor[u] ← 0
6      para cada enlace a ∈ e[D] hacer
7          Enl_Visitado[a] ← Falso
8          Enl_Predecesor[a] ← 0
9      para cada estado u ∈ E[D] hacer
10         si Edo_Visitado[u] = Falso entonces
11             Visita(u,a)
12 devolver P[D]
```

El Algoritmo 2 corresponde a la función Visita() utilizada por el algoritmo 1. Esta función tiene como propósito visitar cada uno de los estados LeGESD contenidos en un diagrama de comportamiento o comunicación. Como parte de esta visita de estados, se realizará la transformación de cada estado visitado a su equivalente proceso ADSD.

Algoritmo 2: VISITA Algoritmo de la función Visita de TLA

Entrada: Un estado y un enlace LeGESD
Salida: Un proceso ADSD

```
1  Visita(u, a):
2  si (Edo_Visitado[u] ≠ Verdadero & Enl_Visitado[a] = Falso) |
   (Edo_Visitado[u] ≠ Verdadero & Enl_Adyacentes[a] = 0) entonces
3      Edo_Visitado[u] ← Verdadero
4  para cada v ∈ Edo_Adyacentes[u] | b ∈ Enl_Adyacentes[a] hacer
5      si (Edo_Visitado[v] = Falso & v ≠ Transición) | (Edo_Visitado[v] = Verdadero &
       v ≠ Transición) entonces
6          Enl_Visitado[a] ← Verdadero
7          Edo_Predecesor[v] ← u
8          Visita(v, b)
9      en otro caso
10         si (Edo_Visitado[v] = Falso & v = Transición) | (Edo_Visitado[v] = Verdadero &
           v = Transición & Enl_Visitado[b] = Falso) entonces
11             si Enl_Adyacentes[a] > 1 entonces
12                 c = Enl_Adyacentes[a]
13             en otro caso
14                 c = b
15             Enl_Visitado[a] ← Verdadero
16             Enl_Predecesor[b] ← a
17             Transformación(u, a, c)
18             Edo_Predecesor[v] ← u
19             Visita(v, b)
20 si (Edo_Adyacentes[u] = 0 & Enl_Adyacentes[a] = 0 entonces
21     P[D] ← T(v')
22 devolver P[D]
```

El Algoritmo 3 corresponde a la función Transformación() utilizada por el algoritmo 2. Esta función tiene como propósito transformar cada uno de los estados LeGESD contenidos en un diagrama de comportamiento o comunicación a su equivalente proceso ADSD, al cual se le asocia un operador ADSD para construir una ecuación de transición. La transformación de estados en el

algoritmo 3 se apoya de los Teoremas 6.1 a 6.5.

Algoritmo 3: TRANSFORMACIÓN Algoritmo de la función Transformación de TLA

Entrada: Un estado y dos enlaces LeGESD
Salida: Una ecuación de transición ADSD

```
1  Transformación(v, a, b):
2  si v, a y b son de la forma: Δ(X)_{α,β} con precondiciones α y postcondiciones β con valores
   (a_1, a_2, ..., a_n) & Mapeo(a_1, a_2, ..., a_n) entonces
3      P[D] ← T(v')^{α'} →^{β'}
4  en otro caso
5      si v, a y b son de la forma: Δ(X)_α Σ_β^β con precondiciones α y post-condiciones β con
       valores (a_1, a_2, ..., a_n) & Mapeo(a_1, a_2, ..., a_n) entonces
6          P[D] ← T(v')^{α'} →^{β'} + T(v')^{α'} →^{β'}
7      en otro caso
8          si v, a y b son de la forma: Δ(X)_{α,β,m} con precondiciones α, post-condiciones β y
           mensajes m con valores (a_1, a_2, ..., a_n) & Mapeo(a_1, a_2, ..., a_n) entonces
9              P[D] ← T(v')Δ^{α',β',m'}
10         en otro caso
11             si v, a y b son de la forma: Δ(X)^{m,α,β} con mensajes m, precondiciones α y
               post-condiciones β con valores (a_1, a_2, ..., a_n) & Mapeo(a_1, a_2, ..., a_n)
               entonces
12                 P[D] ← [T(v')/−]m'Δ^{α',β'}
13             en otro caso
14                 si v, a y b son de la forma: Δ(X)_{I,I} con precondiciones I y post-condiciones I
                   con valores (a_1, a_2, ..., a_n) & Mapeo(a_1, a_2, ..., a_n) entonces
15                     P[D] ← rec I'.T(v')
16 devolver P[D]
```

El Algoritmo 4 corresponde a la función Mapeo() utilizada por el algoritmo 3. Esta función tiene como propósito transformar cada uno de los valores de la precondición y post-condición, asociados a una transición entre estados LeGESD dentro de un diagrama de comportamiento o comunicación, a sus valores equivalentes de la precondición y post-condición asociados al proceso

ADSD.

Algoritmo 4: MAPEO Algoritmo de la función Mapeo TLA

Entrada: Un vector con las variables de la precondición y post-condición asociadas a una transición

Salida: Un valor de falso o verdadero

```
1 Mapeo(A):
2 para cada variable a' ∈ V[A] hacer
3     Var_ADSD[a'] ← 0
4 para cada variable a' ∈ V[A] hacer
5     Var_ADSD[a'] ← a'
6 para cada variable a' ∈ V[A] hacer
7     si Var_ADSD[a'] = 0 entonces
8         devolver falso
9 devolver verdadero
```

6.3.1.2. Teoremas auxiliares

La Tabla 6.2 muestra la notación que será utilizada en los teoremas de transformación de estados LeGESD a procesos ADSD que se desarrollarán en el presente apartado.

Notación	Significado
X	Un estado LeGESD dado.
X'	Un proceso ADSD dado.
$V_D(X)$	Un conjunto de variables definidas en X.
$V_{\Im}(X')$	Un conjunto de variables definidas en X'.
$\Upsilon_D(X)$	Un predicado de la forma $\nabla_{\forall i \in V_D(X)}^{i \doteq valor\ de\ i\ en\ X}$ que describe los valores de las variables en X.
$\Upsilon_{\Im}(X')$	Un predicado de la forma $\nabla_{\forall i \in V_{\Im}(X')}^{i \doteq valor\ de\ i\ en\ X'}$ que describe los valores de las variables en X'.
D	Un diagrama de comportamiento o comunicación en lenguaje LeGESD válido, compuesto por un conjunto de símbolos gráficos LeGESD y por una serie de precondiciones y post-condiciones sobre $V_D(X)$.
$\Im$	Una especificación de un sistema distribuido en el álgebra de procesos ADSD compuesta por $V_{\Im}(X')$, un conjunto de operadores, y un conjunto de ecuaciones de transición definidos en ADSD.
$T(X')$	Un proceso X' resultante de la equivalencia local de un estado X.

Tabla 6.2: Notación a utilizar en los teoremas.

Inicialmente, se presentan dos definiciones que conforman el marco de trabajo base para los teoremas auxiliares propuestos para la transformación de estados LeGESD a procesos ADSD. En las Definiciones 6.18 y 6.19, se explican los significados de los principales términos que serán

utilizados en la serie de teoremas propuestos, los cuales validarán que el Algoritmo de transformación de estados LeGESD a procesos ADSD es correcto, estableciendo que tanto la especificación en LeGESD como la especificación en ADSD son equivalentes para un sistema distribuido o concurrente.

En particular, la Definición 6.19 tiene el propósito de explicar el término de equivalencia local entre un estado LeGESD y un proceso ADSD. Esta definición es importante en las demostraciones posteriores debido a que será invocada para comprobar la equivalencia local entre un estado LeGESD y un proceso ADSD.

Definición 6.18. *Dos variables i y j son compatibles si están definidas sobre el mismo dominio.*

Antes de proporcionar la siguiente definición, es conveniente recordar que la Definición 5.11 indica que un estado LeGESD es un símbolo gráfico que representa la condición actual de ejecución de un diagrama de comportamiento o de comunicación, mientras que la Definición 6.2 indica que un proceso ADSD es un estado computacional que puede interactuar con su ambiente a través de un conjunto de transiciones.

Definición 6.19. *Un estado LeGESD y un proceso ADSD son localmente equivalentes si existe una función biyectiva £, tal que dado un predicado arbitrario γ sobre las variables Γ que componen el estado LeGESD, se cumple que $\gamma(\Gamma) \Leftrightarrow \gamma(£(\Gamma))$.*

A continuación, en los Teoremas 6.1 a 6.5 estableceremos la equivalencia local existente entre la representación gráfica de los estados LeGESD y la representación algebraica de los procesos ADSD.

Así mismo, sean α y β predicados sobre variables $\in V_D(X)$, y α' y β' predicados sobre variables $\in V_\Im(X')$ tal que $\alpha(a_1,a_2,\ldots,a_n) \Leftrightarrow \alpha'(£(a_1),£(a_2),\ldots,£(a_n))$ y $\beta(a_1,a_2,\ldots,a_n) \Leftrightarrow \beta'(£(a_1),£(a_2),\ldots,£(a_n))$, donde £ es una función biyectiva $£{:}V_D \rightarrow V_\Im$.

El Teorema 6.1 afirma que es posible obtener la equivalencia local entre la representación gráfica de una transición de un estado LeGESD X y el proceso ADSD X' con un operador $\rightarrow$ en el álgebra de procesos ADSD (ver Tabla 6.1). Esta equivalencia local permite asociar una transición (controlada por la precondición α y la post-condición β) de X con su correspondiente X' sobre el cual actuará $\rightarrow$. La demostración del teorema consiste de dos partes. La primera parte prueba que la equivalencia local entre el dominio gráfico de LeGESD y el dominio algebraico de ADSD es posible siempre y cuando exista una función que mapee cada una de las variables asociadas a α y β de la transición de X con las variables correspondientes (en α' y β') asociadas al operador $\rightarrow$ de X'. La segunda parte prueba que la equivalencia local entre el dominio algebraico de ADSD y el dominio gráfico de LeGESD también existe utilizando la misma función de mapeo en sentido contrario.

Teorema 6.1. *Sea $X \in D$ y $X' \in \Im$ tal que X y X' son localmente equivalentes. Entonces $\Delta(X)_{\alpha,\beta} \Leftrightarrow T(X')^{\alpha'} \rightarrow {}^{\beta'}$.*

Demostración. La demostración del teorema consiste de dos partes, cada una correspondiente a un sentido de la doble implicación.

Primera parte. $\Delta(X)_{\alpha,\beta} \Rightarrow T(X')^{\alpha'} \rightarrow {}^{\beta'}$. Aplicando la Definición 6.19 se tiene que $\Upsilon_D(X) \wedge \alpha \rightarrow \beta \Rightarrow \Upsilon_\Im(X') \wedge \alpha' \rightarrow \beta'$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha \rightarrow \alpha'$ y $\beta \rightarrow \beta'$. Entonces $\Upsilon_D(X) \rightarrow \Upsilon_\Im(X')$. Por lo tanto $(\Upsilon_\Im(X')) \wedge (\alpha') \rightarrow (\beta')$.

Segunda parte. $T(X')^{\alpha'} \rightarrow {}^{\beta'} \Rightarrow \Delta(X)_{\alpha,\beta}$. De forma análoga al caso anterior, podemos aplicar la Definición 6.19 y obtener $\Upsilon_{\Im}(X') \wedge \alpha' \rightarrow \beta' \Rightarrow \Upsilon_D(X) \wedge \alpha \rightarrow \beta$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha' \rightarrow \alpha$ y $\beta' \rightarrow \beta$. Entonces $\Upsilon_{\Im}(X') \rightarrow \Upsilon_D(X)$. Por lo tanto $(\Upsilon_D(X)) \wedge (\alpha) \rightarrow (\beta)$. Lo que concluye la demostración. □

El Teorema 6.2 afirma que es posible obtener la equivalencia local entre la representación gráfica de la elección de una transición (a partir de un conjunto de transiciones) de un estado LeGESD X y el proceso ADSD X' con un operador $+$ en el álgebra de procesos ADSD (ver Tabla 6.1). Esta equivalencia local permite asociar las diversas transiciones (controlada por la precondición α y la post-condición β) de X con su correspondiente X' sobre el cual actuará $+$. La demostración del teorema consiste de dos partes. La primera parte prueba que la equivalencia local entre el dominio gráfico de LeGESD y el dominio algebraico de ADSD es posible siempre y cuando exista una función que mapee cada una de las variables asociadas a α y β de las posibles transiciones de X con las variables correspondientes (en α' y β') asociadas a $\rightarrow$ de X' dentro de $+$. La segunda parte prueba que la equivalencia local entre el dominio algebraico de ADSD y el dominio gráfico de LeGESD también existe utilizando la misma función de mapeo en sentido contrario.

Teorema 6.2. *Sea $X \in D$ y $X' \in \Im$ tal que X y X' son localmente equivalentes y sean z_1 y z_2 conjuntos de valores diferentes y mutuamente excluyentes asociados con las variables $a_1, a_2, \ldots, a_n$. Entonces $\Delta(X)_{\alpha}\Sigma_{\beta}^{\beta} \Leftrightarrow T(X')^{\alpha'} \rightarrow {}^{\beta'} + T(X')^{\alpha'} \rightarrow {}^{\beta'}$.*

Demostración. La demostración del teorema consiste de dos partes, cada una correspondiente a un sentido de la doble implicación. Primera parte. $\Delta(X)_{\alpha}\Sigma_{\beta}^{\beta} \Rightarrow T(X')^{\alpha'} \rightarrow {}^{\beta'} + T(X')^{\alpha'} \rightarrow {}^{\beta'}$. Aplicando la Definición 6.19 se tiene que $(\Upsilon_D(X) \wedge \alpha \wedge z_1 \rightarrow \beta) \vee (\Upsilon_D(X) \wedge \alpha \wedge z_2 \rightarrow \beta) \Rightarrow (\Upsilon_{\Im}(X') \wedge \alpha' \wedge z_1 \rightarrow \beta') \vee (\Upsilon_{\Im}(X') \wedge \alpha' \wedge z_2 \rightarrow \beta')$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha \rightarrow \alpha'$ y $\beta \rightarrow \beta'$. Entonces $\Upsilon_D(X) \rightarrow \Upsilon_{\Im}(X')$. Por lo tanto $((\Upsilon_{\Im}(X')) \wedge (\alpha') \wedge z_1 \rightarrow (\beta')) \vee ((\Upsilon_{\Im}(X')) \wedge (\alpha') \wedge z_2 \rightarrow (\beta'))$.

Segunda parte. $T(X')^{\alpha'} \rightarrow {}^{\beta'} + T(X')^{\alpha'} \rightarrow {}^{\beta'} \Rightarrow \Delta(X)_{\alpha}\Sigma_{\beta}^{\beta}$. De forma análoga al caso anterior, podemos aplicar la Definición 6.19 y obtener $(\Upsilon_{\Im}(X') \wedge \alpha' \wedge z_1 \rightarrow \beta') \vee (\Upsilon_{\Im}(X') \wedge \alpha' \wedge z_2 \rightarrow \beta') \Rightarrow (\Upsilon_D(X) \wedge \alpha \wedge z_1 \rightarrow \beta) \vee (\Upsilon_D(X) \wedge \alpha \wedge z_2 \rightarrow \beta)$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha' \rightarrow \alpha$ y $\beta' \rightarrow \beta$. Entonces $\Upsilon_{\Im}(X') \rightarrow \Upsilon_D(X)$. Por lo tanto $((\Upsilon_D(X)) \wedge (\alpha) \wedge z_1 \rightarrow \beta)) \vee ((\Upsilon_D(X)) \wedge (\alpha) \wedge z_2 \rightarrow (\beta))$. Lo que concluye la demostración. □

El Teorema 6.3 afirma que es posible obtener la equivalencia local entre la representación gráfica de un estado LeGESD X usada para el envío de mensajes y el proceso ADSD X' con un operador Δ en el álgebra de procesos ADSD (ver Tabla 6.1). Esta equivalencia local permite asociar una transición (controlada por la precondición α, β y m y la post-condición α, β y m) de X con su correspondiente X' sobre el cual actuará Δ. La demostración del teorema consiste de dos partes. La primera parte prueba que la equivalencia local entre el dominio gráfico de LeGESD y el dominio algebraico de ADSD es posible siempre y cuando exista una función que mapee cada una de las variables asociadas a α, β y m de la transición de X con las variables correspondientes (en α', β' y m') asociadas a Δ de X'. La segunda parte prueba que la equivalencia local entre el dominio algebraico de ADSD y el dominio gráfico de LeGESD también existe utilizando la misma función de mapeo en sentido contrario.

Teorema 6.3. *Sea $X \in D$ y $X' \in \Im$ tal que X y X' son localmente equivalentes y sea m un predicado sobre variables $\in V_D(X)$ y m' un predicado sobre variables $\in V_\Im(X')$ tal que $m(a_1, a_2, \ldots, a_n) \Leftrightarrow m'(\pounds(a_1), \pounds(a_2), \ldots, \pounds(a_n))$. Entonces $\Delta(X)_{\alpha,\beta,m} \Leftrightarrow T(X')\Delta^{\alpha',\beta',m'}$.*

Demostración. La demostración del teorema consiste de dos partes, cada una correspondiente a un sentido de la doble implicación. Primera parte. $\Delta(X)_{\alpha,\beta,m} \Rightarrow T(X')\Delta^{\alpha',\beta',m'}$. Aplicando la Definición 6.19 se tiene que $\Upsilon_D(X) \wedge \alpha \wedge m \rightarrow \beta \Rightarrow \Upsilon_\Im(X') \wedge \alpha' \wedge m' \rightarrow \beta'$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha \rightarrow \alpha'$, $\beta \rightarrow \beta'$ y $m \rightarrow m'$. Entonces $\Upsilon_D(X) \rightarrow \Upsilon_\Im(X')$. Por lo tanto $(\Upsilon_\Im(X')) \wedge (\alpha') \wedge (m') \rightarrow (\beta')$.

Segunda parte. $T(X')\Delta^{\alpha',\beta',m'} \Rightarrow \Delta(X)_{\alpha,\beta,m}$. De forma análoga al caso anterior, podemos aplicar la Definición 6.19 y obtener $\Upsilon_\Im(X') \wedge \alpha' \wedge m' \rightarrow \beta' \Rightarrow \Upsilon_D(X) \wedge \alpha \wedge m \rightarrow \beta$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $\alpha' \rightarrow \alpha$, $\beta' \rightarrow \beta$ y $m' \rightarrow m$. Entonces $\Upsilon_\Im(X') \rightarrow \Upsilon_D(X)$. Por lo tanto $(\Upsilon_D(X)) \wedge (\alpha) \wedge (m) \rightarrow (\beta)$. Lo que concluye la demostración. □

El Teorema 6.4 afirma El Teorema 6.4 afirma que es posible obtener la equivalencia local entre la representación gráfica de un estado LeGESD X usada para la recepción de mensajes y el proceso ADSD X' con los operadores $[/]$ y Δ en el álgebra de procesos ADSD (ver Tabla 6.1). Esta equivalencia local permite asociar una transición (controlada por la precondición m, α y β y la post-condición m, α y β) de X con su correspondiente X' sobre el cual actuará $[/]$ y Δ. La demostración del teorema consiste de dos partes. La primera parte prueba que la equivalencia local entre el dominio gráfico de LeGESD y el dominio algebraico de ADSD es posible siempre y cuando exista una función que mapee cada una de las variables asociadas a m y m de la transición de X con las variables correspondientes (en m' y m') asociadas a $[/]$ y Δ de X'. La segunda parte prueba que la equivalencia local entre el dominio algebraico de ADSD y el dominio gráfico de LeGESD también existe utilizando la misma función de mapeo en sentido contrario.

Teorema 6.4. *Sea $X \in D$ y $X' \in \Im$ tal que X y X' son localmente equivalentes y sea m un predicado sobre variables $\in V_D(X)$ y m' un predicado sobre variables $\in V_\Im(X')$ tal que $m(a_1, a_2, \ldots, a_n) \Leftrightarrow m'(\pounds(a_1), \pounds(a_2), \ldots, \pounds(a_n))$ con $m, m' \neq \varnothing$. Entonces $\Delta(X)^{m,\alpha,\beta} \Leftrightarrow [T(X')/-]_{m'}\Delta^{\alpha',\beta'}$.*

Demostración. La demostración del teorema consiste de dos partes, cada una correspondiente a un sentido de la doble implicación. Primera parte. $\Delta(X)^{m,\alpha,\beta} \Rightarrow [T(X')/-]m'\Delta^{\alpha',\beta'}$. Aplicando la Definición 6.19 se tiene que $\Upsilon_D(X) \wedge m \wedge \alpha \rightarrow \beta \Rightarrow \Upsilon_\Im(X') \wedge m' \wedge \alpha' \rightarrow \beta'$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $m \rightarrow m'$, $\alpha \rightarrow \alpha'$ y $\beta \rightarrow \beta'$. Entonces $\Upsilon_D(X) \rightarrow \Upsilon_\Im(X')$. Por lo tanto $(\Upsilon_\Im(X')) \wedge (m') \wedge (\alpha') \rightarrow (\beta')$.

Segunda parte. $[T(X')/-]m'\Delta^{\alpha',\beta'} \Rightarrow \Delta(X)^{m,\alpha,\beta}$. De forma análoga al caso anterior, podemos aplicar la Definición 6.19 y obtener $\Upsilon_\Im(X') \wedge m' \wedge \alpha' \rightarrow \beta' \Rightarrow \Upsilon_D(X) \wedge m \wedge \alpha \rightarrow \beta$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $m' \rightarrow m$, $\alpha' \rightarrow \alpha$ y $\beta' \rightarrow \beta$. Entonces $\Upsilon_\Im(X') \rightarrow \Upsilon_D(X)$. Por lo tanto $(\Upsilon_D(X)) \wedge (m) \wedge (\alpha) \rightarrow (\beta)$. Lo que concluye la demostración. □

El Teorema 6.5 afirma que es posible obtener la equivalencia local entre la representación gráfica de un estado LeGESD X usado en la recursión y el proceso ADSD X' con un operador *rec* en el álgebra de procesos ADSD (ver Tabla 6.1). Esta equivalencia local permite asociar una transición (controlada por la invariante de recursión I) de X con su correspondiente X' sobre el cual actuará *rec*. La demostración del teorema consiste de dos partes. La primera parte prueba que la equivalencia local entre el dominio gráfico de LeGESD y el dominio algebraico de ADSD es posible siempre y cuando exista una función que mapee cada una de las variables asociadas a I

de la transición de X con las variables correspondientes (en I') asociadas a *rec* de X'. La segunda parte prueba que la equivalencia local entre el dominio algebraico de ADSD y el dominio gráfico de LeGESD también existe utilizando la misma función de mapeo en sentido contrario.

Teorema 6.5. *Sea $X \in D$ y $X' \in \Im$ tal que X y X' son localmente equivalentes y sea I una invariante de recursión sobre variables $\in V_D(X)$ e I' una invariante de recursión sobre variables $\in V_\Im(X')$ tal que $I(a_1, a_2, \ldots, a_n) \Leftrightarrow I'(\pounds(a_1), \pounds(a_2), \ldots, \pounds(a_n))$. Entonces $\Delta(X)_{I,I} \Leftrightarrow rec\ I'.T(X')$.*

Demostración. La demostración del teorema consiste de dos partes, cada una correspondiente a un sentido de la doble implicación. Primera parte. $\Delta(X)_{I,I} \Rightarrow rec\ I'.T(X')$. Aplicando la Definición 6.19 se tiene que $\Upsilon_D(X) \wedge I \rightarrow I \Rightarrow \Upsilon_\Im(X') \wedge I' \rightarrow I'$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $I \rightarrow I'$. Entonces $\Upsilon_D(X) \rightarrow \Upsilon_\Im(X')$. Por lo tanto $(\Upsilon_\Im(X')) \wedge (I') \rightarrow (I')$.

Segunda parte. $rec\ I'.T(X') \Rightarrow \Delta(X)_{I,I}$. De forma análoga al caso anterior, podemos aplicar la Definición 6.19 y obtener $\Upsilon_\Im(X') \wedge I' \rightarrow I' \Rightarrow \Upsilon_D(X) \wedge I \rightarrow I$. Ahora, aplicando las Definiciones 6.18 y 6.19 se tiene que $I' \rightarrow I$. Entonces $\Upsilon_\Im(X') \rightarrow \Upsilon_D(X)$. Por lo tanto $(\Upsilon_D(X)) \wedge (I) \rightarrow (I)$. Lo que concluye la demostración. □

La Definición 6.20 tiene el propósito de explicar el significado del termino estado LeGESD alcanzable, el cual será usado para delimitar al Teorema 6.6.

Definición 6.20. *Un estado LeGESD alcanzable en un diagrama de comportamiento o comunicación en LeGESD D válido es un estado al cual se puede llegar partiendo del estado inicial y siguiendo las transiciones en D.*

El Teorema 6.6 afirma que es posible obtener la equivalencia local de la representación gráfica de un estado LeGESD a partir de la generación correspondiente de un proceso ADSD utilizando el Algoritmo 1. Este teorema es importante debido a que permite establecer una equivalencia local entre el espacio gráfico de LeGESD y el espacio algebraico de ADSD.

Teorema 6.6. *Sea `D` un diagrama de comportamiento o comunicación en LeGESD válido entonces TLA(`D`) genera un proceso en ADSD para cada estado LeGESD alcanzable en `D` siendo éstos localmente equivalentes.*

Demostración. Procederemos por contradicción y supondremos que no se genera un proceso en ADSD para un estado LeGESD alcanzable. Llamaremos a este estado e. Como es alcanzable existe un camino p que va del estado inicial i a e. Sea x el primer estado en p que no se ha generado y sea x' su vecino inmediato en p que sí ha sido generado por la línea 7 del Algoritmo 2 y sea c la precondición y post-condición en p entre x' y x. Dado que x' ha sido generado también ha sido transformado. Como x' ha sido transformado también ha transformado a c mediante uno de los casos del Algoritmo 3 entonces la transformación es localmente equivalente. Como x' ha sido generado dentro de p y x es también alcanzable en p entonces x también será generado por el Algoritmo 2. Con base en x se observa que la condición de término del algoritmo no se cumple por lo que llegamos a una contradicción. □

El Teorema 6.7 afirma que es posible obtener una ecuación de transición en el álgebra ADSD a partir de cada transición entre estados LeGESD alcanzables. Este teorema es importante debido a que permite establecer una correspondencia entre una transición en LeGESD y una ecuación en el álgebra ADSD.

Teorema 6.7. *Sea D un diagrama de comportamiento o comunicación en LeGESD válido entonces TLA(D) genera una ecuación de transición verdadera en ADSD por cada transición entre estados LeGESD alcanzables en D si y sólo si la precondición y post-condición de la transición son verdaderas.*

Demostración. Sea x un estado visitado de acuerdo al Teorema 6.6 y sea t una transición del estado x y sea c la precondición y post-condición asociadas a t. Partiendo del resultado obtenido en la demostración del Teorema 6.6 tenemos que x también ha sido transformado. Por lo tanto c son verdaderas generando una ecuación de transición verdadera para la transición t. □

6.3.2. Teorema de equivalencia gráfico-algebraica

En esta sección se presenta el Teorema de equivalencia gráfico-algebraica que establece que la especificación gráfica G de un sistema distribuido por medio de un conjunto de diagramas LeGESD válidos, es equivalente a la especificación algebraica en ADSD generada por el Algoritmo 1, cuando es alimentado con G. Esta *equivalencia* establece que un predicado arbitrario sobre las variables que componen el estado de un sistema distribuido especificado con LeGESD es verdadero si y sólo sí el mismo predicado sobre las variables que definen al proceso ADSD que le corresponde a dicho estado LeGSD es también verdadero. De esta forma, ambas representaciones son equivalentes en el sentido de que cualquier predicado que establezca una propiedad de alguna representación se puede extrapolar a la otra. Así, el Teorema de equivalencia gráfico-algebraica permite utilizar el formalismo del álgebra de procesos para realizar demostraciones acerca del comportamiento de un sistema especificado con LeGESD. En este trabajo, es de particular interés realizar demostraciones de propiedades de *seguridad* sobre las especificaciones gráficas de un sistema concurrente o distribuido. Para ilustrar lo anterior, en el Capítulo 7 se especifican a detalle dos sistemas utilizando LeGESD y se demuestran sus principales propiedades de seguridad utilizando ADSD.

Teorema 6.8. *Sean $G = \{D_1, D_2, \ldots, D_n\}$ un conjunto de diagramas de comportamiento o comunicación en LeGESD válidos que describen n procesos que se ejecutan de manera concurrente, $X_\eta \in D_i$ con $i \in \{1, 2, \ldots, n\}$ un estado LeGESD alcanzable y X'_η el proceso generado por TLA(G) al transformar X_η. Entonces un predicado γ arbitrario sobre las variables que componen el estado X_η es verdadero si y sólo si el predicado $\gamma(\pounds(X'_\eta))$ es también verdadero.*

Demostración. Debido a que G es un conjunto de diagramas de comportamiento o comunicación en LeGESD válidos, D_i posee un estado inicial al cuál denotaremos por X_0. Por otro lado, al ser X_η alcanzable, entonces existe un camino $C_{X_0}^{X_\eta} = X_0, X_1, \ldots, X_\eta$ compuesto por estados LeGESD válidos X_j con $j \in \{0, 1, \ldots, \eta\}$. Por el Teorema 6.6, sabemos que TLA(G) genera un único proceso ADSD por cada estado alcanzable en G, por lo tanto, por cada estado en $C_{X_0}^{X_\eta}$ habrá un proceso en ADSD. Denotaremos por X'_j al proceso ADSD generado por TLA(G) al analizar el estado X_j. Por otro lado, por el Teorema 6.7, sabemos que para toda transición $X_j \rightsquigarrow X_{j+1} \in D_i$ el Algoritmo 3 también genera una ecuación de transición $X'_j \rightsquigarrow X'_{j+1}$ en ADSD. Para el resto de la demostración procederemos por inducción.

Hipótesis de Inducción (HI): Todo estado $X_i \in G$, alcanzable en n transiciones, es localmente equivalente a un proceso ADSD X'_i alcanzable por medio de n ecuaciones de transición.

Caso Base: Por las líneas 2-15 del Algoritmo 3, y la Definición 6.19 sabemos que el estado X_0 y el proceso X'_0 son localmente equivalentes. Lo anterior debido a que el Algoritmo 4 definió una

función biyectiva $\pounds$ en la que por cada variable v definida en el estado X_j, se generó una única variable v' en el proceso X'_j, donde el dominio de las variables v' y v es el mismo (en otras palabras, $v' = \pounds(v)$ para toda variable v' generada por TLA(G)), y ambos conjuntos de variables fueron inicializados a los mismos valores.

Suponemos ahora que la HI es verdadera para cualquier estado alcanzable en k transiciones y sean X_{k+1} un estado alcanzable en $k+1$ transiciones y X'_{k+1} el proceso creado por TLA(G) cuando analizó el estado X_{k+1}. Ahora, que el estado X_{k+1} sea alcanzable en $k+1$ transiciones, implica que existe un estado X_k alcanzable en k transiciones y que además existe una transición $X_k \rightsquigarrow X_{k+1} \in G$. Como ya argumentamos, por el Teorema 6.7, sabemos que el Algoritmo 3 generó la ecuación de transición $X'_k \rightsquigarrow X'_{k+1}$ correspondiente a la transición $X_k \rightsquigarrow X_{k+1}$. Por la HI sabemos que el estado X_k y el proceso X'_k son localmente equivalentes. Para cualquier variable v definida en el estado X_k tenemos dos casos, que haya sido modificada durante la transición $X_k \rightsquigarrow X_{k+1}$ o que no.

Para el caso en que la transición haya modificado el valor de v, sabemos que dicho valor fue modificado de tal manera que antes de la transición la postcondición β_{X_k} era verdadera, y que después de la transición la precondición $\alpha_{X_{k+1}}$ es verdadera. Ahora, como X_k y X'_k son localmente equivalentes y por los Teoremas 6.1 a 6.5 se cumple que $\alpha_{X_k} \Leftrightarrow \alpha_{X'_k}$ y $\beta_{X_k} \Leftrightarrow \beta_{X'_k}$. De igual forma, por la ecuación de transición $X'_k \rightsquigarrow X'_{k+1}$ tenemos que $\beta_{X'_k} \rightarrow \alpha_{X'_{k+1}}$ y por lo tanto podemos concluir que $\alpha_{X_{k+1}}$ si y sólo si $\alpha_{X'_{k+1}}$. Ahora, como v forma parte del predicado $\alpha_{X_{k+1}}$ y v' forma parte del predicado $\alpha_{X'_{k+1}}$ se cumple que $v' = \pounds(v)$ en el estado X_{k+1} y el proceso X'_{k+1} respectivamente. Para el segundo caso, si la variable v no fue modificada por la transición, tampoco v' fue modificada por la ecuación de transición y por lo tanto $v' = \pounds(v)$ en el estado X_{k+1} y el proceso X'_{k+1} respectivamente. Finalmente, como los valores de v y v' son los mismos para todas las variables involucradas en el estado X_{k+1} y el proceso X'_{k+1} respectivamente podemos concluir que ambos estados son localmente equivalentes.

Como X_{k+1} fue elegido de manera arbitraria podemos concluir que para cualquier estado $X_\eta \in D_i$ con $i \in \{1,2,\ldots,n\}$ y su respectivo proceso X'_η se cumple que un predicado γ arbitrario sobre las variables que componen el estado X_η es verdadero si y sólo si el predicado $\gamma(\pounds(X'_\eta))$ es también verdadero □

Con esto concluye la presentación de ADSD, el cual conforma la semántica del lenguaje LeGESD y que complementa el desarrollo de la propuesta de solución que propone la presente tesis. En el siguiente capítulo, se muestran dos casos de estudio de la aplicación de LeGESD en la especificación de sistemas distribuidos.

7 Casos de estudio

La aplicación de LeGESD a la especificación de un sistema distribuido se realiza a través de dos casos de estudios. Estos casos se relacionan con el sistema distribuido productor-consumidor y con la cena de criptógrafos. El objetivo del presente capítulo es desarrollar los casos de estudio especificándolos con LeGESD.

7.1. Introducción

En este capítulo se presentan dos casos de estudio relacionados con dos sistemas distribuidos especificados con el lenguaje LeGESD, tanto en su parte visual como en su transformación al álgebra de procesos ADSD. Adicionalmente, se muestra a partir de su álgebra de procesos ADSD, la verificación de las propiedades de seguridad en la especificación de cada uno de los sistemas.

Es importante en este momento recordar la hipótesis planteada en la presente tesis, debido a que con el desarrollo de estos casos de estudio se podrá validar o desechar la hipótesis planteada. La hipótesis es:

Es posible definir un lenguaje gráfico para la especificación de sistemas distribuidos que incorpore como su semántica formal un álgebra de procesos, de tal manera que la especificación gráfica de

un sistema distribuido sea equivalente a su especificación algebraica permitiendo la verificación de propiedades de seguridad de la especificación gráfica a partir de su especificación algebraica.

Los casos de estudio propuestos permiten observar de manera clara la forma en que se especifica un sistema distribuido utilizando LeGESD a través de sus diversas etapas. Los casos de estudio a especificar son: el problema clásico del productor-consumidor con búfer limitado y el problema de la cena de criptógrafos.

El problema del productor-consumidor con búfer limitado fue planteado por C. A. R. Hoare [59]. Este problema es de relevancia en este trabajo de tesis debido a que, como se mencionó en el capítulo 5 en las restricciones para la comunicación en sistemas distribuidos que establece LeGESD, el modelo de comunicación a especificar en LeGESD es exclusivamente del tipo productor-consumidor, es decir, siempre que se produzca un mensaje éste deberá ser consumido. Este modelo es importante en los sistemas distribuidos, debido a que el modelo cliente-servidor se ajusta a este modelo de comunicación.

El sistema consiste de dos componentes, el productor y el consumidor, los cuales se comunican a través de un búfer compartido de tamaño limitado. El productor genera un dato, colocándolo en el búfer compartido y enseguida genera el siguiente dato a colocar en el búfer compartido. Así continúa su trabajo hasta una condición de terminación. Al mismo tiempo, el consumidor recoge el dato, es decir remueve el dato del búfer compartido, un dato a la vez. El problema consiste en que el productor puede seguir generando datos aún cuando el límite del búfer se haya alcanzado, es decir que el búfer este lleno; por otra parte el consumidor puede intentar recoger un dato del búfer compartido aún cuando no se haya generado algún dato, es decir que el búfer este vacío.

La solución al problema del productor-consumidor consiste en asegurar que el productor no agregará algún dato si el búfer compartido se encuentra lleno; a la vez se debe asegurar que el consumidor no intente remover algún dato si el búfer compartido se encuentra vacío.

El problema de la cena de criptógrafos fue planteado por D. Chaum [60], y consiste en tres criptógrafos (C_1, C_2 y C_3), los cuales están sentados a la mesa en su restaurante favorito. Su camarero les informa que se han hecho arreglos para pagar la cuenta de forma anónima. Uno de los criptógrafos podría haber pagado la cena, o podría haber sido pagada por la NSA (Agencia de Seguridad Nacional de E.U.A.). Los tres criptógrafos respetan el derecho de cada uno de hacer el pago en el anonimato, pero se preguntan si la NSA habrá sido quien pagó o si pagó alguno de ellos.

La solución al problema de la cena de criptógrafos consiste en que cada criptógrafo lanzará una moneda no sesgada al aire (volado) de manera privada, el resultado del volado lo compartirá con el criptógrafo a su derecha, de manera que sólo ellos dos pueden ver el resultado. A continuación, cada criptógrafo anuncia en voz alta si los volados cayeron del mismo lado o de lados diferentes. Si los volados son iguales, el criptográfo anunciará el resultado 0, si son distintos anunciará el resultado 1, sin embargo si un criptógrafo pagó, entonces anunciará el resultado inverso al de los volados. Finalmente, se suman los resultados anunciados, si la suma es impar entonces la cena la pagó uno de los criptógrafo, en caso contrario la NSA pagó la cena.

Para verificar que el protocolo es seguro (anonimato conservado), es necesario observar las situaciones para las cuales un criptógrafo, que no ha pagado la cuenta, desea averiguar cuál criptógrafo ha pagado. Lo anterior implica verificar que:

- Sean los volados de las dos monedas que C_2 observa los mismos, y uno de los otros criptógra-

fos anunció 1 (diferentes) y el otro anunció 0 (iguales). Si el volado de la moneda que C_2 no observa es el mismo que los volados de las monedas que él observa (caso (c) de la figura 7.1), entonces el criptógrafo que anunció 1 es el que ha pagado la cuenta. Por otra parte, si el volado de la moneda que C_2 no observa es distinto al de los volados de las monedas que él observa (caso (d) de la figura 7.1), entonces el criptógrafo que anunció 0 es el que ha pagado la cuenta. Sin embargo, como la moneda que está oculta a C_2 es no sesgada, los dos casos anteriores son equiprobables.

- Sean los volados de las dos monedas que C_1 observa distintos. Si los otros dos criptógrafos anunciaron 1 (caso (e) de la figura 7.1), entonces quien pagó la cuenta está más cercano a la moneda cuyo volado es el mismo al que C_1 no observa. Por otra parte, si los otros dos criptógrafos anunciaron 0 (caso (d) de la figura 7.1), entonces quien pagó está más cercano a la moneda cuyo volado es diferente al que C_1 no observa. Nuevamente, los dos casos anteriores son equiprobables.

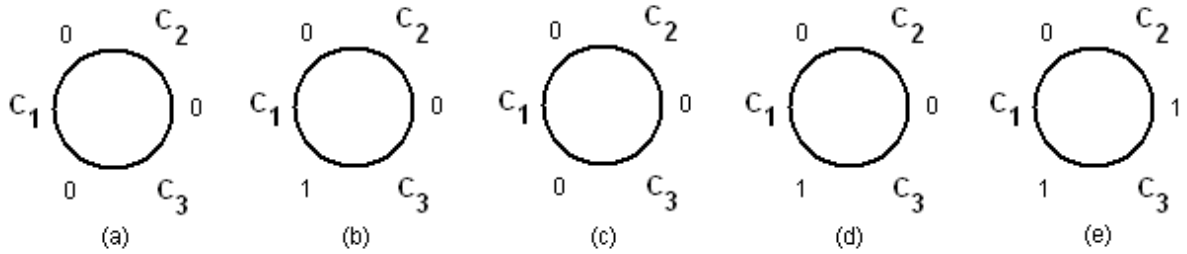


Figura 7.1: Cena de criptógrafos. (a) NSA ha pagado, suma de diferencias igual a 0. (b) NSA ha pagado, suma de diferencias igual a 2. (c) C_3 ha pagado, anuncia 1, suma de diferencias igual a 1. (d) C_3 ha pagado, anuncia 0, suma de diferencias igual a 1. (e) Similar a (d) con distinta distribución.

De esta manera para cada caso, un criptógrafo que no ha pagado la cuenta, no descubre cual de los otros dos criptógrafos ha pagado, verificándose la conservación del anonimato. Éstos son entonces los sistemas que se especificarán a través del lenguaje LeGESD.

7.2. Especificación del sistema productor-consumidor utilizando el lenguaje LeGESD

La especificación del sistema inicia con la identificación de las entidades lógicas que integran al sistema distribuido a especificar, una entidad lógica se entiende bajo la Definición 5.2 dada en el capítulo 5. Las entidades lógicas que integran al sistema son las siguientes:

- El productor.
- El consumidor.

- El búfer limitado.

La estructura de la especificación de un sistema distribuido que LeGESD propone, consiste en la idea de que la arquitectura de software global de las entidades lógicas que componen al sistema distribuido en el nivel más alto de abstracción, están integradas por:

- Vista de sistema.
- Vista de implementación.

7.2.1. Vista de sistema

La vista de sistema representa la estructura estática del sistema distribuido a especificar, que incluye los elementos estáticos del sistema y sus relaciones estáticas, de acuerdo a la Definición 5.3. La vista de sistema está integrada por dos niveles de componentes gráficos:

- Componentes de comportamiento, nombrados trabajos LeGESD (Definición 5.5).
- Componentes de comunicación, nombrados medios LeGESD (Definición 5.6).

Para el ejemplo que se está especificando, la vista de sistema está integrada por los trabajos LeGESD siguientes:

- Produc.
- Búf.
- Consum.

Estos trabajos LeGESD se llevan a cabo de forma concurrente, comunicándose de forma distribuida para realizar sus actividades designadas. Los trabajos LeGESD listados previamente son descritos a través de sus respectivos diagrama de comportamiento que se muestran en la Figura 7.2 (lado derecho), un diagrama de comportamiento se entiende bajo la Definición 5.7. En cada uno de estos diagramas se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD correspondiente, el cual contiene un conjunto de símbolos gráficos: estados y enlaces. Los estados y enlaces se entienden bajo las Definiciones 5.11 y 5.12 respectivamente.

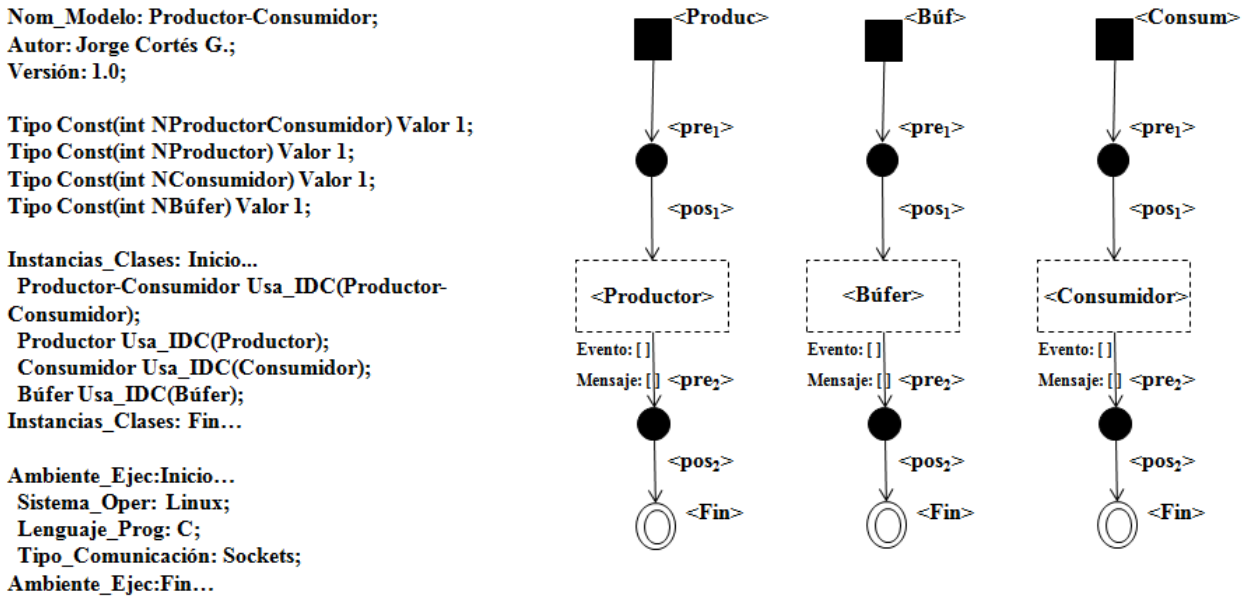


Figura 7.2: Vista de sistema y de implementación del sistema productor-consumidor utilizando el lenguaje LeGESD.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento de cada trabajo LeGESD, se utilizan las Tablas 7.1 - 7.3 para agrupar la precondición (que activa a la transición) y la post-condición (generada después de la transición) relacionadas con cada transición. Cada una de las tablas asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, void *Ep=null, const N=10, int Ns=N, msj msjProd1=null, msj msjProd2=null, int v=0>
pos_1	Post-condición	<2, Ep=null, Ns=N, N=10, msjProd1=null, msjProd2=null, v=0>
pre_2	Precondición	<1, v!=0>
pos_2	Post-condición	<2, v=1>

Tabla 7.1: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Produc.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, msj msjBuf=null, msj msjBuf1=null, msj msjBuf2=null, void *Ep=null, const N=10, int Ns=N, int S=0, int t=1, int v=0>
pos_1	Post-condición	<2, msjBuf=null, msjBuf1=null, msjBuf2=null, *Ep=null, N=10, Ns=N, S=0, t=1, v=0>
pre_2	Precondición	<1, v!=0>
pos_2	Post-condición	<2, v=1>

Tabla 7.2: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Búf.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, void *Ec=null, int Ne=0, void *I=null, msj msjCons=null, int v=0>
pos_1	Post-condición	<2, Ec=null, Ne=0, I=null, msjCons=null, v=0>
pre_2	Precondición	<1, v!=0>
pos_2	Post-condición	<2, v=1>

Tabla 7.3: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Consum.

Los estados utilizados para especificar los diagramas de comportamiento de los trabajos LeGESD del Productor-Consumidor son los siguientes:

- Estado Inicio nombrado Produc.
- Estado Inicio nombrado Búf.
- Estado Inicio nombrado Consum.
- Estado Compuesto nombrado Productor.
- Estado Compuesto nombrado Búfer.
- Estado Compuesto nombrado Consumidor.
- Estados Transición.
- Estados Fin nombrados Fin.

Los enlaces utilizados para especificar los diagramas de comportamiento de los trabajos LeGESD del Productor-Consumidor son del tipo enlaces simples etiquetados. Un enlace simple se entiende bajo la Definición 5.13.

Los trabajos LeGESD del Productor-Consumidor mostrados en la vista de sistema, representan el nivel más alto de la composición jerárquica de la especificación. El siguiente nivel de la composición jerárquica está dado por los estados Compuestos (Productor, Búfer y Consumidor) contenidos en sus respectivos diagramas de comportamiento.

En la especificación de cada uno de los estados que conforman los diagramas de comportamiento de los trabajos LeGESD del Productor-Consumidor se tiene:

- Estados Inicio: tienen asociada una etiqueta que corresponde al nombre del estado (Produc, Búf y Consum) y cumplen con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Compuesto Productor: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Productor, no se establecen declaraciones ni eventos ni mensajes asociados al estado, y cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Compuesto Consumidor: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Consumidor, no se establecen declaraciones de acciones ni eventos ni mensajes asociados al estado, y cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Compuesto Búfer: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Búfer, no se establecen declaraciones de acciones ni eventos ni mensajes asociados al estado, y cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estados Transición: no se establecen declaraciones de control y cumplen con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estados Fin: tienen asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin, y cumplen con los requisitos de especificación establecidos en el capítulo 5 para este estado.

En la especificación de cada uno de los enlaces simples etiquetados que conforman los diagramas de comportamiento de los trabajos LeGESD del Productor-Consumidor se tiene:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Inicio y el estado Productor tienen como etiquetas la precondición <1, void *Ep=null, const N=10, int Ns=N, msj msjProd1=null, msj msjProd2=null, int v=0> y la post-condición <2, Ep=null, N=10, Ns=N, msjProd1=null, msjProd2=null, v=0>. Ep, Ns, $msjProd1$, $msjProd2$ y v son variables mientras que N es una constante, las cuales se declaran utilizando el pseudocódigo LeGESD que se presentó en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.
- Los enlaces correspondientes a la transición (estado Transición) entre el estado Productor y el estado Fin tienen como etiquetas la precondición <1, v!=0> y la post-condición <2, v=1>.
- Los enlaces correspondientes a la transición (estado Transición) entre el estado Inicio y el estado Búfer tienen como etiquetas la precondición <1, msj msjBuf=null, msj msjBuf1=null, msj msjBuf2=null, void *Ep=null, const N=10, int Ns=N, int S=0, int t=1, int v=0> y la post-condición <2, msjBuf=null, msjBuf1=null, msjBuf2=null, *Ep=null, N=10, Ns=N, S=0,

t=1, v=0>. $msjBuf$, $msjBuf1$, $msjBuf2$, Ep, Ns, S, t y v son variables mientras que N es una constante, las cuales se declaran utilizando el pseudocódigo LeGESD que se presentó en el capítulo 5. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Búfer y el estado Fin tienen como etiquetas la precondición <1, v!=0> y la post-condición <2, v=1>.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Inicio y el estado Consumidor tienen como etiquetas la precondición <1, void *Ec=null, int Ne=0, void *I=null, msj msjCons=null, int v=0> y la post-condición <2, Ec=null, Ne=0, I=null, msjCons=null, v=0>. Ec, Ne, I, $msjCons$ y v son variables que se declaran utilizando el pseudocódigo LeGESD que se presentó en el capítulo 5. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Consumidor y el estado Fin tienen como etiquetas la precondición <1, v!=0> y la post-condición <2, v=1>.

Una vez explicados cada uno de los estados y enlaces que componen los diagramas de comportamiento del trabajo LeGESD del Productor-Consumidor, a continuación de describe el comportamiento del trabajo:

1. Comenzando en el estado Inicio de los diagramas de comportamiento de Produc, Búf y Consum, es posible cambiar de estado hacia el estado Productor, Búfer y Consumidor respectivamente. Este cambio de estado se lleva a cabo de manera concurrente, por lo cual se tienen tres flujos de comportamiento.

   Primer flujo de comportamiento (trabajo LeGESD Produc).

2. Para cambiar del estado Produc hacia Productor se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables $*Ep, Ns, msjProd1, msjProd2, v$, y la constante N, las cuales deben haberse declarado e inicializado para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores. El significado de estas variables es el siguiente:
   Ep:= Elemento producido.
   Ns:= Número de espacios en el búfer.
   N:= Tamaño del búfer.
   msjProd1, msjProd2:= Mensajes de comunicación.
   v:= Valor de terminación del trabajo LeGESD.

3. Para cambiar del estado Productor hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona

con la verificación del valor de la variable v, la cual debe ser diferente de 0 para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con el valor asignado a la variable anterior.

Segundo flujo de comportamiento (trabajo LeGESD Búf).

2. Para cambiar del estado Búf hacia Búfer se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables $msjBuf, msjBuf1, msjBuf2, Ep, Ns, S, t, v$ y la constante N, las cuales deben haberse declarado e inicializado para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores. El significado de estas variables es el siguiente:
msjBuf:= Mensaje recibido del medio de comunicación que contiene el valor de Ns.
msjBuf1:= Mensaje recibido del medio de comunicación que contiene el valor de I.
msjBuf2:= Mensaje recibido del medio de comunicación que contiene el valor de Ne.
v:= Valor de terminación del trabajo LeGESD.

3. Para cambiar del estado Búfer hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable v, la cual debe ser diferente de 0 para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con el valor asignado a la variable anterior.

Tercer flujo de comportamiento (trabajo LeGESD Consum).

2. Para cambiar del estado Consum hacia Consumidor se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables $*Ec, Ne, *I, msjCons, v$, las cuales deben haberse declarado e inicializado para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores. El significado de estas variables es el siguiente:
Ec:= Elemento consumido.
Ne:= Número de elementos en el búfer.
I:= Valor obtenido.
msjCons:=Mensaje de comunicación.
v:= Valor de terminación del trabajo LeGESD.

3. Para cambiar del estado Consumidor hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable v, la cual debe ser diferente de 0 para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con el valor asignado a la variable anterior.

De esta forma queda especificada la vista de sistema del ejemplo utilizando el lenguaje LeGESD, a continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica a cada diagrama de comportamiento del trabajo LeGESD del Productor-Consumidor. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición de los diagramas de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces simples etiquetados, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en las Tablas 7.4 - 7.3 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Produc)} \overset{pre_1}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_1} \text{T(Productor)} \tag{7.2.1}$$

$$\text{T(Productor)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_2} \text{T(Fin)} \tag{7.2.2}$$

$$\text{T(Búf)} \overset{pre_1}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_1} \text{T(Búfer)} \tag{7.2.3}$$

$$\text{T(Búfer)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_2} \text{T(Fin)} \tag{7.2.4}$$

$$\text{T(Consum)} \overset{pre_1}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_1} \text{T(Consumidor)} \tag{7.2.5}$$

$$\text{T(Consumidor)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_2} \text{T(Fin)} \tag{7.2.6}$$

Con esto se concluye la descripción de la vista de sistema, sin embargo ésta no es suficiente para especificar completamente el ejemplo trabajado con LeGESD. Esta vista sólo es el nivel más alto en la composición jerárquica de la especificación, para completar la especificación se deben desarrollar los diagramas de comportamiento de los trabajos LeGESD Productor, Consumidor y

Búfer que representan el siguiente nivel en la composición jerárquica de la especificación. Adicionalmente, se debe especificar el medio de comunicación distribuida a emplear en el ejemplo, mediante los diagramas de comportamiento del medio LeGESD.

Es obligatorio para estos trabajos desarrollar sus diagramas de comportamiento por tratarse de estados Compuestos, los cuales deben cumplir con la regla sintáctica 12 de LeGESD. En secciones posteriores se presentarán estos trabajos y el medio LeGESD para completar la especificación del ejemplo.

7.2.2. Vista de implementación

La vista de implementación representa las restricciones en la implementación del sistema distribuido, la cual consiste básicamente en el ambiente de ejecución del sistema a especificar de acuerdo a la Definición 5.4. La vista de implementación se compone de anotaciones textuales dentro de la vista de sistema, como se observa en la Figura 7.2 (parte izquierda). Estas anotaciones textuales consisten en palabras reservadas del pseudocódigo LeGESD (Tabla 5.1 del capítulo 5) escritas con la sintaxis dada en la tabla. Para el ejemplo se utilizan las palabras reservadas siguientes:

- Nom_Modelo. Nombra el ejemplo que se está especificando: Productor-Consumidor.
- Autor. Nombra al creador de la especificación: Jorge Cortés G.
- Versión. Numero de la versión de la especificación: 1.0.
- Tipo_Const(). Declara las variables consideradas como estáticas (invariantes) a lo largo de la ejecución del sistema: NProductorConsumidor, NProductor, NConsumidor, NBúfer. Con estas variables se indica que componentes integran, de manera global, al sistema distribuido a especificar.
- Valor. Asigna valores a las variables declaradas con Tipo_Const(): NProductorConsumidor = 1, NProductor = 1, NConsumidor = 1, NBúfer = 1. Con estos valores se indica el número de componentes que integran, de manera global, al sistema distribuido a especificar.
- Instancias_Clases:Inicio.... Inicia la declaración de los componentes globales que integran al sistema distribuido a especificar: Productor-Consumidor, Productor, Consumidor, Búfer.
- Usa_IDC(). Define el identificador asociado a cada componente global declarado: Productor-Consumidor, Productor, Consumidor, Búfer.
- Instancias_Clases:Fin.... Termina la declaración de los componentes globales que integran al sistema distribuido a especificar.
- Ambiente_Ejec:Inicio.... Inicia la definición del ambiente de ejecución del sistema distribuido a especificar.
- Sistema_Oper. Definición del sistema operativo: Linux.
- Lenguaje_Prog. Definición del lenguaje de programación: C.
- Tipo_Comunicación. Definición del tipo de comunicación: Sockets.

- Ambiente_Ejec:Fin. . . . Termina la definición del ambiente de ejecución del sistema distribuido a especificar.

7.2.3. Especificación del trabajo LeGESD Productor

El trabajo LeGESD nombrado Productor es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.3. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Productor, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Productor, se utiliza la Tabla 7.4 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.4 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

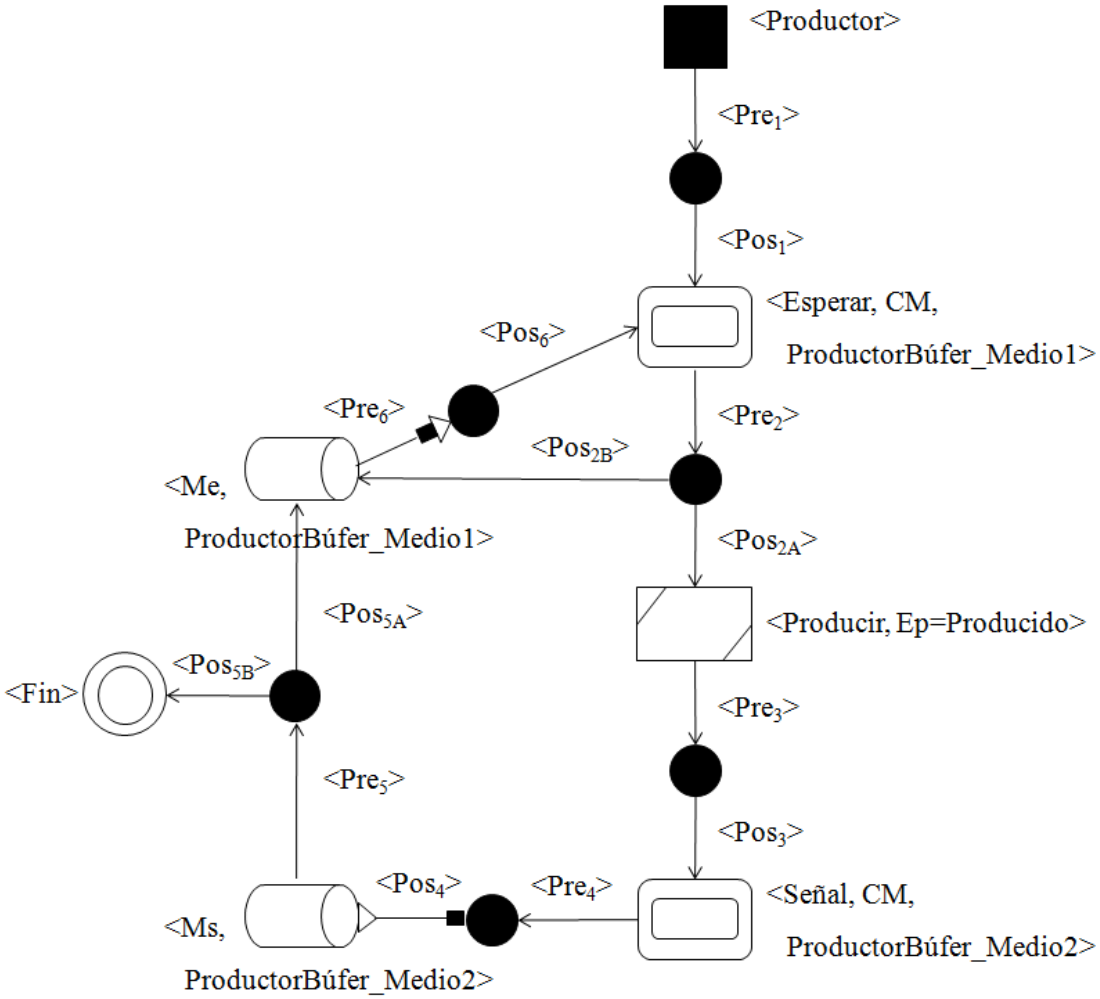


Figura 7.3: Diagrama de comportamiento del trabajo LeGESD Productor.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, Ep==null && Ns==N && v==0>
pos_1	Post-condición	<2, Ep=null, Ns=N, v=0>
pre_2	Precondición	<1, Ns!=0 || Ns==0>
pos_{2A}	Post-condición	<2, Ns!=0>
pos_{2B}	Post-condición	<2, Ns=0>
pre_3	Precondición	<1, Ep!=null>
pos_3	Post-condición	<2, Ep=Producido>
pre_4	Precondición	<1, msjProd1==null && msjProd2==null>
pos_4	Post-condición	<2, msjProd1.vDato=Ep, msjProd2.vDato=-1>
pre_5	Precondición	<1, v!=0 || v==0>
pos_{5A}	Post-condición	<2, v=0>
pos_{5B}	Post-condición	<2, v=1>
pre_6	Precondición	<1, Ns!=0>
pos_6	Post-condición	<2, Ns=msjBuf.vDato>
Me	Mensaje de entrada	<127.0.0.1, 127.0.0.1, Ns=msjBuf.vDato [msjBuf.tDato=int, msjBuf.tmDato=1]>
Ms	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjProd1.vDato=Ep [msjProd1.tDato=void *, msjProd1.tmDato=1], msjProd2.vDato=-1 [msjProd2.tDato=int, msjProd2.tmDato=1]>

Tabla 7.4: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Productor.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Productor son los siguientes:

- Estado Inicio nombrado Productor.
- Estado Comunicación nombrado Esperar.
- Estado Punto de acceso con la referencia ProductorBúfer_Medio1.
- Estado Interno nombrado Producir.
- Estado Comunicación nombrado Señal.
- Estado Punto de acceso con la referencia ProductorBúfer_Medio2.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Productor son del tipo enlaces simples etiquetados y enlaces de mensaje etiquetado-entrada y etiquetado-salida.

Cabe resaltar que los estados Esperar y Señal están enlazados con los estados Punto de acceso ProductorBúfer_Medio1 y ProductorBúfer_Medio2 respectivamente, lo que implica especificar el medio LeGESD que utilizarán estos estados para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en el capítulo.

En la especificación de cada uno de los estados que conforman el diagrama de comportamiento del trabajo LeGESD Productor se tiene lo siguiente:

- Estado Inicio: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Productor, además cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.

- Estado Comunicación Esperar: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Esperar, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establece la referencia del estado Punto de acceso (ProductorBúfer_Medio1) asociado a Esperar.

- Estado Punto de acceso ProductorBúfer_Medio1: tiene asociado el mensaje a recibir de su correspondiente Punto de acceso, este mensaje es <127.0.0.1, 127.0.0.1, Ns=msjBuf.vDato [msjBuf.tDato = int, msjBuf.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ProductorBúfer_Medio1.

- Estado Interno Producir: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Producir, se establecen declaraciones de acciones a realizar en el estado, esto es Ep = Producido.

- Estado Comunicación Señal: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Señal, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establece la referencia del estado Punto de acceso (ProductorBúfer_Medio2) asociado a Señal.

- Estado Punto de acceso ProductorBúfer_Medio2: tiene asociado los mensajes a enviar hacia su correspondiente Punto de acceso, estos mensaje son <127.0.0.1, 127.0.0.1, msjProd1.vDato = Ep [msjProd1.tDato = void *, msjProd1.tmDato = 1], msjProd2.vDato = -1 [msjProd2.tDato = int, msjProd2.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ProductorBúfer_Medio2.

- Estado Transición: no se establecen declaraciones de control.

- Estado Fin: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin.

En la especificación de cada uno de los enlaces simples etiquetados y los enlaces de mensaje etiquetado-entrada y etiquetado-salida, que conforman el diagrama de comportamiento del trabajo LeGESD Productor, se tiene lo siguiente:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Productor y el estado Esperar tienen como etiquetas la precondición <1, Ep==null && Ns==N && v==0> y la post-condición <2, Ep=null, Ns=N, v=0>. *Ep*, *Ns*, *v* y *N* son las variables y la constante que se declararon en el trabajo Produc durante la transición entre los estados Inicio y Productor. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Esperar y el estado Producir tienen como etiquetas las precondiciones <1, Ns!=0 || Ns==0> y la post-condición <2, Ns!=0>. Se cumple con las reglas sintácticas 3, 4, 5, 6 y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Esperar y el estado ProductorBúfer_Medio1 tienen como etiquetas la precondición <1, Ns!=0 || Ns==0> y la post-condición <2, Ns==0>. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8 y 9 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Producir y el estado Señal tienen como etiquetas la precondición <1, Ep!=null> y la post-condición <2, Ep=Producido>. Se cumple con las reglas sintácticas 3, 4, 5, 6 y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Señal y el estado ProductorBúfer_Medio2 tienen como etiquetas la precondición <1, msjProd1==null, msjProd2==null> y la post-condición <2, msjProd1.vDato = Ep, msjProd2.vDato = -1>. Los estados Señal y ProductorBúfer_Medio2 están enlazados a través de un enlace de mensaje etiquetado-salida, indicando la interacción con el medio de comunicación al enviar mensajes hacia el estado ProductorBúfer_Medio2 del trabajo LeGESD Búf. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9 y 14 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ProductorBúfer_Medio2 y el estado ProductorBúfer_Medio1 tienen como etiquetas la precondición <1, v!=0 || v==0> y la post-condición <2, v=0>. Se cumple con las reglas sintácticas 3, 4, 5, y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ProductorBúfer_Medio1 y el estado Esperar tienen como etiquetas la precondición <1, Ns!=0> y la post-condición <2, Ns=msjBuf.vDato>. La variable de mensaje msjBuf fue declarada en el trabajo Buf durante la transición entre los estados Búf y Búfer. Los estados ProductorBúfer_Medio1 y Esperar están enlazados a través de un enlace de mensaje etiquetado-entrada, indicando la interacción con el medio de comunicación al recibir mensajes del estado ProductorBúfer_Medio1 del trabajo LeGESD Búf. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9 y 14 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ProductorBúfer_Medio2 y el estado Fin tienen como etiquetas la precondición <1, v!=0 || v==0> y la post-condición <2, v=1>. Se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

Una vez explicados cada uno de los estados y enlaces que componen al diagrama de comportamiento del trabajo LeGESD Productor, a continuación de describe el comportamiento del trabajo:

1. Se inicia en el estado Productor, para cambiar de estado de Productor hacia Esperar se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de las variables Ep, Ns y v, las cuales han sido declaradas e inicializadas previamente en el trabajo LeGESD Produc. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores.

2. Del estado Esperar es posible cambiar de estado a Producir o a ProductorBúfer_Medio1, la elección entre estas dos posibles transiciones depende del valor de la variable Ns, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (Ns!=0 || Ns==0). Al cumplir la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

3. El primer enlace simple etiquetado tiene como post-condición Ns!=0, es decir, existen espacios en el búfer para producir un elemento como resultado de la transición, y se realiza el cambio de estado hacia Producir.

4. El segundo enlace simple etiquetado tiene como post-condición Ns==0, es decir, no existen espacios libres en el búfer para almacenar un elemento producido como resultado de la transición, y se realiza el cambio de estado hacia ProductorBúfer_Medio1.

5. Para cambiar de estado de Producir hacia Señal se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable Ep (Ep!=null). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición asigna el elemento Producido a la variable Ep, es decir, un elemento ha sido producido y se colocará en el búfer.

6. Para cambiar de estado de Señal hacia ProductorBúfer_Medio2 se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la comprobación de los valores de las variables de mensaje msjProd1 y msjProd2. Verificada la precondición, se realiza la transición y del estado Transición sale un enlace de mensaje etiquetado-salida con la post-condición msjProd1.vDato = Ep, msjProd2.vDato = -1, es decir se construye el mensaje Ms a enviar. Es importante hacer notar que el estado ProductorBúfer_Medio2 involucra una comunicación distribuida. En este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Búfer para enviar los mensajes que contendrán el elemento producido y el número de espacios a reducir en el búfer, es decir los valores msjProd1.vDato = Ep y msjProd2.vDato = -1 respectivamente.

7. Del estado ProductorBúfer_Medio2 es posible cambiar de estado a Fin o a ProductorBúfer_Medio1. La elección entre estas dos posibles transiciones depende del valor de la variable v, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra

al estado Transición (v!=0 || v==0). Al cumplir la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

8. El primer enlace simple etiquetado tiene como post-condición v=0, es decir, se comprueba si el valor de v es cero como resultado de la transición. Finalmente se alcanza el estado ProductorBúfer_Medio1.

9. Para cambiar de estado de ProductorBúfer_Medio1 hacia Esperar, se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relaciona con la comprobación del valor de la variable Ns (Ns!=0). Como post-condición se asigna valor a la variable anterior (Ns=msjBuf.vDato), cuyo valor es recibido remotamente del trabajo LeGESD Búfer en el mensaje Me. Es importante hacer notar que el estado ProductorBúfer_Medio1 involucra también una comunicación distribuida. En este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Búfer para recibir el mensaje que contendrá los datos relacionados con la variable Ns, es decir, el número de espacios que quedan en el búfer (variable Ns).

10. El segundo enlace simple etiquetado tiene como post-condición v=1, es decir, se comprueba que el valor de v es uno como resultado de la transición, alcanzándose el estado Fin.

De esta forma queda especificado el trabajo LeGESD Productor. A continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Productor. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.4 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\mathrm{T(Productor)}^{pre_1}\overset{pos_1}{\rightarrow}\mathrm{T(Esperar)} \tag{7.2.7}$$

$$\mathrm{T(Esperar)}^{pre_2}\overset{pos_{2A}}{\rightarrow}\mathrm{T(Producir)} + \mathrm{T(Esperar)}^{pre_2}\overset{pos_{2B}}{\rightarrow}\mathrm{T(ProductorBúfer\_Medio1)} \tag{7.2.8}$$

$$\text{T(Producir)}^{pre_3}\rightarrow^{pos_3}\text{T(Señal)} \tag{7.2.9}$$

$$\text{T(Señal)}\Delta^{pre_4,pos_4,Ms}\text{(T(ProductorBúfer\_Medio2))} \tag{7.2.10}$$

$$\begin{aligned}\text{T(ProductorBúfer\_Medio2)}^{pre_5}\rightarrow^{pos_{5A}}\text{T(ProductorBúfer\_Medio1)} + \\ \text{T(ProductorBúfer\_Medio2)}^{pre_5}\rightarrow^{pos_{5B}}\text{T(Fin)}\end{aligned} \tag{7.2.11}$$

$$[\text{T(ProductorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}\text{(T(Esperar))} \tag{7.2.12}$$

7.2.4. Especificación del trabajo LeGESD Búfer

El trabajo LeGESD nombrado Búfer es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.4. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Búfer, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

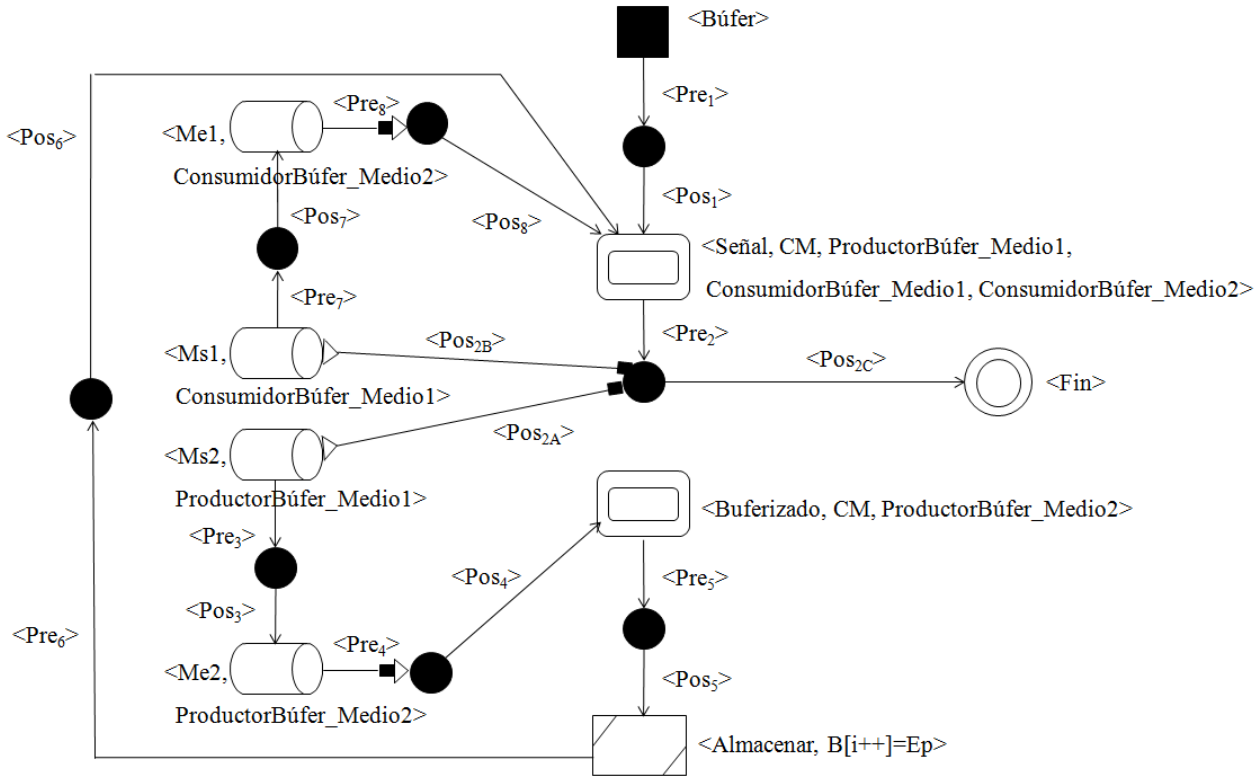


Figura 7.4: Diagrama de comportamiento del trabajo LeGESD Búfer.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Búfer, se utiliza la Tabla 7.5 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.5 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, Ep==null && Ne==0 && Ns==N && S==0 && msjBuf==null && msjBuf1==null && msjBuf2==null && t==1 && v==0 >
pos_1	Post-condición	<2, Ep=null, Ne=0, Ns=N, S=0, msjBuf=null, msjBuf1=null, msjBuf2=null, t=1, v=0>
pre_2	Precondición	<1, 0<=(Ns=Ns+S)<=N || 0<=(Ne=N-Ns)<=N || msjBuf==null || msjBuf1==null || msjBuf2==null || v==0>
pos_{2A}	Post-condición	<2, Ns>0, msjBuf=Ns, t=1, v=0>
pos_{2B}	Post-condición	<2, Ne>0, msjBuf1=B[i-1], msjBuf2=Ne, t=0, v=0>
pos_{2C}	Post-condición	<2, v=1>
pre_3	Precondición	<1, Ns!=0>
pos_3	Post-condición	<2, Ns>0>
pre_4	Precondición	<1, msjProd1!=null && msjProd2!=null>
pos_4	Post-condición	<2, Ep=msjProd1.vDato, S=msjProd2.vDato>
pre_5	Precondición	<1, Ep!=null && Ns!=0>
pos_5	Post-condición	<2, Ep!=null, Ns>0>
pre_6	Precondición	<1, Ep!=null && Ns!=0>
pos_6	Post-condición	<2, Ns>0, t=t-1>
pre_7	Precondición	<1, Ne!=0>
pos_7	Post-condición	<2, Ne>0>
pre_8	Precondición	<1, msjCons.vDato!=null>
pos_8	Post-condición	<2, S=msjCons.vDato>
$Me1$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, S=msjCons.vDato [msjCons.tDato = int, msjCons.tmDato = 1]>
$Me2$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, Ep = msjProd1.vDato [msjProd1.tDato = void *, msjProd1.tmDato = 1], S = msjProd2.vDato [msjProd2.tDato = int, msjProd2.tmDato = 1]>
$Ms1$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjBuf1.vDato = B[i-1] [msjBuf1.tDato = void *, msjBuf1.tmDato = 1], msjBuf2.vDato = Ne [msjBuf2.tDato = int, msjBuf2.tmDato = 1]>
$Ms2$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjBuf.vDato=Ns [msjBuf.tDato = int, msjBuf.tmDato = 1]>

Tabla 7.5: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Búfer.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Búfer son los siguientes:

- Estado Inicio nombrado Búfer.
- Estado Comunicación nombrado Señal.
- Estado Punto de acceso con la referencia ProductorBúfer_Medio1.
- Estado Punto de acceso con la referencia ConsumidorBúfer_Medio1.
- Estado Comunicación nombrado Buferizado.
- Estado Punto de acceso con la referencia ProductorBúfer_Medio2.
- Estado Punto de acceso con la referencia ConsumidorBúfer_Medio2.
- Estado Interno nombrado Almacenar.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Búfer son del tipo enlaces simples etiquetados y enlaces de mensaje etiquetados-entrada y etiquetados-salida.

Cabe resaltar que los estados Señal y Buferizado están enlazados con los estados Punto de acceso ProductorBúfer_Medio1, ConsumidorBúfer_Medio1 y ConsumidorBúfer_Medio2, y ProductorBúfer_Medio2 respectivamente, lo que implica especificar el medio LeGESD que utilizarán estos estados para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en este capítulo.

En la especificación de cada uno de los estados que conforman el diagrama de comportamiento del trabajo LeGESD Búfer se tiene lo siguiente:

- Estado Inicio: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Búfer, además cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Comunicación Señal: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Señal, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establecen las referencias de los estados Puntos de acceso (ProductorBúfer_Medio1, ConsumidorBúfer_Medio1 y ConsumidorBúfer_Medio2) asociados a Señal.
- Estado Punto de acceso ProductorBúfer_Medio1: tiene asociado el mensaje a enviar hacia su correspondiente Punto de acceso, este mensaje es <127.0.0.1, 127.0.0.1, msjBuf.vDato=Ns [msjBuf.tDato = int, msjBuf.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ProductorBúfer_Medio1.

- Estado Punto de acceso ProductorBúfer_Medio2: tiene asociado los mensajes a recibir de su correspondiente Punto de acceso, estos mensaje son <127.0.0.1, 127.0.0.1, Ep = msjProd1.vDato [msjProd1.tDato = void *, msjProd1.tmDato = 1], S = msjProd2.vDato [msjProd2.tDato = int, msjProd2.tm Dato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ProductorBúfer_Medio2.

- Estado Comunicación Buferizado: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Buferizado, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establecen las referencias de los estados Puntos de acceso (ProductorBúfer_Medio2) asociado a Buferizado.

- Estado Interno Almacenar: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Almacenar, se establecen declaraciones de acciones a realizar en el estado, esto es B[i++]=Ep.

- Estado Punto de acceso ConsumidorBúfer_Medio1: tiene asociado los mensajes a enviar hacia su correspondiente Punto de acceso, estos mensaje son <127.0.0.1, 127.0.0.1, msjBuf1.vDato = B[i-1] [msjBuf1.tDato = void *, msjBuf1.tmDato = 1], msjBuf2.vDato = Ne [msjBuf2.tDato = int, msjBuf2.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ConsumidorBúfer_Medio1.

- Estado Punto de acceso ConsumidorBúfer_Medio2: tiene asociado el mensaje a recibir de su correspondiente Punto de acceso, este mensaje es <127.0.0.1, 127.0.0.1, S=msjCons.vDato [msjCons.tDato = int, msjCons.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ConsumidorBúfer_Medio2.

- Estado Transición: no se establecen declaraciones de control.

- Estado Fin: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin.

En la especificación de cada uno de los enlaces simples etiquetados y los enlaces de mensaje etiquetados-entrada y etiquetados-salida, que conforman el diagrama de comportamiento del trabajo LeGESD Búfer, se tiene lo siguiente:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Búfer y el estado Señal tienen como etiquetas la precondición <1, Ep==null && Ne==0 && Ns==N && S==0 && msjBuf==null && msjBuf1==null && msjBuf2==null && t==1 && v==0 > y la post-condición <2, Ep=null, Ne=0, Ns=N, S=0, msjBuf=null, msjBuf1=null, msjBuf2=null, t=1, v=0>. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Ep, Ne, Ns, S, $msjBuf$, $msjBuf1$, $msjBuf2$, t, v y N son las variables y la constante que se declararon en el trabajo Búf durante la transición entre los estados Búf y Búfer. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Señal y el estado ProductorBúfer_Medio1 tienen como etiquetas la precondición <1, 0<=(Ns=Ns+S)<=N || 0<=(Ne=N-Ns)<=N || msjBuf==null || msjBuf1==null || msjBuf2==null || v==0> y la post-condición <2, Ns>0, msjBuf=Ns, t=1, v=0>. Los estados Señal y ProductorBúfer_Medio1 están enlazados a través de un enlace de mensaje etiquetado-salida, indicando la interacción con el medio de comunicación al enviar mensajes hacia el estado ProductorBúfer_Medio1 del trabajo LeGESD Productor. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ProductorBúfer_Medio1 y el estado ProductorBúfer_Medio2 tienen como etiquetas la precondición <1, Ns!=0> y la post-condición <2, Ns>0>. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ProductorBúfer_Medio2 y el estado Buferizado tienen como etiquetas la precondición <1, msjProd1!=null && msjProd2!=null> y la post-condición <2, Ep=msjProd1.vDato, S=msjProd2.vDato>. Los estados ProductorBúfer_Medio2 y Buferizado están enlazados a través de un enlace de mensaje etiquetado-entrada, indicando la interacción con el medio de comunicación al recibir mensajes del estado ProductorBúfer_Medio2 del trabajo LeGESD Productor. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Buferizado y el estado Almacenar tienen como etiquetas la precondición <1, Ep!=null && Ns!=0> y la post-condición <2, Ep!=null, Ns>0>. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Almacenar y el estado Señal tienen como etiquetas la precondición <1, Ep!=null && Ns!=0> y la post-condición <2, Ns>0, t=t-1>. Se cumple con las reglas sintácticas 3, 4, 5, 6 y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Señal y el estado ConsumidorBúfer_Medio1 tienen como etiquetas la precondición <1, 0<=(Ns=Ns+S)<=N || 0<=(Ne=N-Ns)<=N || msjBuf==null || msjBuf1==null || msjBuf2==null || v==0> y la post-condición <2, Ne>0, msjBuf1=B[i-1], msjBuf2=Ne, t=0, v=0>. El estado ConsumidorBúfer_Medio1 está enlazado al estado Señal a través de un enlace de mensaje etiquetado-salida, indicando la interacción con el medio de comunicación al enviar mensajes hacia el estado ConsumidorBúfer_Medio1 del trabajo LeGESD Consumidor. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ConsumidorBúfer_ Medio1 y el estado ConsumidorBúfer_Medio2 tienen como etiquetas la precondición <1, Ne!=0> y la post-condición <2, Ne>0>. Se cumple con las reglas sintácticas 3, 4, 5, y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ConsumidorBúfer_ Medio2 y el estado Señal tienen como etiquetas la precondición <1, msjCons.vDato!=null> y la post-condición <2, S=msjCons.vDato>. Los estados ConsumidorBúfer_Me-

dio2 y Señal están enlazados a través de un enlace de mensaje etiquetado-entrada, indicando la interacción con el medio de comunicación al recibir mensajes del estado ConsumidorBúfer_Medio2 del trabajo LeGESD Consumidor. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Señal y el estado Fin tienen como etiquetas la precondición <1, 0<=(Ns=Ns+S)<=N || 0<=(Ne=N-Ns)<=N || msjBuf==null || msjBuf1==null || msjBuf2==null || v==0> y la post-condición <2, V=1>. Se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

Una vez explicados cada uno de los estados y enlaces que componen al diagrama de comportamiento del trabajo LeGESD Búfer, a continuación de describe el comportamiento del trabajo:

1. Se inicia en el estado Búfer, para cambiar de estado de Búfer hacia Señal se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relacionan con la verificación de los valores de las variables Ep, Ne, Ns, S, $msjBuf$, $msjBuf1$, $msjBuf2$, t y v, las cuales han sido declaradas e inicializadas previamente en el trabajo LeGESD Búf. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores.

2. Del estado Señal es posible cambiar de estado a ProductorBúfer_Medio1 o a ConsumidorBúfer_Medio1 o a Fin, la elección entre estas tres posibles transiciones depende del valor de las variables Ns, Ne, $msjBuf$, $msjBuf1$, $msjBuf2$ y v, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (0<=(Ns=Ns+S)<=N || 0<=(Ne=N-Ns)<=N || msjBuf==null || msjBuf1==null || msjBuf2 ==null || v==0). Al cumplir la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces de mensaje etiquetados-salida y un enlace simple etiquetado con las post-condiciones generadas.

3. El primer enlace de mensaje etiquetado-salida tiene como post-condición Ns>0, msjBuf=Ns, t=1, v=0, es decir, existen espacios libres en el búfer para almacenar un elemento producido como resultado de la transición, y se realiza el cambio de estado hacia ProductorBúfer_Medio1. Es importante hacer notar que el estado ProductorBúfer_Medio1 involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Productor para enviar el mensaje contenido en la variable $msjBuf$ que tendrá los datos relacionados con la variable Ns, es decir el número de espacios que quedan en el búfer (variable Ns).

4. El segundo enlace de mensaje etiquetado-salida tiene como post-condición Ne>0, msjBuf1= B[i-1], msjBuf2=Ne, t=0, v=0, es decir existen elementos en el búfer para consumir un elemento producido, y se realiza el cambio de estado hacia ConsumidorBúfer_Medio1. Es importante hacer notar que el estado ConsumidorBúfer_Medio1 involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Consumidor para enviar los mensajes contenidos en las variables $msjBuf1$ y $msjBuf2$ que tendrán los datos relacionados con la variable $B[i-1]$ y Ne, es decir, el elemento a consumir (variable $B[i-1]$).

5. El enlace simple etiquetado tiene como post-condición v=1, es decir, se comprueba que el valor de *v* es uno como resultado de la transición, alcanzándose el estado Fin.

6. Para cambiar de estado de ProductorBúfer_Medio1 hacia ProductorBúfer_Medio2 se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *Ns* (Ns!=0), una vez verificado esto se realiza la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

7. Para cambiar de estado de ProductorBúfer_Medio2 hacia Buferizado se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relaciona con la comprobación de los valores de las variables *msjProd*1 y *msjProd*2 (msjProd1!=null y msjProd2!=null). Verificada la precondición, se realiza la transición y del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables Ep y S a partir de las variables *msjProd*1 y *msjProd*2. Es importante hacer notar que también el estado ProductorBúfer_Medio2 involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Productor para recibir los mensajes que contendrán el elemento producido y el número de espacios a reducir en el búfer, es decir, los valores de Ep y S.

8. Para cambiar de estado de Buferizado hacia Almacenar se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la comprobación de la asignación de valores de las variables Ep y Ns (Ep!=null && Ns!=0). Verificada la precondición, se realiza la transición y del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores.

9. Para cambiar de estado de Almacenar hacia Señal se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la comprobación de los valores de las variables Ep y Ns (Ep!=null && Ns!=0). Verificada la precondición, se realiza la transición y del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables Ns y t.

10. Para cambiar de estado de ConsumidorBúfer_Medio1 hacia ConsumidorBúfer_Medio2 se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *Ne* (Ne!=0), una vez verificado esto se realiza la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con el valor asignado a la variable anterior.

11. Para cambiar de estado de ConsumidorBúfer_Medio2 hacia Señal se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relaciona con la comprobación del valor de la variable *msjCons.vDato*

(msjCons.vDato!=null). Verificada la precondición, se realiza la transición y del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con el valor asignado a la variable *S*. Es importante hacer notar que el estado ConsumidorBúfer_Medio2 involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Consumidor para recibir el mensaje contenido en la variable $msjCons.vDato$ que tendrá el número de elementos consumidos.

De esta forma queda especificado el trabajo LeGESD Búfer. A continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Búfer. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la abla 7.5 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Búfer)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Señal)} \tag{7.2.13}$$

$$\begin{aligned}\text{T(Señal)}\Delta^{pre_2,pos_{2A},Ms2}\text{(T(ProductorBúfer\_Medio1))} + \\ \text{T(Señal)}\Delta^{pre_2,pos_{2B},Ms1}\text{(T(ConsumidorBúfer\_Medio1))} + \\ \text{T(Señal)}^{pre_2}\!\!\rightarrow^{pos_{2C}}\text{T(Fin)}\end{aligned} \tag{7.2.14}$$

$$\text{T(ProductorBúfer\_Medio1)}^{pre_3}\!\!\rightarrow^{pos_3}\text{T(ProductorBúfer\_Medio2)} \tag{7.2.15}$$

$$[\text{T(ProductorBúfer\_Medio2)/-}]_{Me2}\Delta^{pre_4,pos_4}\text{(T(Buferizado))} \tag{7.2.16}$$

$$\text{T(Buferizado)}^{pre_5}\!\!\rightarrow^{pos_5}\text{T(Almacenar)} \tag{7.2.17}$$

$$\text{T(Almacenar)} \overset{pre_6}{\rightarrow} \overset{pos_6}{} \text{T(Señal)} \tag{7.2.18}$$

$$\text{T(ConsumidorBúfer\_Medio1)} \overset{pre_7}{\rightarrow} \overset{pos_7}{} \text{T(ConsumidorBúfer\_Medio2)} \tag{7.2.19}$$

$$[\text{T(ConsumidorBúfer\_Medio2)/-}]_{Me1} \Delta^{pre_8, pos_8}(\text{T(Señal)}) \tag{7.2.20}$$

7.2.5. Especificación del trabajo LeGESD Consumidor

El trabajo LeGESD nombrado Consumidor es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.5. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Consumidor, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Consumidor, se utiliza la Tabla 7.6 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.6 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

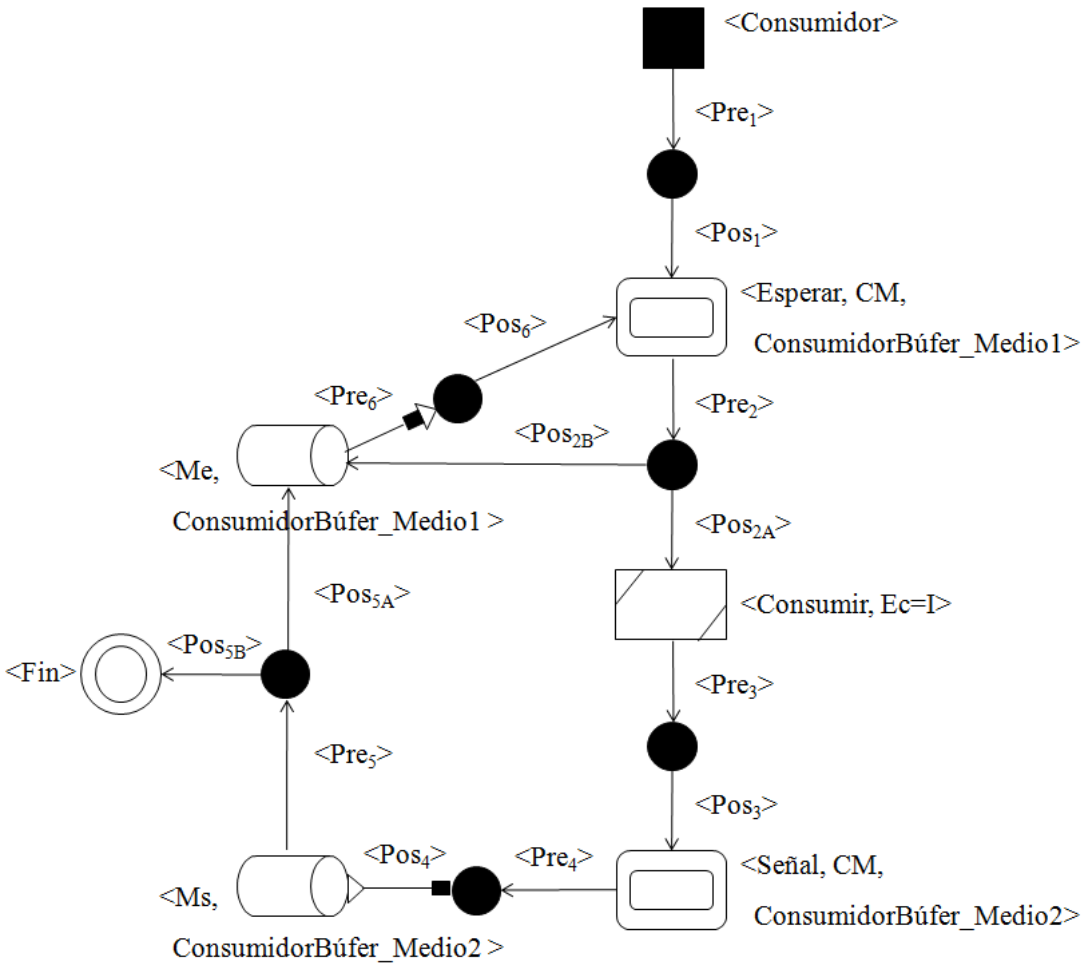


Figura 7.5: Diagrama de comportamiento del trabajo LeGESD Consumidor.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, Ec==null && Ne==0 && I==null && v==0>
pos_1	Post-condición	<2, Ec=null, Ne=0, I=null, v=0>
pre_2	Precondición	<1, Ne!=0 || Ne==0>
pos_{2A}	Post-condición	<2, Ne!=0>
pos_{2B}	Post-condición	<2, Ne=0>
pre_3	Precondición	<1, Ec!=null>
pos_3	Post-condición	<2, Ec=I>
pre_4	Precondición	<1, msjCons==null>
pos_4	Post-condición	<2, msjCons.vDato = 1>
pre_5	Precondición	<1, v!=0 || v==0>
pos_{5A}	Post-condición	<2, v=0>
pos_{5B}	Post-condición	<2, v=1>
pre_6	Precondición	<1, I!=null && Ne!=0>
pos_6	Post-condición	<2, I=msjBuf1.vDato, Ne=msjBuf2.vDato>
Me	Mensaje de entrada	<127.0.0.1, 127.0.0.1, I=msjBuf1.vDato [msjBuf1.tDato=void *, msjBuf1.tmDato=1], Ne=msjBuf2.vDato [msjBuf2.tDato=int, msjBuf2.tmDato=1]>
Ms	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjCons.vDato=1 [msjCons.tDato=int, msjCons.tmDato=1]>

Tabla 7.6: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Consumidor.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Consumidor son los siguientes:

- Estado Inicio nombrado Consumidor.
- Estado Comunicación nombrado Esperar.
- Estado Punto de acceso con la referencia ConsumidorBúfer_Medio1.
- Estado Interno nombrado Consumir.
- Estado Comunicación nombrado Señal.
- Estado Punto de acceso con la referencia ConsumidorBúfer_Medio2.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Consumidor son del tipo enlaces simples etiquetados y enlaces de mensaje etiquetado-entrada y etiquetado-salida. Un enlace de mensaje se entiende bajo la Definición 5.14.

Cabe resaltar que los estados Esperar y Señal están enlazados con los estados Punto de acceso ConsumidorBúfer_Medio1 y ConsumidorBúfer_Medio2 respectivamente, lo que implica especificar el medio LeGESD que utilizarán estos estados para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en el capítulo.

En la especificación de cada uno de los estados que conforman el diagrama de comportamiento del trabajo LeGESD Consumidor se tiene lo siguiente:

- Estado Inicio: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Consumidor, además cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.

- Estado Comunicación Esperar: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Esperar, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establece la referencia del estado Punto de acceso (ConsumidorBúfer_Medio1) asociado a Esperar.

- Estado Punto de acceso ConsumidorBúfer_Medio1: tiene asociado los mensajes a recibir de su correspondiente Punto de acceso, estos mensajes son <127.0.0.1, 127.0.0.1, I=msjBuf1. vDato [msjBuf1.tDato = void *, msjBuf1.tmDato = 1], Ne=msjBuf2.vDato [msjBuf2.tDato = int, msjBuf2.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ConsumidorBúfer_Medio1.

- Estado Interno Consumir: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Consumir, se establecen declaraciones de acciones a realizar en el estado, esto es Ec = I.

- Estado Comunicación Señal: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Señal, se establecen las declaraciones de acciones de comunicación con la constante LeGESD CM para definir un tipo de mecanismo de comunicación por mensajes, también se establece la referencia del estado Punto de acceso (ConsumidorBúfer_Medio2) asociado a Señal.

- Estado Punto de acceso ConsumidorBúfer_Medio2: tiene asociado el mensaje a enviar hacia su correspondiente Punto de acceso, este mensaje es <127.0.0.1, 127.0.0.1, msjCons.vDato = 1 [msjCons.tDato = int, msjCons.tmDato = 1]>, también se asocia la referencia al Punto de acceso que complementa la conexión distribuida, esta referencia es ConsumidorBúfer_Medio2.

- Estados Transición: no se establecen declaraciones de control.

- Estado Fin: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin.

En la especificación de cada uno de los enlaces simples etiquetados y los enlaces de mensaje etiquetado-entrada y etiquetado-salida, que conforman el diagrama de comportamiento del trabajo LeGESD Consumidor, se tiene lo siguiente:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Consumidor y el estado Esperar tienen como etiquetas la precondición <1, Ec==null && Ne==0 && I==null && v==0> y la post-condición <2, Ec=null, Ne=0, I=null, v=0>. *Ec*, *Ne*, *I* y *v* son las variables que se declararon en el trabajo Consum durante la transición entre los estados Inicio y Consumidor. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Esperar y el estado Consumir tienen como etiquetas las precondiciones <1, Ne!=0 || Ne==0> y la post-condición <2, Ne!=0>. Se cumple con las reglas sintácticas 3, 4, 5, 6 y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Esperar y el estado ConsumidorBúfer_Medio1 tienen como etiquetas la precondición <1, Ne!=0 || Ne==0> y la post-condición <2, Ne=0>. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7 y 13 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Consumir y el estado Señal tienen como etiquetas la precondición <1, Ec!=null> y la post-condición <2, Ec=I>. Se cumple con las reglas sintácticas 3, 4, 5, 6 y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Señal y el estado ConsumidorBúfer_Medio2 tienen como etiquetas la precondición <1, msjCons==null> y la post-condición <2, msjCons.vDato = 1>. Los estados Señal y ConsumidorBúfer_Medio2 están enlazados a través de un enlace de mensaje etiquetado-salida, indicando la interacción con el medio de comunicación al enviar mensajes hacia el estado ConsumidorBúfer_Medio2 del trabajo LeGESD Búf. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9 y 14 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ConsumidorBúfer_Medio2 y el estado ConsumidorBúfer_Medio1 tienen como etiquetas la precondición <1, v!=0 || v==0> y la post-condición <2, v=0>. Se cumple con las reglas sintácticas 3, 4, 5, y 7 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ConsumidorBúfer_Medio1 y el estado Esperar tienen como etiquetas la precondición <1, I!=null && Ne!=0> y la post-condición <2, I=msjBuf1.vDato, Ne=msjBuf2.vDato>. Las variables de mensaje msjBuf1 y msjBuf2 fueron declaradas en el trabajo Búf durante la transición entre los estados Búf y Búfer. Los estados ConsumidorBúfer_Medio1 y Transición están enlazados a través de un enlace de mensaje etiquetado-entrada, indicando la interacción con el medio de comunicación al recibir mensajes del estado ConsumidorBúfer_Medio1 del trabajo LeGESD Búf. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9 y 14 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado ConsumidorBúfer_Medio2 y el estado Fin tienen como etiquetas la precondición <1, v!=0 || v==0> y la post-condición <2, v=1>. Se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

Una vez explicados cada uno de los estados y enlaces que componen al diagrama de comportamiento del trabajo LeGESD Consumidor, a continuación de describe el comportamiento del trabajo:

1. Se inicia en el estado Consumidor, para cambiar de estado de Consumidor hacia Esperar se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación de los valores de las variables *Ec*, *Ne*, *I* y *v*, las cuales han sido declaradas e inicializadas previamente en el trabajo LeGESD Consum. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada, esta post-condición se relaciona con los valores asignados a las variables anteriores.

2. Del estado Esperar es posible cambiar de estado a ConsumidorBúfer_Medio1 o a Consumir, la elección entre estas dos posibles transiciones depende del valor de la variable *Ne*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (Ne!=0 || Ne==0). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

3. El primer enlace simple etiquetado tiene como post-condición Ne!=0, es decir, existen elementos a consumir del búfer como resultado de la transición, y se realiza el cambio de estado hacia Consumir.

4. El segundo enlace simple etiquetado tiene como post-condición Ne==0, es decir, no existen elementos a consumir del búfer como resultado de la transición, y se realiza el cambio de estado hacia ConsumidorBúfer_Medio1.

5. Para cambiar de estado de Consumir hacia Señal se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *Ec* (Ec!=null). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con la asignación del valor de la variable *I* a la variable *Ec*, es decir, el elemento que se obtuvo del búfer ha sido consumido.

6. Para cambiar de estado de Señal hacia ConsumidorBúfer_Medio2 se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la comprobación del valor de la variable de mensaje *msjCons*. Verificada la precondición, se realiza la transición y del estado Transición sale un enlace de mensaje etiquetado-salida con la post-condición msjCons.vDato = 1, construyéndose el mensaje *Ms* a enviar. Es importante hacer notar que el estado ConsumidorBúfer_Medio2 involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Búfer para enviar el mensaje que contendrá el número de elementos consumidos, es decir, el valor msjCons.vDato = 1 indica al búfer que fue consumido un elemento.

7. Del estado ConsumidorBúfer_Medio2 es posible cambiar de estado a Fin o a ConsumidorBúfer_Medio1, la elección entre estas dos posibles transiciones depende del valor de

la variable *v*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (v!=0 || v==0). Al llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

8. El primer enlace simple etiquetado tiene como post-condición v=0, es decir, se comprueba si el valor de *v* es cero como resultado de la transición. Finalmente se alcanza el estado ConsumidorBúfer_Medio1.

9. Para cambiar de estado de ConsumidorBúfer_Medio1 hacia Esperar se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relacionan con la comprobación de los valores de las variables *I* y *Ne*. Como post-condición se asignan valores a las variables anteriores (I=msjBuf1.vDato, Ne=msjBuf2.vDato), cuyos valores son recibidos remotamente del trabajo LeGESD Búfer en el mensaje *Me*. Es importante hacer notar que el estado ConsumidorBúfer_Medio1 involucra también una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Búfer para recibir los mensajes que contendrán los datos relacionados con las variables *I* y *Ne*, es decir, el valor que se encuentra en el búfer y que es consumido (variable *I*), y el número de elementos que quedan en el búfer (variable *Ne*).

10. El segundo enlace simple etiquetado tiene como post-condición v=1, es decir, se comprueba que el valor de *v* es uno como resultado de la transición, alcanzándose el estado Fin.

De esta forma queda especificado el trabajo LeGESD Consumidor. A continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Consumidor. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.6 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Consumidor)} \overset{pre_1}{\rightarrow}{}^{pos_1} \text{T(Esperar)} \tag{7.2.21}$$

$$\text{T(Esperar)} \overset{pre_2}{\rightarrow}{}^{pos_{2A}} \text{T(Consumir)} + \text{T(Esperar)} \overset{pre_2}{\rightarrow}{}^{pos_{2B}} \text{T(ConsumidorBúfer\_Medio1)} \tag{7.2.22}$$

$$\text{T(Consumir)}^{pre_3}\rightarrow^{pos_3}\text{T(Señal)} \tag{7.2.23}$$

$$\text{T(Señal)}\Delta^{pre_4,pos_4,Ms}\text{(T(ConsumidorBúfer\_Medio2))} \tag{7.2.24}$$

$$\begin{aligned}\text{T(ConsumidorBúfer\_Medio2)}^{pre_5}\rightarrow^{pos_{5A}}\text{T(ConsumidorBúfer\_Medio1)} + \\ \text{T(ConsumidorBúfer\_Medio2)}^{pre_5}\rightarrow^{pos_{5B}}\text{T(Fin)}\end{aligned} \tag{7.2.25}$$

$$[\text{T(ConsumidorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}\text{(T(Esperar))} \tag{7.2.26}$$

7.2.6. Especificación del medio LeGESD

El medio LeGESD (Definición 5.6 del capítulo 5) que se utiliza en el ejemplo, es descrito a través de un diagrama de comunicación, un diagrama de comunicación se entiende bajo la Definición 5.8 dada en el capítulo 5. El diagrama de comunicación se compone de dos diagramas: el diagrama de comunicación del medio de inicialización (Figura 7.6) y el diagrama de comunicación del medio de ejecución (Figura 7.7). Un medio de inicialización se entiende bajo la Definición 5.9 dada en el capítulo 5, y un medio de ejecución se entiende bajo la Definición 5.10 dada en el mismo capítulo. En ambos diagramas se muestra el grafo dirigido que describe tanto la inicialización como la ejecución de la comunicación del medio LeGESD, los cuales contienen un conjunto de símbolos gráficos: estados y enlaces.

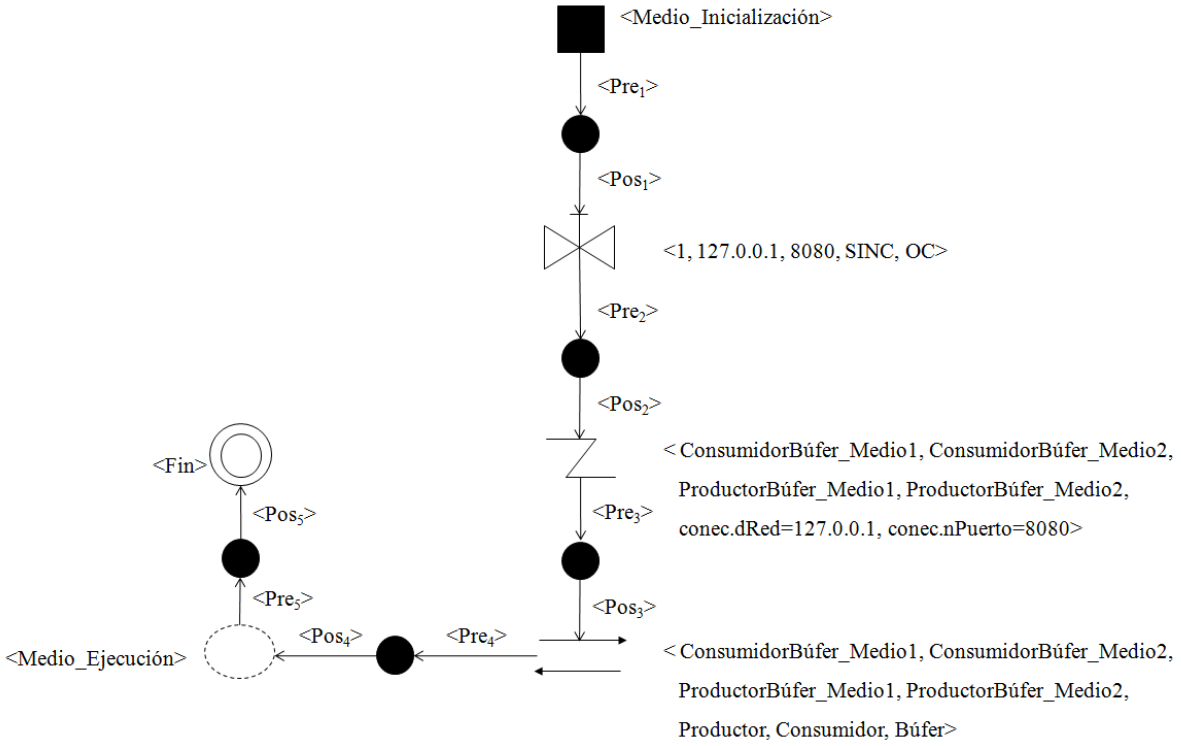


Figura 7.6: Diagrama de comunicación del medio de inicialización LeGESD para el Productor-Consumidor.

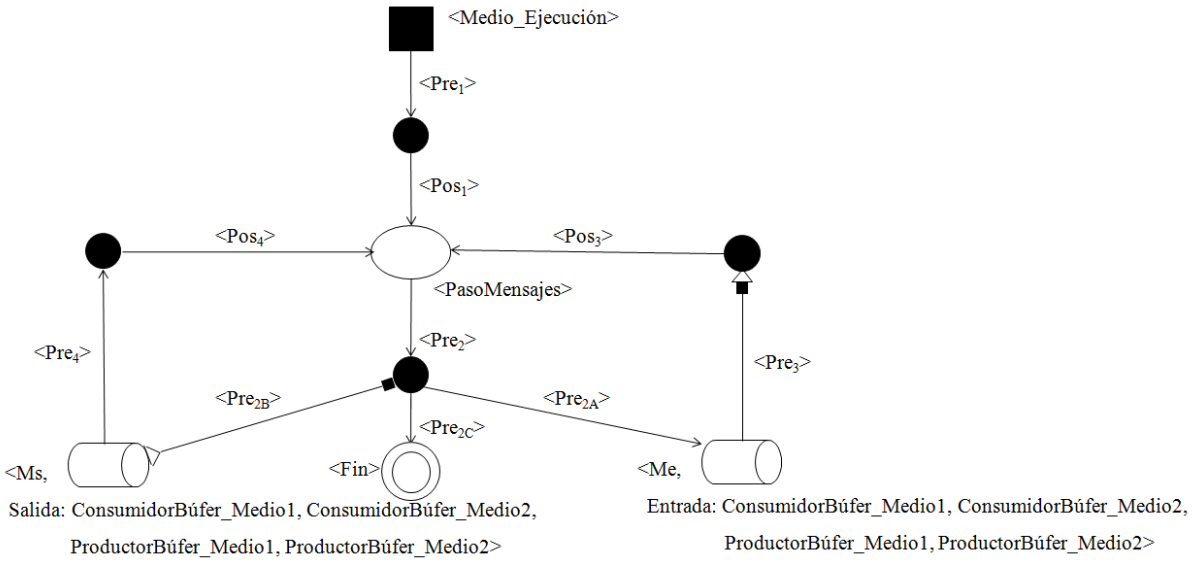


Figura 7.7: Diagrama de comunicación del medio de ejecución LeGESD para el Productor-Consumidor.

Para proporcionar sencillez en la interpretación del diagrama de comunicación del medio LeGESD, se utilizan las Tablas 7.7 y 7.8 para agrupar la precondición (que activa a la transición),

y la post-condición (generada después de la transición) relacionadas con cada transición de los diagramas del medio de inicialización y del medio de ejecución. Las Tablas 7.7 y 7.8 asocian a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, atd conec=null, int v=0>
pos_1	Post-condición	<2, conec=null>
pre_2	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_2	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080>
pre_3	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_3	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=0>
pre_4	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_4	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=1>
pre_5	Precondición	<1, v==1>
pos_5	Post-condición	<2, v=1 >

Tabla 7.7: Precondiciones y post-condiciones de las transiciones del diagrama de comunicación del medio de inicialización.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, msj men=null, int v=0>
pos_1	Post-condición	<2, men=null, v=0>
pre_2	Precondición	<1, men!=null && v==0>
pos_{2A}	Post-condición	<2, men=null, v=0>
pos_{2B}	Post-condición	<2, men!=null, v=0>
pos_{2C}	Post-condición	<2, v=1>
pre_3	Precondición	<1, men!=null>
pos_3	Post-condición	<2, men=menEntrada>
pre_4	Precondición	<1, men==null>
pos_4	Post-condición	<2, men=null>
Me	Mensaje de entrada	Consumidor: [Me], Productor: [Me], Búfer: [Me1, Me2]
Ms	Mensaje de salida	Consumidor: [Ms], Productor: [Ms], Búfer: [Ms1, Ms2]

Tabla 7.8: Precondiciones y post-condiciones de las transiciones del diagrama de comunicación del medio de ejecución.

Diagrama de comunicación del medio de inicialización. Los estados utilizados para especificar el diagrama de comunicación del medio de inicialización LeGESD son los siguientes:

- Estado Inicio nombrado Medio_Inicialización.
- Estado Apertura.

- Estado Atado.
- Estado Comienzo.
- Estado Compuesto nombrado Medio_Ejecución .
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comunicación del medio de inicialización LeGESD son del tipo enlaces simples etiquetados.

En la especificación de cada uno de los estados que conforman el diagrama de comunicación del medio de inicialización LeGESD se tiene lo siguiente:

- Estado Inicio: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Medio_Inicialización, además cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Apertura: tiene asociado como identificador el valor de 1, como dirección de red del destino (servidor) 127.0.0.1, como número de puerto el valor 8080, y como condiciones de apertura las constantes LeGESD SIN y OC (declaradas en el pseudocódigo LeGESD en el capítulo 5).
- Estado Atado: tiene asociada la lista de referencias de todos los estados Puntos de acceso que intervienen en la comunicación, es decir ConsumidorBufer_Medio1, ConsumidorBufer_Medio2, ProductorBufer_Medio1 y ProductorBufer_Medio2, así como la dirección de red del destino (conec.dRed= 127.0.0.1) y el número de puerto (conec.nPuerto=8080). La variable conec es del tipo atd (declarada con el pseudocódigo LeGESD en el capítulo 5).
- Estado Comienzo: tiene asociada la misma lista de referencias de los estados Puntos de acceso declarados en el estado Atado, así como la lista de trabajos LeGESD que intervendrán en la comunicación, es decir Productor, Consumidor y Búfer.
- Estado Compuesto: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Medio_Ejecución. Este estado debe ser especificado por el diagrama de comunicación del medio de ejecución LeGESD.
- Estado Transición: no se establecen declaraciones de control.
- Estado Fin: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin.

En la especificación de cada uno de los enlaces simples etiquetados que conforman el diagrama de comunicación del medio de inicialización LeGESD se tiene lo siguiente:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Medio_Inicialización y el estado Apertura tienen como etiquetas la precondición <1, atd conec=null, int v=0> y la post-condición <2, conec=null>. La variable conec se declara utilizando el pseudocódigo LeGESD que se presentó en el capítulo 5. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Adicionalmente, se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Apertura y el estado Atado tienen como etiqueta la precondición <1, conec.dRed!=null && conec.nPuerto !=0> y la post-condición <2, conec.dRed=127.0.0.1, conec.nPuerto=8080>. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Atado y el estado Comienzo tienen como etiquetas la precondición <1, conec.dRed!=null && conec. nPuerto!=0> y la post-condición <2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=0>. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Comienzo y el estado Medio_Ejecución tienen como etiquetas la precondición <1, conec.dRed!=null && conec.nPuerto!= 0> y la post-condición <2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=1>. Se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Medio_Ejecución y el estado Fin tienen como etiquetas la precondición <1, V==1> y la post-condición <2, v=1>. Se cumple con las reglas sintácticas 3, 4, 5 y 6 de LeGESD.

Una vez explicados cada uno de los estados y enlaces que componen al diagrama de comunicación del medio de inicialización LeGESD, a continuación de describe el comportamiento del diagrama:

1. Se comienza en el estado Medio, para cambiar de estado de Medio_Inicialización hacia Apertura se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables *conec* y *v*. La declaración e inicialización es verificada para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable *conec*.

2. Para cambiar de estado de Apertura hacia Atado se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la dirección de red del destino y del número de puerto (conec.dRed!=null && conec.nPuerto!=0). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a las variables anteriores.

3. Para cambiar de estado de Atado hacia Comienzo se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la dirección de red del destino y del número de puerto (conec.dRed!=null && conec.nPuerto!=0). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a las variables anteriores, además del valor asignado a la variable *v* (v=0).

4. Para cambiar de estado de Comienzo hacia Medio_Ejecución se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la dirección de red del destino y del número de puerto (conec.dRed!=null && conec.nPuerto!=0). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable *v* (v=1).

5. Para cambiar de estado de Medio_Ejecución hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *v*. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable *v* (v=1).

De esta forma queda especificado el medio de inicialización LeGESD. A continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica al diagrama de comunicación del medio de inicialización LeGESD del Productor-Consumidor. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición del diagrama de comunicación, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces simples etiquetados, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.7 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\mathrm{T(Medio)} \overset{pre_1}{\rightarrow} \overset{pos_1}{\mathrm{T(Apertura)}} \tag{7.2.27}$$

$$\mathrm{T(Apertura)} \overset{pre_2}{\rightarrow} \overset{pos_2}{\mathrm{T(Atado)}} \tag{7.2.28}$$

$$\text{T(Atado)} \overset{pre_3}{\rightarrow} \overset{pos_3}{} \text{T(Comienzo)} \tag{7.2.29}$$

$$\text{T(Comienzo)} \overset{pre_4}{\rightarrow} \overset{pos_4}{} \text{T(Medio\_Ejecución)} \tag{7.2.30}$$

$$\text{T(Medio\_Ejecución)} \overset{pre_5}{\rightarrow} \overset{pos_5}{} \text{T(Fin)} \tag{7.2.31}$$

Diagrama de comunicación del medio de ejecución. Este diagrama completa la especificación del medio LeGESD del ejemplo, y corresponde al estado Medio_Ejecución del diagrama de comunicación del medio de inicialización. Los estados utilizados para especificar el diagrama de comunicación del medio de ejecución LeGESD son los siguientes:

- Estado Inicio nombrado Medio_Ejecución.
- Estado Simple nombrado PasoMensajes.
- Estado Punto de acceso de entrada con las referencias: ConsumidorBufer_Medio1, ConsumidorBufer_Medio2, ProductorBufer_Medio1 y ProductorBufer_Medio2.
- Estado Punto de acceso de salida con las referencias: ConsumidorBufer_Medio1, ConsumidorBufer_Medio2, ProductorBufer_Medio1 y ProductorBufer_Medio2.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comunicación del medio de ejecución LeGESD son del tipo enlaces simples etiquetados y de mensaje etiquetado-entrada y etiquetado-salida.

En la especificación de cada uno de los estados que conforman el diagrama de comunicación del medio de ejecución LeGESD se tiene lo siguiente:

- Estado Inicio: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Medio_Ejecución, además cumple con los requisitos de especificación establecidos en el capítulo 5 para este estado.
- Estado Simple: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es PasoMensajes, no define declaraciones de acciones.
- Estado Punto de acceso de entrada: tiene asociado los mensajes a recibir de los estados Punto de acceso que intervienen en la comunicación, estos mensajes son los definidos en los trabajos Consumidor: [Me], Productor: [Me], Búfer: [Me1, Me2]. También se asocia la lista de referencias de los Puntos de acceso que intervienen en la comunicación, estos puntos son: ConsumidorBufer_Medio1, ConsumidorBufer_Medio2, ProductorBufer_Medio1 y ProductorBufer_Medio2.

- Estado Punto de acceso de salida: tiene asociado los mensajes a enviar hacia los estados Punto de acceso que intervienen en la comunicación, estos mensajes son los definidos en los trabajos Consumidor: [Ms], Productor: [Ms], Búfer: [Ms1, Ms2]. También se asocia la lista de referencias de los Puntos de acceso que intervienen en la comunicación, estos puntos son: ConsumidorBufer_Medio1, ConsumidorBufer_Medio2, ProductorBufer_Medio1 y ProductorBufer_Medio2.

- Estado Transición: no se establecen declaraciones de control.

- Estado Fin: tiene asociada una etiqueta que corresponde al nombre del estado, este nombre es Fin.

En la especificación de cada uno de los enlaces simples etiquetados y los enlaces de mensaje etiquetado-entrada y etiquetado-salida, que conforman el diagrama de comunicación del medio de ejecución LeGESD, se tiene lo siguiente:

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Medio_Ejecución y el estado PasoMensajes tienen como etiquetas la precondición <1, msj men=null, int v=0> y la post-condición <2, men=null, v=0>. Las variables *men* y *v* se declaran e inicializan utilizando el pseudocódigo LeGESD que se presentó en el capítulo 5. La declaración de la precondición y post-condición se realiza de acuerdo a la especificación que se dio en el capítulo 5. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado PasoMensajes y el estado Punto de acceso de entrada tienen como etiquetas la precondición <1, men!=null && v==0> y la post-condición <2, men=null, v=0>. Los estados PasoMensajes y Punto de acceso de entrada están enlazados a través de un enlace de mensaje etiquetado-entrada, indicando la interacción con los trabajos LeGESD del ejemplo al recibir mensajes de cualquiera de éstos. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD (cabe recordar que un estado Simple como PasoMensajes es idéntico a un estado Comunicación, por lo que las reglas sintácticas y las transformaciones de ADSD del estado Comunicación se aplican para el estado Simple).

- Los enlaces correspondientes a la transición (estado Transición) entre el estado PasoMensajes y el estado Punto de acceso de salida tienen como etiquetas la precondición <1, men!=null && v==0> y la post-condición <2, men!=null, v=0>. Los estados PasoMensajes y Punto de acceso de salida están enlazados a través de un enlace de mensaje etiquetado-salida, indicando la interacción con los trabajos LeGESD del ejemplo al enviarse mensajes a cualquiera de éstos. Se cumple con las reglas sintácticas 3, 4, 5, 6, 7, 8, 9, 14 y 15 de LeGESD

- Los enlaces correspondientes a la transición (estado Transición) entre el estado PasoMensajes y el estado Fin tienen como etiquetas la precondición <1, men!=null && v==0> y la post-condición <2, v=1>. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD dadas en el mismo capítulo.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Punto de acceso de entrada y el estado PasoMensajes tienen como etiquetas la precondición <1, men!=null> y la post-condición <2, men=menEntrada>. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD.

- Los enlaces correspondientes a la transición (estado Transición) entre el estado Punto de acceso de salida y el estado PasoMensajes tienen como etiquetas la precondición <1, men== null> y la post-condición <2, men=null>. Se cumple con las reglas sintácticas 3, 4, 5, y 6 de LeGESD.

Una vez explicados cada uno de los estados y enlaces que componen al diagrama de comunicación del medio de ejecución LeGESD, a continuación se describe el comportamiento del diagrama:

1. Se comienza en el estado Medio_Ejecución, para cambiar de estado de Medio_Ejecución hacia PasoMensajes se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables *men* y *v*. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a las variables anteriores.

2. Del estado PasoMensajes es posible cambiar de estado al Punto de acceso de entrada o al Punto de acceso de salida o a Fin, la elección entre estas tres posibles transiciones depende del valor de las variables *men* y *v*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (men!=null && v==0). Al cumplir la precondición y llevarse a cabo la transición, del estado Transición salen tres enlaces simples etiquetados con las post-condiciones generadas.

3. El primer enlace simple etiquetado tiene como post-condición men=null, v=0, es decir, no existe mensaje recibido en el medio como resultado de la transición, y se realiza el cambio de estado hacia el estado Punto de acceso de entrada. Es importante hacer notar que el estado Punto de acceso de entrada involucra una comunicación distribuida, este estado en el medio de comunicación conecta a cualquiera de los trabajos LeGESD del ejemplo (mediante las referencias de sus Puntos de acceso) para recibir los mensajes que contendrán los datos a trasmitir por el medio.

4. El segundo enlace simple etiquetado tiene como post-condición men!=null, v=0, es decir, existe algún mensaje a enviar por el medio como resultado de la transición, y se realiza el cambio de estado hacia el estado Punto de acceso de salida. Es importante hacer notar que el estado Punto de acceso de salida también involucra una comunicación distribuida, este estado en el medio de comunicación conecta a cualquiera de los trabajos LeGESD del ejemplo (mediante las referencias de sus Puntos de acceso) para enviarles los mensajes que contienen los datos trasmitidos por el medio.

5. El tercer enlace simple etiquetado tiene como post-condición v=1, es decir, se asigna el valor de *v* como resultado de la transición, en cuyo caso se alcanza el estado Fin.

6. Para cambiar de estado del estado Punto de acceso de entrada hacia PasoMensajes se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *men* con el mensaje que se haya recibido por parte de alguno de los trabajos LeGESD del ejemplo (men!=null). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con la asignación del valor recibido a la variable *men*.

7. Para cambiar de estado del estado Punto de acceso de salida hacia PasoMensajes se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *men* con un valor nulo (men==null), indicando que el mensaje ha sido transmitido a su destino. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con la asignación del valor nulo a la variable *men*.

De esta forma queda especificado el diagrama de comunicación del medio de ejecución LeGESD, y junto con la especificación del diagrama de comunicación del medio de inicialización realizada anteriormente, queda completamente especificado el medio LeGESD del ejemplo. A continuación se presenta la aplicación de ADSD en la especificación. Aplicar ADSD consiste en establecer la semántica operacional y realizar el análisis de comportamiento. La semántica operacional se entiende bajo la Definición 6.15 y el análisis de comportamiento se entiende bajo la Definición 6.17 dadas en el capítulo 6, y que consiste en obtener un conjunto de transformaciones de estados LeGESD a procesos ADSD (entendiéndose a estos procesos bajo la Definición 6.2), así como sus transiciones correspondientes.

Para conseguir lo anterior, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1 del capítulo 6. Este algoritmo se aplica al diagrama de comunicación del medio de ejecución LeGESD del Productor-Consumidor. Las ecuaciones de transición se entienden bajo la Definición 6.16.

Estas ecuaciones se construyen en cada estado Transición de los diagramas de comunicación, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos en la Definición 6.14 para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.8 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Medio\_Ejecución)} \overset{pre_1}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_1} \text{T(PasoMensajes)} \tag{7.2.32}$$

$$\begin{aligned} &\text{T(PasoMensajes)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_{2A}} \text{T(Entrada)} + \\ &\text{T(PasoMensajes)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_{2B}} \text{T(Salida)} + \\ &\text{T(PasoMensajes)} \overset{pre_2}{\rightarrow}\!\!\!\!\!\!\rightarrow^{pos_{2C}} \text{T(Fin)} \end{aligned} \tag{7.2.33}$$

$$[\text{T(Entrada)/-}]_{Me}\Delta^{pre_3,pos_3}(\text{T(PasoMensajes)}) \tag{7.2.34}$$

$$\text{T(Salida)}\Delta^{pre_4,pos_4,Ms}(\text{T(PasoMensajes)}) \tag{7.2.35}$$

Con esto concluye la especificación completa en el lenguaje LeGESD del sistema productor-consumidor. Para el ejemplo sólo se consideró la existencia de un productor y un consumidor, sin embargo se puede generalizar para N productores y M consumidores debido la existencia del trabajo LeGESD Búfer, el cual se encarga de controlar el búfer del sistema permitiendo la comunicación con múltiples productores y consumidores. En el siguiente apartado se presenta la especificación del sistema cena de criptógrafos.

7.3. Especificación del sistema cena de criptógrafos utilizando el lenguaje LeGESD

La especificación del sistema cena de criptógrafos se simplifica asumiendo dos criptógrafos y un tercero que fungirá como un maestro (la NSA), el cuál dará seguimiento del cumplimento del protocolo. La especificación del sistema cena de criptógrafos inicia con la identificación de las entidades lógicas que integran al sistema distribuido a especificar. Las entidades lógicas que integran al sistema son las siguientes:

- Dos criptógrafos.
- El maestro.

A continuación se presentan la Vista de sistema y la Vista de implementación para el sistema especificado con LeGESD.

7.3.1. Vista de sistema

La vista de sistema está integrada por los trabajos LeGESD siguientes para el ejemplo especificado:

- Cript_1.
- Cript_2.
- Maest.

Estos trabajos LeGESD se llevan a cabo de forma concurrente, comunicándose de forma distribuida para realizar sus actividades designadas. Los trabajos LeGESD listados previamente son descritos a través de sus respectivos diagramas de comportamiento que se muestran en la Figura 7.8 (lado derecho). En cada uno de estos diagramas se muestra el grafo dirigido que describe el

comportamiento del trabajo LeGESD correspondiente, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

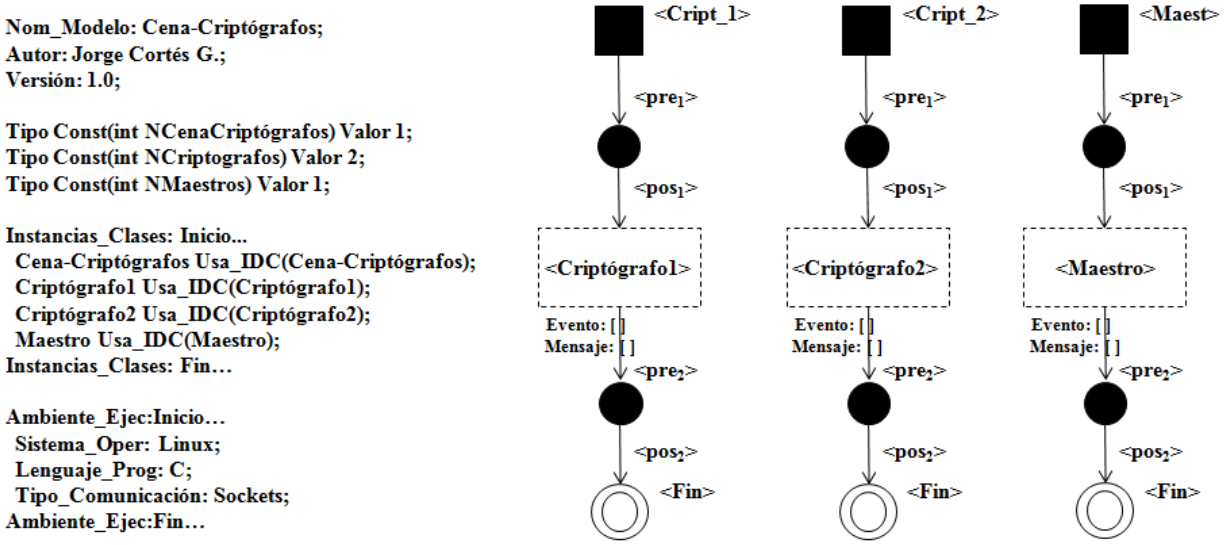


Figura 7.8: Vista de sistema y de implementación del sistema cena de criptógrafos utilizando el lenguaje LeGESD.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento de cada trabajo LeGESD, se utilizan las Tablas 7.9 - 7.11 para agrupar la precondición (que activa a la transición) y la post-condición (generada después de la transición) relacionadas con cada transición. Cada una de las tablas asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, int rVolado=-1, int vVolado=-1, int anuncio=-1, int pague=-1, int mensajes=0, int resultado[NCriptógrafos]=null, msj msjCript1=null, msj msjTxSal=null>
pos_1	Post-condición	<2, rVolado=-1, vVolado=-1, anuncio=-1, pague=-1, mensajes=0, resultado[NCriptógrafos]=null, msjCript1=null, msjTxSal=null>
pre_2	Precondición	<1, anuncio==-2>
pos_2	Post-condición	<2, anuncio=-2>

Tabla 7.9: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Cript_1.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, int rVolado=-1, int vVolado=-1, int anuncio=-1, int pague=-1, int mensajes=0, int resultado[NCriptógrafos]=null, msj msjCript2=null, msj msjTxSal=null>
pos_1	Post-condición	<2, rVolado=-1, vVolado=-1, anuncio=-1, pague=-1, mensajes=0, resultado[NCriptógrafos]=null, msjCript2=null, msjTxSal=null>
pre_2	Precondición	<1, anuncio==-2>
pos_2	Post-condición	<2, anuncio=-2>

Tabla 7.10: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Cript_2.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, int mensajes=0, int valor=0, int recibido=-1, char pagado_por=' '>
pos_1	Post-condición	<2, mensajes=0, valor=0, recibido=-1, pagado_por=' '>
pre_2	Precondición	<1, pagado_por=='F'>
pos_2	Post-condición	<2, pagado_por='F'>

Tabla 7.11: Precondiciones y post-condiciones de las transiciones del trabajo LeGESD Maest.

Los estados utilizados para especificar los diagramas de comportamiento de los trabajos LeGESD de la cena de criptógrafos son los siguientes:

- Estado Inicio nombrado Cript_1.
- Estado Inicio nombrado Cript_2.
- Estado Inicio nombrado Maest.
- Estado Compuesto nombrado Criptógrafo1.
- Estado Compuesto nombrado Criptógrafo2.
- Estado Compuesto nombrado Maestro.
- Estados Transición.
- Estados Fin nombrados Fin.

Los enlaces utilizados para especificar los diagramas de comportamiento de los trabajos LeGESD de la cena de criptógrafos son del tipo enlaces simples etiquetados. Los trabajos LeGESD de la cena de criptógrafos mostrados en la vista de sistema, representan el nivel más alto de la composición jerárquica de la especificación. El siguiente nivel de la composición jerárquica está dado por los estados Compuestos (Criptógrafo1, Criptógrafo2 y Maestro) contenidos en sus respectivos diagramas de comportamiento.

Una vez identificados los estados y enlaces que componen los diagramas de comportamiento del trabajo LeGESD de la cena de criptógrafos, a continuación de describe el comportamiento del trabajo:

1. Comenzando en el estado Inicio de los diagramas de comportamiento de Cript_1, Cript_2 y Maest, es posible cambiar de estado hacia el estado Criptógrafo1, Criptógrafo2 y Maestro respectivamente. Este cambio de estado se lleva a cabo de manera concurrente, por lo cual se tienen tres flujos de comportamiento.

   Primer flujo de comportamiento (trabajo LeGESD Cript_1).

2. Para cambiar del estado Cript_1 hacia Criptógrafo1 se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la declaración e inicialización de las variables $rVolado$, $vVolado$, $anuncio$, $pague$, $mensajes$, $resultado[NCriptógrafos]$, $msjCript1$, $msjTxSal$, las cuales deben haberse declarado e inicializado para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores. El significado de estas variables es el siguiente:
   rVolado:= Resultado del volado lanzado por el criptógrafo.
   vVolado:= Resultado del volado lanzado por el criptógrafo vecino.
   anuncio:= Anuncio de los resultado de los volados.
   pague:= Decisión de pagar o no la cuenta del criptógrafo.
   mensajes:= Contador de los criptógrafos que han enviado su mensaje.
   resultado[NCriptógrafos]:= Resultado anunciado por un criptógrafo.
   msjCript1, msjTxSal:= mensajes de comunicación.

3. Para cambiar del estado Criptógrafo1 hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *anuncio*, la cual debe ser igual a -2 para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

   Segundo flujo de comportamiento (trabajo LeGESD Cript_2).

2. Los pasos 2 y 3 son idénticos al del trabajo LeGESD Cript_1.

   Tercer flujo de comportamiento (trabajo LeGESD Maest).

2. Para cambiar del estado Maest hacia Maestro se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición: Esta precondición se relaciona con la declaración e inicialización de las variables $mensajes$, $valor$, $recibido$, $pagado\_por$, las cuales deben haberse declarado e inicializado para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las

variables anteriores. El significado de estas variables es el siguiente:
mensajes:= Contador de los criptógrafos que han enviado su mensaje.
valor:= Valor de la suma de los resultados enviados por los criptógrafos.
recibido:= Valor del mensaje enviado por un criptógrafo.
pagado_por:= Anuncia quien pagó la cuenta.

3. Para cambiar del estado Maestro hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición: Esta precondición se relaciona con la verificación del valor de la variable *pagado_por*, la cual debe ser igual a 'F ' para realizar la transición de estado. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

De esta forma queda especificada la vista de sistema del ejemplo utilizando el lenguaje LeGESD. A continuación se presenta la aplicación de ADSD en la especificación. Para esto, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1. Este algoritmo se aplica a cada diagrama de comportamiento del trabajo LeGESD de la Cena de criptógrafos. Estas ecuaciones se construyen en cada estado Transición de los diagramas de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces simples etiquetados, haciendo uso de los operadores establecidos para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en las Tablas 7.9 - 7.11 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Cript\_1)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Criptógrafo1)} \tag{7.3.1}$$

$$\text{T(Criptógrafo1)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Fin)} \tag{7.3.2}$$

$$\text{T(Cript\_2)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Criptógrafo2)} \tag{7.3.3}$$

$$\text{T(Criptógrafo2)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Fin)} \tag{7.3.4}$$

$$\text{T(Maest)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Maestro)} \tag{7.3.5}$$

$$\text{T(Maestro)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Fin)} \tag{7.3.6}$$

Con esto se concluye la descripción de la vista de sistema, sin embargo ésta no es suficiente para especificar completamente el ejemplo trabajado con LeGESD. Esta vista sólo es el nivel más alto en la composición jerárquica de la especificación. Para completar la especificación se deben desarrollar los diagramas de comportamiento de los trabajos LeGESD Criptógrafo1, Criptógrafo2 y Maestro que representan el siguiente nivel en la composición jerárquica de la especificación. Adicionalmente, se debe especificar el medio de comunicación distribuida a emplear en el ejemplo, mediante los diagramas de comportamiento del medio LeGESD.

Es obligatorio para estos trabajos desarrollar sus diagramas de comportamiento por tratarse de estados Compuestos. En secciones posteriores se presentarán estos trabajos y el medio LeGESD para completar la especificación del ejemplo.

7.3.2. Vista de implementación

La vista de implementación representa las restricciones en la implementación del sistema distribuido, la cual consiste básicamente en el ambiente de ejecución del sistema a especificar. La vista de implementación se compone de anotaciones textuales dentro de la vista de sistema, como se observa en la Figura 7.8 (parte izquierda). Estas anotaciones textuales consisten en palabras reservadas del pseudocódigo LeGESD (Tabla 5.1). Para el ejemplo se utilizan las palabras mostradas en la figura anterior.

7.3.3. Especificación del trabajo LeGESD Criptógrafo1

El trabajo LeGESD nombrado Criptógrafo1 es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.9. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Criptógrafo1, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

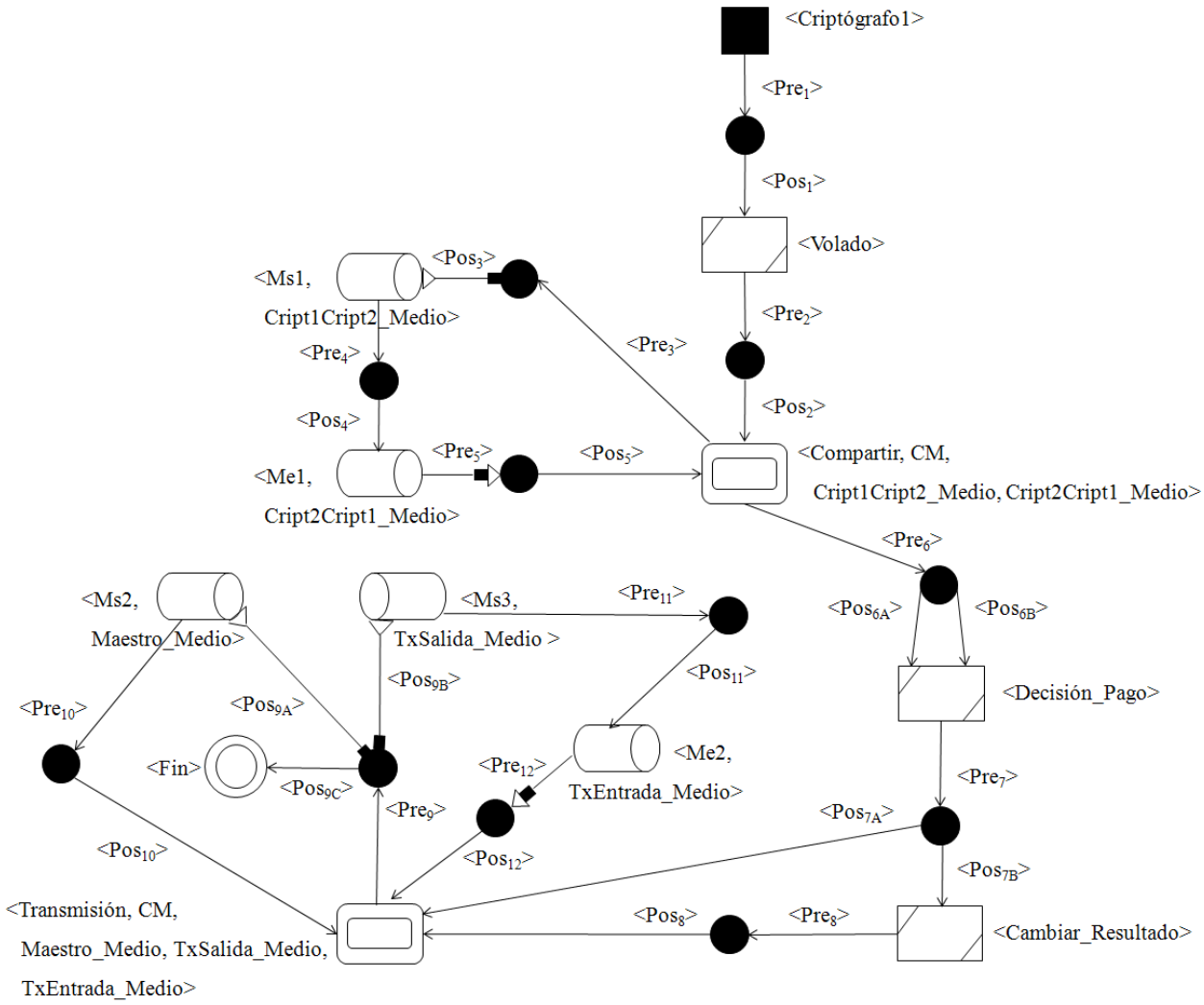


Figura 7.9: Diagrama de comportamiento del trabajo LeGESD Criptógrafo1.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, rVolado==-1 && int pague==-1>
pos_1	Post-condición	<2, rVolado=-1, pague=-1>
pre_2	Precondición	<1, rVolado==0 || rVolado==1>
pos_2	Post-condición	<2, rVolado!=-1>
pre_3	Precondición	<1, msjCript1==null>
pos_3	Post-condición	<2, msjCript1.vDato=rVolado>
pre_4	Precondición	<1, rVolado!=-1>
pos_4	Post-condición	<2, rVolado>= 0>
pre_5	Precondición	<1, msjCript2!=null>
pos_5	Post-condición	<2, vVolado=msjCript2.vDato>
pre_6	Precondición	<1, vVolado==rVolado || vVolado!=rVolado>
pos_{6A}	Post-condición	<2, anuncio=0>
pos_{6B}	Post-condición	<2, anuncio=1>
pre_7	Precondición	<1, pague==0 || pague==1>
pos_{7A}	Post-condición	<2, pague=0, msjCript1=null>
pos_{7B}	Post-condición	<2, pague=1>
pre_8	Precondición	<1, anuncio==0 || anuncio==1>
pos_8	Post-condición	<2, anuncio=!anuncio, msjCript1=null>
pre_9	Precondición	<1, msjCript1==null || (msjCript1!=null && mensajes<NCriptógrafos) || (anuncio>=0 && mensajes== NCriptógrafos)>
pos_{9A}	Post-condición	<2, msjCript1=null, msjCript1.vDato=anuncio>
pos_{9B}	Post-condición	<2, msjCript1!=null, mensajes<NCriptógrafos, msjTxSal.vDato=anuncio>
pos_{9C}	Post-condición	<2, anuncio=-2>
pre_{10}	Precondición	<1, anuncio!=-1>
pos_{10}	Post-condición	<2, anuncio>=0>
pre_{11}	Precondición	<1, msjCript1!=null && mensajes<NCriptógrafos>
pos_{11}	Post-condición	<2, anuncio>=0>
pre_{12}	Precondición	<1, msjTxEnt!=null>
pos_{12}	Post-condición	<2, resultado[mensajes]=msjTxEnt.vDato, mensajes=mensajes+1>
$Me1$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, vVolado= msjCript2.vDato [msjCript2.tDato=int, msjCript2.tmDato=1]>
$Me2$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, resultado[mensajes]= msjTxEnt.vDato [msjTxEnt.tDato=int, msjTxEnt.tmDato=1]>
$Ms1$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjCript1.vDato=rVolado [msjCript1.tDato=int, msjCript1.tmDato=1]>
$Ms2$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjCript1.vDato=anuncio [msjCript1.tDato=int, msjCript1.tmDato=1]>
$Ms3$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjTxSal.vDato=anuncio [msjTxSal.tDato=int, msjTxSal.tmDato=1]>

Tabla 7.12: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Criptógrafo1.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Criptógrafo1, se utiliza la Tabla 7.12 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.12 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Criptógrafo1 son los siguientes:

- Estado Inicio nombrado Criptógrafo1.
- Estado Interno nombrado Volado.
- Estado Comunicación nombrado Compartir.
- Estado Punto de acceso con la referencia Cript1Cript2_Medio.
- Estado Punto de acceso con la referencia Cript2Cript1_Medio.
- Estado Interno nombrado Decisión_Pago.
- Estado Interno nombrado Cambiar_Resultado.
- Estado Comunicación nombrado Transmisión.
- Estado Punto de acceso con la referencia Maestro_Medio.
- Estado Punto de acceso con la referencia TxSalida_Medio.
- Estado Punto de acceso con la referencia TxEntrada_Medio.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Criptógrafo1 son del tipo enlaces simples etiquetados y enlaces de mensaje etiquetado-entrada y etiquetado-salida. Cabe resaltar que el estado Compartir está enlazado con los estados Punto de acceso Cript1Cript2_Medio y Cript2Cript1_Medio, mientras que el estado Transmisión está enlazado con los estados Punto de acceso Maestro_Medio, TxSalida_Medio y TxEntrada_Medio, lo que implica especificar el medio LeGESD que utilizarán estos estados para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en el capítulo.

Una vez identificados cada uno de los estados y enlaces que componen al diagrama de comportamiento del trabajo LeGESD Criptógrafo1, a continuación de describe el comportamiento del trabajo:

1. Se inicia en el estado Criptógrafo1, para cambiar de estado de Criptógrafo1 hacia Volado se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación de los valores de las variables *rVolado* y *pague*, las cuales han sido declaradas e inicializadas previamente en el trabajo

LeGESD Cript_1. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores.

2. Para cambiar de estado de Volado hacia Compartir se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *rVolado* (rVolado==0 || rVolado==1). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

3. Para cambiar de estado de Compartir hacia Cript1Cript2_Medio se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable $msjCript1$ (msjCript1==null). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición msjCript1.vDato=rVolado, construyéndose el mensaje $Ms1$ a enviar. Es importante hacer notar que el estado Cript1Cript2_Medio involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Criptógrafo2, el cual es el vecino de Criptógrafo1, para enviar el mensaje $Ms1$ con el resultado del volado lanzado por Criptógrafo1, es decir el valor msjCript1.vDato=rVolado.

4. Para cambiar de estado de Cript1Cript2_Medio hacia Cript2Cript1_Medio se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *rVolado* (rVolado!=-1). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

5. Para cambiar de estado de Cript2Cript1_Medio hacia Compartir se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relacionan con la comprobación del valor de la variable $msjCript2$ (msjCript2!=null). Como post-condición se asigna valor a la variable *vVolado* (vVolado=msjCript2.vDato), cuyo valor es recibido remotamente del trabajo LeGESD Criptógrafo2. Es importante hacer notar que el estado Cript2Cript1_Medio involucra también una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Criptógrafo2 para recibir el mensaje $Me1$ que contendrá el dato relacionado con la variable *vVolado* (vVolado=msjCript2.vDato), es decir el valor resultante del volado lanzado por el criptógrafo vecino.

6. Del estado Compartir es posible cambiar a Decisión_Pago de dos posibles maneras, la elección entre estas dos posibles transiciones depende de los valores de las variables *rVolado* y *vVolado*, los cuales deben ser verificados como precondición en el enlace simple etiquetado que entra al estado Transición (vVolado==rVolado || vVolado!=rVolado). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

7. El primer enlace simple etiquetado tiene como post-condición anuncio=0, es decir, los volados lanzados por Criptógrafo1 y Criptógrafo2 son iguales, y se realiza el cambio de estado hacia Decisión_Pago.

8. El segundo enlace simple etiquetado tiene como post-condición anuncio=1, es decir, los volados lanzados por Criptógrafo1 y Criptógrafo2 son diferentes, y se realiza el cambio de estado hacia Decisión_Pago.

9. Del estado Decisión_Pago es posible cambiar de estado a Transmisión o a Cambiar_Resultado, la elección entre estas dos posibles transiciones depende del valor de la variable *pague*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (pague==0 || pague==1). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

10. El primer enlace simple etiquetado tiene como post-condición pague=0, msjCript1=null, es decir, Criptógrafo1 ha decidido no pagar la cuenta, y se realiza el cambio de estado hacia Transmisión.

11. El segundo enlace simple etiquetado tiene como post-condición pague=1, es decir, Criptógrafo1 ha decidido pagar la cuenta, y se realiza el cambio de estado hacia Cambiar_Resultado.

12. Para cambiar de estado de Cambiar_Resultado hacia Transmisión se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *anuncio* (anuncio==0 || anuncio==1). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anuncio=!anuncio, es decir, Criptógrafo1 cambia el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2.

13. Del estado Transmisión es posible cambiar de estado a Maestro_Medio o a TxSalida_Medio o a Fin, la elección entre estas dos posibles transiciones depende de los valores de las variables *msjCript*1, *mensajes* y *anuncio*, los cuales deben ser verificados como precondición en el enlace simple etiquetado que entra al estado Transición (msjCript1==null || (msjCript1!=null && mensajes<NCriptógrafos) || (anuncio>=0 && mensajes== NCriptógrafos)). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces de mensaje etiquetados-salida y un enlace simple etiquetado con las post-condiciones generadas.

14. El primer enlace de mensaje etiquetado-salida tiene como post-condición msjCript1=null, msjCript1.vDato=anuncio, es decir, Criptógrafo1 anuncia el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 al maestro, y se realiza el cambio de estado hacia Maestro_Medio. Es importante hacer notar que también el estado Maestro_Medio involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con el trabajo LeGESD Maestro para enviar el mensaje *Ms*2 que contendrá el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2, es decir, el valor msjCript1.vDato=anuncio.

15. El segundo enlace de mensaje etiquetado-salida tiene como post-condición msjCript1!=null, mensajes<NCriptógrafos, msjTxSal.vDato=anuncio, es decir, Criptógrafo1 realiza la difusión del resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 a los demás criptógrafos, y se realiza el cambio de estado hacia TxSalida_Medio. Es importante hacer notar que también el estado TxSalida_Medio involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con otros trabajos LeGESD del tipo CriptógrafoN para enviar por difusión el mensaje $Ms3$ que contendrá el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2, es decir, el valor msjTxSal.vDato=anuncio.

16. El enlace simple etiquetado tiene como post-condición anuncio=-2, es decir, se comprueba que el valor de *anuncio* es menos dos como resultado de la transición, alcanzándose el estado Fin.

17. Para cambiar de estado de Maestro_Medio hacia Transmisión se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *anuncio* (anuncio!=-1). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con el valor asignado a la variable anterior.

18. Para cambiar de estado de TxSalida_Medio hacia TxEntrada_Medio se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación de los valores de las variables $msjCript1$ y $mensajes$ (msjCript1!=null && mensajes<NCriptógrafos). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada anuncio>=0.

19. Para cambiar de estado de TxEntrada_Medio hacia Transmisión se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relaciona con la comprobación del valor de la variable $msjTxEnt$ (msjTxEnt!=null). Como post-condición se asignan valores a las variables $resultado$ y $mensajes$ (resultado[mensajes]=msjTxEnt.vDato, mensajes=mensajes+1), cuyos valores son recibidos remotamente de otros trabajos LeGESD del tipo CriptógrafoN. Es importante hacer notar que el estado TxEntrada_Medio involucra también una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con otros trabajos LeGESD del tipo CriptógrafoN para recibir por difusión el mensaje $Me2$ que contendrá el dato relacionado con la variable $resultado$ (resultado[mensajes]=msjTxEnt.vDato), es decir, el resultado de los volados lanzados por otros pares de criptógrafos.

De esta forma queda especificado el trabajo LeGESD Criptógrafo1. A continuación se presenta la aplicación de ADSD en la especificación. Para esto, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Criptógrafo1. Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación),

con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.12 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Criptógrafo1)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Volado)} \tag{7.3.7}$$

$$\text{T(Volado)}^{pre_2}\!\!\rightarrow^{pos_2}\text{T(Compartir)} \tag{7.3.8}$$

$$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript1Cript2\_Medio))} \tag{7.3.9}$$

$$\text{T(Cript1Cript2\_Medio)}^{pre_4}\!\!\rightarrow^{pos_4}\text{T(Cript2Cript1\_Medio)} \tag{7.3.10}$$

$$[\text{T(Cript2Cript1\_Medio)/-}]_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))} \tag{7.3.11}$$

$$\text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6A}}\text{T(Decisión\_Pago)} + \text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6B}}\text{T(Decisión\_Pago)} \tag{7.3.12}$$

$$\begin{aligned}\text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7A}}\text{T(Transmisión)} + \\ \text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7B}}\text{T(Cambiar\_Resultado)}\end{aligned} \tag{7.3.13}$$

$$\text{T(Cambiar\_Resultado)}^{pre_8}\!\!\rightarrow^{pos_8}\text{T(Transmisión)} \tag{7.3.14}$$

$$\begin{aligned}\text{T(Transmisión)}\Delta^{pre_9,pos_{9A},Ms2}\text{(T(Maestro\_Medio))} + \\ \text{T(Transmisión)}\Delta^{pre_9,pos_{9B},Ms3}\text{(T(TxSalida\_Medio))} + \\ \text{T(Transmisión)}^{pre_9}\!\!\rightarrow^{pos_{9C}}\text{T(Fin)}\end{aligned} \tag{7.3.15}$$

$$\text{T(Maestro\_Medio)}^{pre_{10}}\!\!\rightarrow^{pos_{10}}\text{T(Transmisión)} \tag{7.3.16}$$

$$\text{T(TxSalida\_Medio)}^{pre_{11}}\!\!\rightarrow^{pos_{11}}\text{T(TxEntrada\_Medio)} \tag{7.3.17}$$

$$[\text{T(TxEntrada\_Medio}]_{Me2}\Delta^{pre_{12},pos_{12}}\text{(T(Transmisión))} \tag{7.3.18}$$

7.3.4. Especificación del trabajo LeGESD Criptógrafo2

El trabajo LeGESD nombrado Criptógrafo2 es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.10. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Criptógrafo2, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

Como se puede observar del diagrama, el comportamiento de Criptógrafo2 es idéntico al de Criptógrafo1 descrito en el apartado anterior, teniéndose sólo como variantes: el nombre del estado Inicio (Criptógrafo2), la variable msjCript2, y el sentido de la recepción y envío de los estados Punto de acceso (Cript2Cript1_Medio y Cript1Cript2_Medio). Debido a lo expuesto, la explicación del comportamiento del trabajo Criptógrafo2 no se detallará en este apartado pues ya ha sido explicado en el apartado anterior.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Criptógrafo2, se utiliza la Tabla 7.13 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.13 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

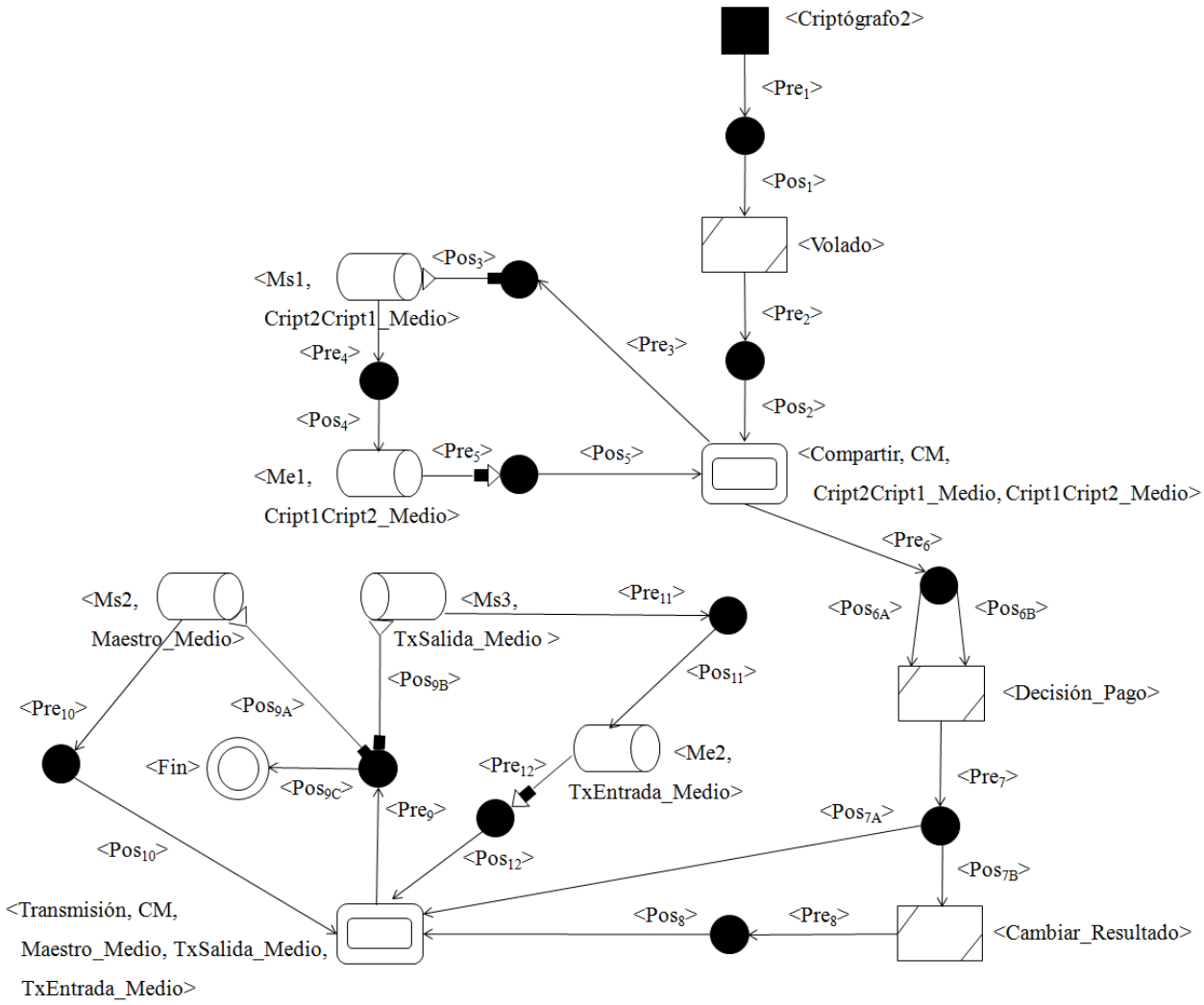


Figura 7.10: Diagrama de comportamiento del trabajo LeGESD Criptógrafo2.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, rVolado==-1 && int pague==-1>
pos_1	Post-condición	<2, rVolado=-1, pague=-1>
pre_2	Precondición	<1, rVolado==0 || rVolado==1>
pos_2	Post-condición	<2, rVolado!=-1>
pre_3	Precondición	<1, msjCript2==null>
pos_3	Post-condición	<2, msjCript2.vDato=rVolado>
pre_4	Precondición	<1, rVolado!=-1>
pos_4	Post-condición	<2, rVolado>= 0>
pre_5	Precondición	<1, msjCript1!=null>
pos_5	Post-condición	<2, vVolado=msjCript1.vDato>
pre_6	Precondición	<1, vVolado==rVolado || vVolado!=rVolado>
pos_{6A}	Post-condición	<2, anuncio=0>
pos_{6B}	Post-condición	<2, anuncio=1>
pre_7	Precondición	<1, pague==0 || pague==1>
pos_{7A}	Post-condición	<2, pague=0, msjCript2=null>
pos_{7B}	Post-condición	<2, pague=1>
pre_8	Precondición	<1, anuncio==0 || anuncio==1>
pos_8	Post-condición	<2, anuncio=!anuncio, msjCript2=null>
pre_9	Precondición	<1, msjCript2==null || (msjCript2!=null && mensajes<NCriptógrafos) || (anuncio>=0 && mensajes== NCriptógrafos)>
pos_{9A}	Post-condición	<2, msjCript2=null, msjCript2.vDato=anuncio>
pos_{9B}	Post-condición	<2, msjCript2!=null, mensajes<NCriptógrafos, msjTxSal.vDato=anuncio>
pos_{9C}	Post-condición	<2, anuncio=-2>
pre_{10}	Precondición	<1, anuncio!=-1>
pos_{10}	Post-condición	<2, anuncio>=0>
pre_{11}	Precondición	<1, msjCript2!=null && mensajes<NCriptógrafos>
pos_{11}	Post-condición	<2, anuncio>=0>
pre_{12}	Precondición	<1, msjTxEnt!=null>
pos_{12}	Post-condición	<2, resultado[mensajes]=msjTxEnt.vDato, mensajes=mensajes+1>
$Me1$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, vVolado= msjCript1.vDato [msjCript1.tDato=int, msjCript1.tmDato=1]>
$Me2$	Mensaje de entrada	<127.0.0.1, 127.0.0.1, resultado[mensajes]= msjTxEnt.vDato [msjTxEnt.tDato=int, msjTxEnt.tmDato=1]>
$Ms1$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjCript2.vDato=rVolado [msjCript2.tDato=int, msjCript2.tmDato=1]>
$Ms2$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjCript2.vDato=anuncio [msjCript2.tDato=int, msjCript2.tmDato=1]>
$Ms3$	Mensaje de salida	<127.0.0.1, 127.0.0.1, msjTxSal.vDato=anuncio [msjTxSal.tDato=int, msjTxSal.tmDato=1]>

Tabla 7.13: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Criptógrafo2.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Criptógrafo2 son los siguientes:

- Estado Inicio nombrado Criptógrafo2.
- Estado Interno nombrado Volado.
- Estado Comunicación nombrado Compartir.
- Estado Punto de acceso con la referencia Cript2Cript1_Medio.
- Estado Punto de acceso con la referencia Cript1Cript2_Medio.
- Estado Interno nombrado Decisión_Pago.
- Estado Interno nombrado Cambiar_Resultado.
- Estado Comunicación nombrado Transmisión.
- Estado Punto de acceso con la referencia Maestro_Medio.
- Estado Punto de acceso con la referencia TxSalida_Medio.
- Estado Punto de acceso con la referencia TxEntrada_Medio.
- Estados Transición.
- Estado Fin nombrado Fin.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Criptógrafo2 son del tipo enlaces simples etiquetados y enlaces de mensaje etiquetado-entrada y etiquetado-salida. Cabe resaltar que el estado Compartir está enlazado con los estados Punto de acceso Cript2Cript1_Medio y Cript1Cript2_Medio, mientras que el estado Transmisión está enlazado con los estados Punto de acceso Maestro_Medio, TxSalida_Medio y TxEntrada_Medio, lo que implica especificar el medio LeGESD que utilizarán estos estados para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en el capítulo.

De esta forma queda especificado el trabajo LeGESD Criptógrafo2. A continuación se presenta la aplicación de ADSD en la especificación. Para esto, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Criptógrafo2. Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.13 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Criptógrafo2)} \overset{pre_1}{\rightarrow} \overset{pos_1}{\text{T(Volado)}} \tag{7.3.19}$$

$$\text{T(Volado)}^{pre_2}\!\!\rightarrow^{pos_2}\text{T(Compartir)} \tag{7.3.20}$$

$$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript2Cript1\_Medio))} \tag{7.3.21}$$

$$\text{T(Cript2Cript1\_Medio)}^{pre_4}\!\!\rightarrow^{pos_4}\text{T(Cript1Cript2\_Medio)} \tag{7.3.22}$$

$$[\text{T(Cript1Cript2\_Medio)/-}]_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))} \tag{7.3.23}$$

$$\text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6A}}\text{T(Decisión\_Pago)} + \text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6B}}\text{T(Decisión\_Pago)} \tag{7.3.24}$$

$$\begin{aligned}\text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7A}}\text{T(Transmisión)} + \\ \text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7B}}\text{T(Cambiar\_Resultado)}\end{aligned} \tag{7.3.25}$$

$$\text{T(Cambiar\_Resultado)}^{pre_8}\!\!\rightarrow^{pos_8}\text{T(Transmisión)} \tag{7.3.26}$$

$$\begin{aligned}\text{T(Transmisión)}\Delta^{pre_9,pos_{9A},Ms2}\text{(T(Maestro\_Medio))} + \\ \text{T(Transmisión)}\Delta^{pre_9,pos_{9B},Ms3}\text{(T(TxSalida\_Medio))} + \\ \text{T(Transmisión)}^{pre_9}\!\!\rightarrow^{pos_{9C}}\text{T(Fin)}\end{aligned} \tag{7.3.27}$$

$$\text{T(Maestro\_Medio)}^{pre_{10}}\!\!\rightarrow^{pos_{10}}\text{T(Transmisión)} \tag{7.3.28}$$

$$\text{T(TxSalida\_Medio)}^{pre_{11}}\!\!\rightarrow^{pos_{11}}\text{T(TxEntrada\_Medio)} \tag{7.3.29}$$

$$[\text{T(TxEntrada\_Medio}]_{Me2}\Delta^{pre_{12},pos_{12}}\text{(T(Transmisión))} \tag{7.3.30}$$

7.3.5. Especificación del trabajo LeGESD Maestro

El trabajo LeGESD nombrado Maestro es descrito a través del diagrama de comportamiento que se muestra en la Figura 7.11. En este diagrama se muestra el grafo dirigido que describe el comportamiento del trabajo LeGESD Maestro, el cual contiene un conjunto de símbolos gráficos: estados y enlaces.

Para proporcionar sencillez en la interpretación del diagrama de comportamiento del trabajo LeGESD Maestro, se utiliza la Tabla 7.14 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición. La Tabla 7.14 asocia a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

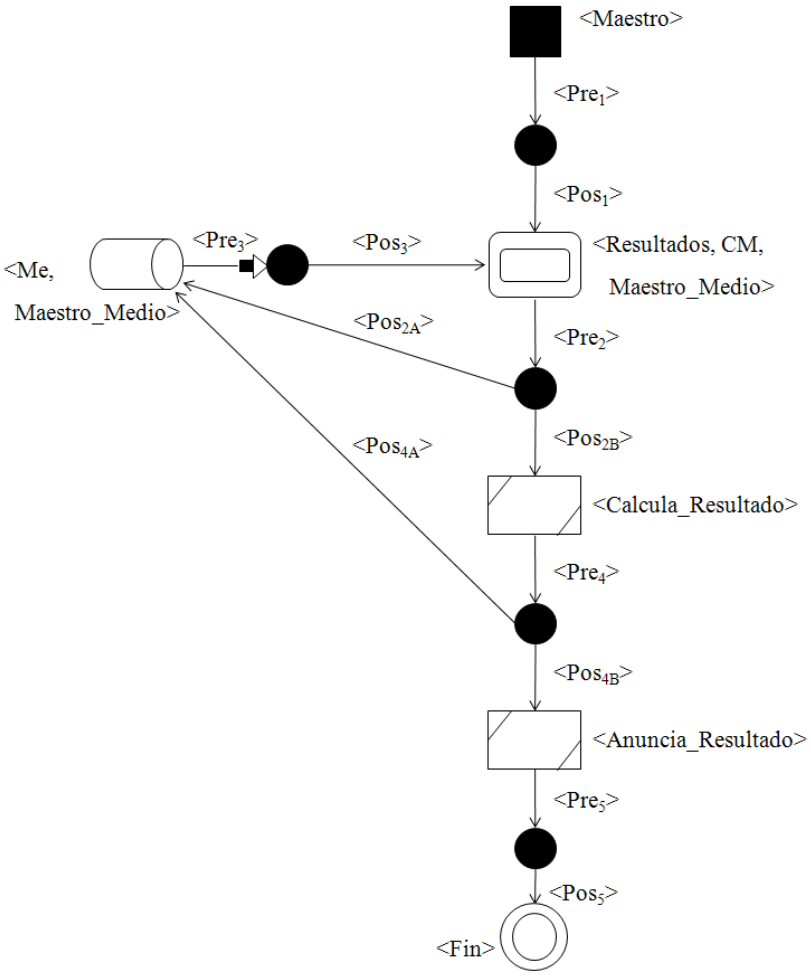


Figura 7.11: Diagrama de comportamiento del trabajo LeGESD Maestro.

Los estados utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Maestro son los siguientes:

- Estado Inicio nombrado Maestro.
- Estado Comunicación nombrado Resultados.

- Estado Punto de acceso con la referencia Maestro_Medio.
- Estado Interno nombrado Calcula_Resultado.
- Estado Interno nombrado Anuncia_Resultado.
- Estados Transición.
- Estado Fin nombrado Fin.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, mensajes==0 && valor==0 && recibido==-1>
pos_1	Post-condición	<2, mensajes=0, valor=0, recibido=-1>
pre_2	Precondición	<1, recibido==-1 || recibido!=-1>
pos_{2A}	Post-condición	<2, recibido=-1>
pos_{2B}	Post-condición	<2, recibido!=-1>
pre_3	Precondición	<1, msjCript(i)!=null>
pos_3	Post-condición	<2, recibido=msjCript(i).vDato, mensajes=mensajes+1>
pre_4	Precondición	<1, valor>=0 && mensajes<NCriptógrafos>
pos_{4A}	Post-condición	<2, mensajes<=NCriptógrafos, valor=recibido>
pos_{4B}	Post-condición	<2, mensajes=NCriptógrafos, pagado_por=' '>
pre_5	Precondición	<1, pagado_por=='C' || pagado_por=='N'>
pos_5	Post-condición	<2, pagado_por='F'>
Me	Mensaje de entrada	<127.0.0.1, 127.0.0.1, recibido= msjCript(i).vDato [msjCript(i).tDato=int, msjCript(i).tmDato=1]>

Tabla 7.14: Precondiciones y post-condiciones de las transiciones del diagrama de comportamiento del trabajo LeGESD Maestro.

Los enlaces utilizados para especificar el diagrama de comportamiento del trabajo LeGESD Maestro son del tipo enlaces simples etiquetados y un enlace de mensaje etiquetado-entrada. Cabe resaltar que el estado Resultados está enlazado con el estado Punto de acceso Maestro_Medio, lo que implica especificar el medio LeGESD que utilizará este estado para llevar a cabo la comunicación distribuida. Esta especificación del medio LeGESD se realiza más adelante en el capítulo.

Una vez identificados cada uno de los estados y enlaces que componen al diagrama de comportamiento del trabajo LeGESD Maestro, a continuación se describe el comportamiento del trabajo:

1. Se inicia en el estado Maestro, para cambiar de estado de Maestro hacia Resultados se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación de los valores de las variables *mensaje*, *valor* y *recibido*, las cuales han sido declaradas e inicializadas previamente en el trabajo LeGESD Maest. Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con los valores asignados a las variables anteriores.

2. Del estado Resultados es posible cambiar de estado a Maestro_Medio o a Calcula_Resultado, la elección entre estas dos posibles transiciones depende del valor de la variable *recibido*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (recibido==-1 || recibido!=-1). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

3. El primer enlace simple etiquetado tiene como post-condición recibido=-1, es decir, Maestro no ha recibido un mensaje con el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2.

4. El segundo enlace simple etiquetado tiene como post-condición recibido!=-1, es decir, Maestro ha recibido un mensaje con el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2.

5. Para cambiar de estado de Maestro_Medio hacia Resultados se debe cumplir la precondición contenida en el enlace de mensaje etiquetado-entrada que entra al estado Transición. Esta precondición se relaciona con la comprobación del valor de la variable $msjCript(i)$ (msjCript(i)!=null). Como post-condición se asigna valor a la variable *recibido* (recibido=msjCript(i).vDato), cuyo valor es recibido remotamente de alguno de los trabajos LeGESD Criptógrafo(i) (para este ejemplo *i* puede variar de 1 a 2). Es importante hacer notar que el estado Maestro_Medio involucra una comunicación distribuida, en este estado se accede al medio de comunicación conectándose con alguno de los trabajos LeGESD Criptógrafo(i) para recibir el mensaje *Me* que contendrá el dato relacionado con la variable *recibido* (recibido=msjCript(i).vDato), es decir, el valor resultante de los volado lanzados por el Criptógrafo1 y Criptógrafo2 para el ejemplo que se está trabajando.

6. Del estado Calcula_Resultado es posible cambiar de estado a Maestro_Medio o a Anuncia_Resultado, la elección entre estas dos posibles transiciones depende del valor de la variable *mensajes*, el cual debe ser verificado como precondición en el enlace simple etiquetado que entra al estado Transición (valor>=0 && mensajes<=NCriptógrafos). Al verificarse la precondición y llevarse a cabo la transición, del estado Transición salen dos enlaces simples etiquetados con las post-condiciones generadas.

7. El primer enlace simple etiquetado tiene como post-condición mensajes<=NCriptógrafos, valor= recibido, es decir, Maestro aún no ha recibido todos los mensajes con el resultado de los volados lanzados por los criptógrafos.

8. El segundo enlace simple etiquetado tiene como post-condición mensajes=NCriptógrafos, pagado_por =' ', es decir, Maestro ha recibido todos los mensajes con el resultado de los volados lanzados por los criptógrafos.

9. Para cambiar de estado de Anuncia_Resultado hacia Fin se debe cumplir la precondición contenida en el enlace simple etiquetado que entra al estado Transición. Esta precondición se relaciona con la verificación del valor de la variable *pagado_por* (pagado_por=='C' || pagado_por=='N'). Como resultado de la transición, del estado Transición sale un enlace simple etiquetado con la post-condición generada. Esta post-condición se relaciona con pagado_por='F '.

De esta forma queda especificado el trabajo LeGESD Maestro. A continuación se presenta la aplicación de ADSD en la especificación. Para esto, es necesario determinar las ecuaciones de transición generadas a través del Algoritmo 1. Este algoritmo se aplica al diagrama de comportamiento del trabajo LeGESD Maestro. Estas ecuaciones se construyen en cada estado Transición del diagrama de comportamiento, resultando una ecuación de transición por cada estado Transición. La ecuación consiste en relacionar cada proceso ADSD (término de la ecuación), con la precondición y post-condición contenidas en los enlaces, haciendo uso de los operadores establecidos para la especificación composicional de ADSD. Estas ecuaciones hacen uso de las etiquetas definidas en la Tabla 7.14 para las precondiciones y post-condiciones.

De esta manera, se obtienen las siguientes ecuaciones de transición después de aplicar el Algoritmo 1:

$$\text{T(Maestro)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Resultados)} \tag{7.3.31}$$

$$\text{T(Resultados)} \overset{pre_2}{\rightarrow} \overset{pos_{2A}}{} \text{T(Maestro\_Medio)} + \text{T(Resultados)} \overset{pre_2}{\rightarrow} \overset{pos_{2B}}{} \text{T(Calcula\_Resultado)} \tag{7.3.32}$$

$$[\text{T(Maestro\_Medio)/-}]_{Me} \Delta^{pre_3, pos_3}(\text{T(Resultados)}) \tag{7.3.33}$$

$$\begin{aligned} &\text{T(Calcula\_Resultado)} \overset{pre_4}{\rightarrow} \overset{pos_{4A}}{} \text{T(Maestro\_Medio)} + \\ &\text{T(Calcula\_Resultado)} \overset{pre_4}{\rightarrow} \overset{pos_{4B}}{} \text{T(Anuncia\_Resultado)} \end{aligned} \tag{7.3.34}$$

$$\text{T(Anuncia\_Resultado)} \overset{pre_5}{\rightarrow} \overset{pos_5}{} \text{T(Fin)} \tag{7.3.35}$$

7.3.6. Especificación del medio LeGESD

El medio LeGESD especificado en el presente ejemplo es el mismo que ha sido explicado en el sistema productor-consumidor, presentando como cambios únicamente los nombres de los estados Punto de acceso y los mensajes utilizados en la especificación de la cena de criptógrafos. Para consultar a detalle la descripción del medio LeGESD revise el ejemplo del sistema productor-consumidor.

El medio LeGESD es descrito a través de un diagrama de comunicación, un diagrama de comunicación se compone de dos diagramas: el diagrama de comunicación del medio de inicialización (Figura 7.12) y el diagrama de comunicación del medio de ejecución (Figura 7.13). En ambos diagramas se muestra el grafo dirigido que describe tanto la inicialización como la ejecución de la comunicación del medio LeGESD, los cuales contienen un conjunto de símbolos gráficos: estados y enlaces.

Para proporcionar sencillez en la interpretación del diagrama de comunicación del medio LeGESD, se utilizan las Tablas 7.15 y 7.16 para agrupar la precondición (que activa a la transición), y la post-condición (generada después de la transición) relacionadas con cada transición de los diagramas del medio de inicialización y del medio de ejecución. Las Tablas 7.15 y 7.16 asocian a cada precondición y post-condición una etiqueta, la cual se utilizará como nombre corto de la precondición o post-condición en la transición a la que corresponda.

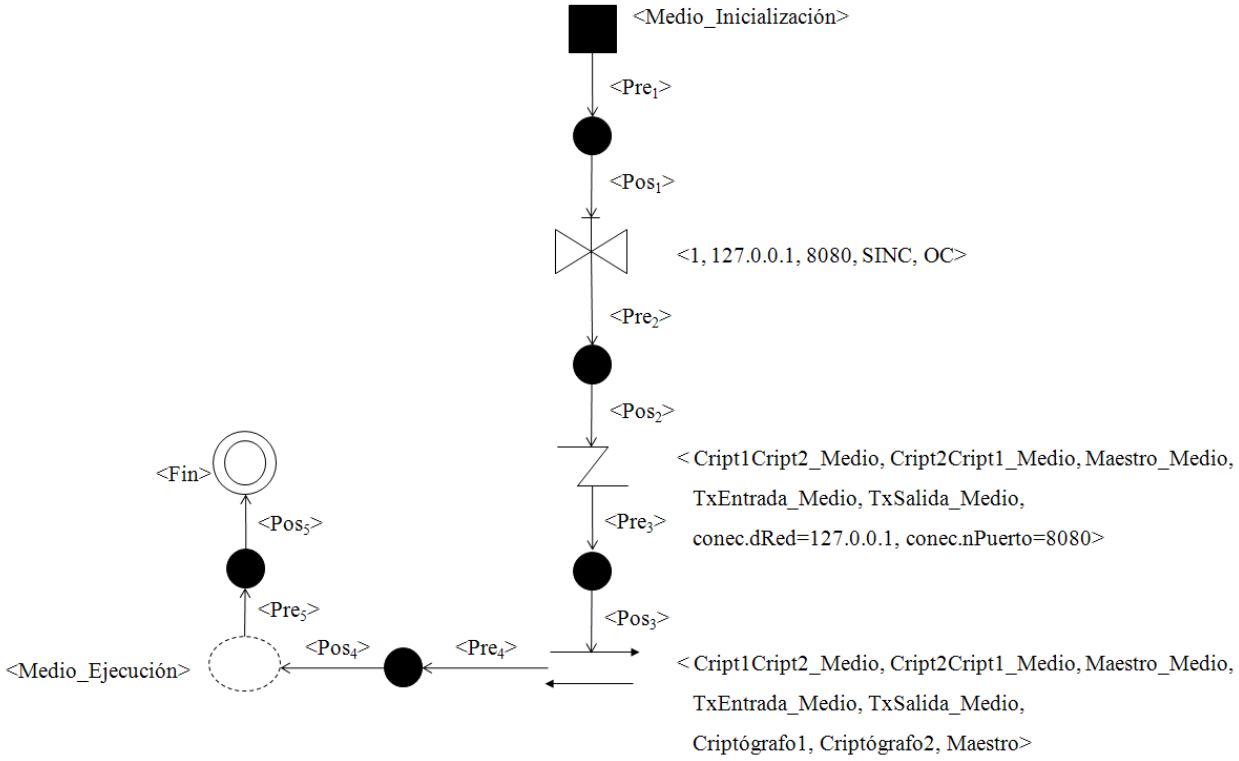


Figura 7.12: Diagrama de comunicación del medio de inicialización LeGESD para la cena de criptógrafos.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, atd conec=null, int v=0>
pos_1	Post-condición	<2, conec=null>
pre_2	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_2	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080>
pre_3	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_3	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=0>
pre_4	Precondición	<1, conec.dRed!=null && conec.nPuerto!=0>
pos_4	Post-condición	<2, conec.dRed=127.0.0.1, conec.nPuerto=8080, v=1>
pre_5	Precondición	<1, v==1>
pos_5	Post-condición	<2, v=1 >

Tabla 7.15: Precondiciones y post-condiciones de las transiciones del diagrama de comunicación del medio de inicialización para la cena de criptógrafos.

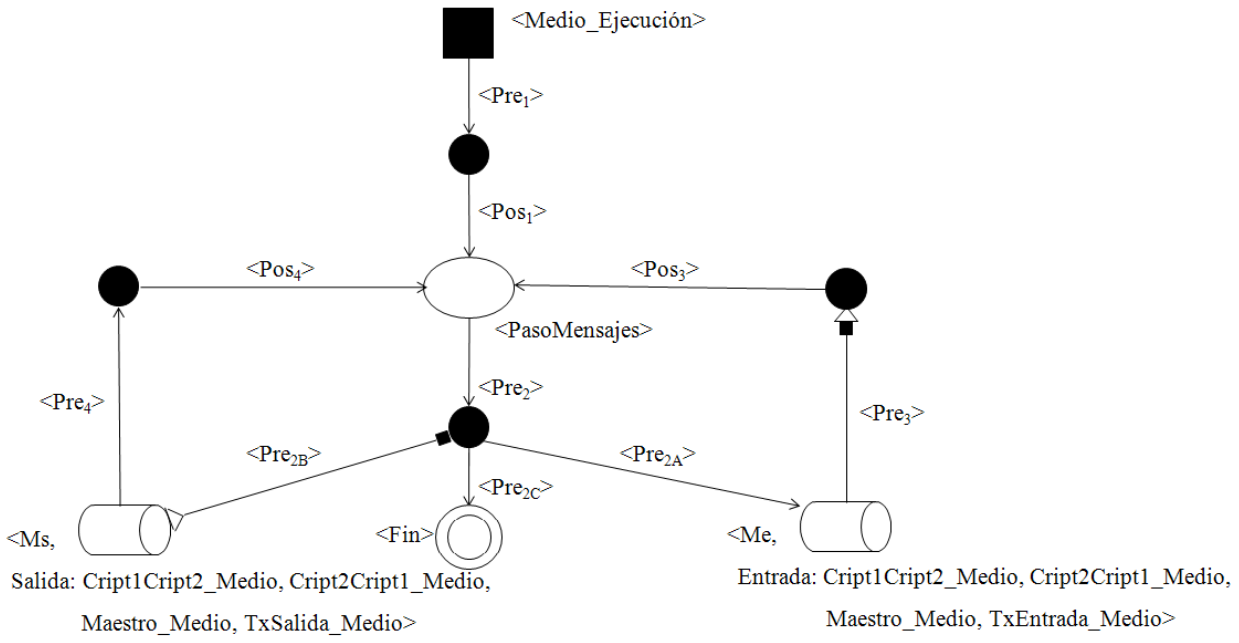


Figura 7.13: Diagrama de comunicación del medio de ejecución LeGESD para la cena de criptógrafos.

Etiqueta	Tipo	Valor
pre_1	Precondición	<1, msj men=null, int v==0>
pos_1	Post-condición	<2, men=null, v=0>
pre_2	Precondición	<1, men!=null && v==0>
pos_{2A}	Post-condición	<2, men=null, v=0>
pos_{2B}	Post-condición	<2, men!=null, v=0>
pos_{2C}	Post-condición	<2, v=1>
pre_3	Precondición	<1, men!=null>
pos_3	Post-condición	<2, men=menEntrada>
pre_4	Precondición	<1, men==null>
pos_4	Post-condición	<2, men=null>
Me	Mensaje de entrada	Criptógrafo1: [Me1, Me2], Criptógrafo2: [Me1, Me2], Maestro: [Me]
Ms	Mensaje de salida	Criptógrafo1: [Ms1, Ms2, Ms3], Criptógrafo2: [Ms1, Ms2, Ms3]

Tabla 7.16: Precondiciones y post-condiciones de las transiciones del diagrama de comunicación del medio de ejecución para la cena de criptógrafos.

Las ecuaciones de transición para el medio de inicialicación se obtienen después de aplicar el Algoritmo 1 al diagrama de comportamiento correspondiente, siendo las siguientes:

$$\text{T(Medio)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Apertura)} \tag{7.3.36}$$

$$\text{T(Apertura)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Atado)} \tag{7.3.37}$$

$$\text{T(Atado)} \overset{pre_3}{\rightarrow} \overset{pos_3}{} \text{T(Comienzo)} \tag{7.3.38}$$

$$\text{T(Comienzo)} \overset{pre_4}{\rightarrow} \overset{pos_4}{} \text{T(Medio\_Ejecución)} \tag{7.3.39}$$

$$\text{T(Medio\_Ejecución)} \overset{pre_5}{\rightarrow} \overset{pos_5}{} \text{T(Fin)} \tag{7.3.40}$$

Las ecuaciones de transición para el medio de ejecución se obtienen después de aplicar el Algoritmo 1 al diagrama de comportamiento correspondiente, siendo las siguientes:

$$\text{T(Medio\_Ejecución)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(PasoMensajes)} \tag{7.3.41}$$

$$\begin{aligned} &\text{T(PasoMensajes)}^{pre_2} \overset{pos_{2A}}{\rightarrow} \text{T(Entrada)} + \\ &\text{T(PasoMensajes)}^{pre_2} \overset{pos_{2B}}{\rightarrow} \text{T(Salida)} + \\ &\text{T(PasoMensajes)}^{pre_2} \overset{pos_{2C}}{\rightarrow} \text{T(Fin)} \end{aligned} \tag{7.3.42}$$

$$[\text{T(Entrada)/-}]_{Me}\Delta^{pre_3,pos_3}(\text{T(PasoMensajes)}) \tag{7.3.43}$$

$$\text{T(Salida)}\Delta^{pre_4,pos_4,Ms}(\text{T(PasoMensajes)}) \tag{7.3.44}$$

Con esto concluye la especificación completa en el lenguaje LeGESD del sistema cena de criptógrafos. Para el ejemplo sólo se consideró la existencia de dos criptógrafos y un maestro, sin embargo se puede generalizar para N criptógrafos debido la existencia del trabajo LeGESD Maestro, el cual se encarga de verificar el resultado del sistema permitiendo la comunicación con múltiples criptógrafos. En el siguiente apartado se realiza un análisis de resultados y la comprobación de la hipótesis de trabajo de la presente tesis.

7.4. Análisis de resultados y comprobación de hipótesis

Los resultados obtenidos a partir de los ejemplos desarrollados del sistema productor-consumidor y del sistema cena de criptógrafos son sus especificaciones tanto gráfica como en el álgebra de procesos de LeGESD. El análisis de los resultados consiste en la comprobación de las propiedades de seguridad de estos sistemas mediante el álgebra de procesos de LeGESD (ADSD). La comprobación de las propiedades de seguridad del sistema productor-consumidor, así como también del sistema cena de criptógrafos, se realiza a través de una técnica del álgebra de procesos llamada *equivalencia por traza* [61]. La selección de esta técnica es debido a que varias propiedades de seguridad son formalizadas utilizando la noción de equivalencia por traza del álgebra de procesos [61].

De manera general, la equivalencia por traza establece que dos procesos p y q son equivalentes si sus conjuntos respectivos de trazas son iguales. En ADSD, las ecuaciones de transición obtenidas representan el conjunto de trazas de un proceso dado, y para obtener la equivalencia por traza se compara la ecuación de transición en un punto dado de un proceso p con la ecuación de transición en el mismo punto de un proceso q, si ambas ecuaciones son iguales entonces existe equivalencia por traza.

Para la comprobación de las propiedades de seguridad del sistema productor-consumidor, se utiliza la activación de algunas de las ecuaciones de transición (en una traza) obtenidas para cada trabajo LeGESD del ejemplo.

Propiedad 1. El sistema nunca remueve un elemento de un búfer vacío o intenta agregar un elemento a un búfer lleno.

Comprobación. La primera parte de la comprobación consiste en verificar que nunca se remueve un elemento de un búfer vacío; esto implica comprobar que la siguiente ecuación de transición dada en un proceso que nunca remueve un elemento de un búfer vacío se cumple:

$$[\text{T(ConsumidorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$$

Para ello hacemos uso de las ecuaciones de transición 7.2.21, 7.2.22 y 7.2.26 del trabajo Consumidor asumiendo que Ne=0 (búfer vacío):

En la Ecuación 7.2.21 se realiza una transición del proceso T(Consumidor) al proceso T(Espera) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas:

$$\text{T(Consumidor)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Esperar)}$$

En la Ecuación 7.2.22 se realiza una transición del proceso T(Esperar) al proceso T(Consumidor Búfer_ Medio1) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de *Ne* es igual con cero; este valor es el asumido como hipótesis (Ne=0). Por lo tanto, la expresión a la derecha del operador + es la que se activa, generándose la transición correspondiente con la post-condición pos_2 siendo verdadera:

$$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2A}}\text{T(Consumir)} + \text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(ConsumidorBúfer\_Medio1)} =$$

$$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(ConsumidorBúfer\_Medio1)}$$

En la Ecuación 7.2.26 no se realiza la transición del proceso T(ConsumidorBúfer_Medio1) al proceso T(Esperar) debido a que la precondición pre_6 resulta falsa. La precondición permanecerá como falsa hasta que el trabajo LeGESD Búfer envíe un mensaje donde el valor de Ne sea diferente de cero y que exista un elemento a consumir, mientras tanto el Consumidor permanecerá en el proceso T(ConsumidorBúfer_Medio1):

$$[\text{T(ConsumidorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$$

De 7.2.21 se tiene que para cambiar del proceso T(Consumidor) hacia T(Esperar) se debe cumplir como precondición que Ne sea cero, es decir, que el búfer esté vacío inicialmente. En 7.2.22 se tiene que al ser Ne==0 (búfer vacío), el operador + de la ecuación cambia al proceso T(ConsumidorBúfer_Medio1). De este proceso por el operador Δ*, no se realizará una transición hasta que se reciban los mensajes msjBuf1.vDato y msjBuf2.vDato desde el trabajo Búfer, por lo tanto el trabajo Consumidor queda bloqueado al no existir elementos a consumir, desbloqueándose cuando se reciben los mensajes correspondientes desde el trabajo Búfer, avisando que hay elementos a consumir*.

$\Rightarrow$ El sistema nunca remueve un elemento de un búfer vacío.

La segunda parte de la comprobación consiste en verificar que nunca se intenta agregar un elemento a un búfer lleno; esto implica comprobar que la siguiente ecuación de transición dada en un proceso que nunca intenta agregar un elemento a un búfer lleno se cumple:

$[\text{T(ProductorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$

Para ello hacemos uso de las ecuaciones de transición 7.2.7, 7.2.8 y 7.2.12 del trabajo Productor asumiendo que Ns=0 (búfer lleno):

En la Ecuación 7.2.7 se realiza una transición del proceso T(Productor) al proceso T(Espera) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas:

$\text{T(Productor)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Esperar)}$

En la Ecuación 7.2.8 se realiza una transición del proceso T(Esperar) al proceso T(Productor Búfer_ Medio1) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de *Ns* es igual con cero; este valor es el asumido como hipótesis (Ns=0). Por lo tanto, la expresión a la derecha del operador + es la que se activa, generándose la transición correspondiente con la post-condición pos_2 siendo verdadera:

$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2A}}\text{T(Producir)} + \text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(ProductorBúfer\_Medio1)} =$

$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(ProductorBúfer\_Medio1)}$

En la Ecuación 7.2.12 no se realiza la transición del proceso T(ProductorBúfer_Medio1) al proceso T(Esperar), debido a que la precondición pre_6 resulta falsa. La precondición permanecerá como falsa hasta que el trabajo LeGESD Búfer envíe un mensaje donde el valor de Ns sea diferente de cero y que exista un elemento producido, mientras tanto el Productor permanecerá en el proceso T(ProductorBúfer_Medio1):

$[\text{T(ProductorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$

De 7.2.7 se tiene que para cambiar del proceso T(Productor) hacia T(Esperar) se debe cumplir como precondición que Ns sea N, es decir, que el búfer esté vacio inicialmente. Asumiendo en 7.2.8 que Ns==0 (búfer lleno), el operador + de la ecuación cambia al proceso T(ProductorBúfer_ Medio1). De este proceso por el operador Δ*, no se realizará una transición hasta que se reciba el mensaje msjBuf.vDato desde el trabajo Búfer; es decir, que el trabajo Productor queda bloqueado por no existir espacio en el búfer para producir elementos, desbloqueándose cuando se recibe el mensaje correspondiente del trabajo Búfer, avisando que hay espacio en el búfer.*

$\Rightarrow$ El sistema nunca intenta agregar un elemento a un búfer lleno.

La comprobación está completa.

$\rightarrow$ La propiedad 1 ha sido comprobada.

Propiedad 2. El sistema no provoca un abrazo mortal.

Comprobación. Un abrazo mortal puede ocurrir cuando tanto el trabajo Productor como el Consumidor estén bloqueados, es decir que Ne=0 y Ns=0, lo que significaría que Ne=0 y Ne=N ocurren al mismo tiempo. Esto implica comprobar que las siguientes ecuaciones de transición dadas en un proceso que no provoca un abrazo mortal se cumplen:

$[\text{T(ConsumidorBúfer\_Medio2)/-}]_{Me2}\Delta^{pre_8,pos_8}\text{(T(Señal))}$

$[\text{T(ProductorBúfer\_Medio2)/-}]_{Me1}\Delta^{pre_4,pos_4}\text{(T(Buferizado))}$

Para comprobar que esto no es posible en la especificación realizada, se utilizan las ecuaciones de transición 7.2.13, 7.2.14 del trabajo Búfer evaluándolas con Ne=0 y Ne=N al mismo tiempo.

Con Ne=N se tiene que:

En la Ecuación 7.2.13 se realiza una transición del proceso T(Búfer) al proceso T(Señal) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas:

$\text{T(Búfer)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Señal)}$

En la Ecuación 7.2.14 se realiza una transición del proceso T(Señal) al proceso T(Consumidor Búfer_ Medio1) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de Ne es igual con N, este valor es el asumido como hipótesis (Ne=N). Por lo tanto, la expresión a la derecha del primer operador + es la que se activa, generándose la transición correspondiente con la post-condición pos_{2B} y el mensaje dos de salida $Ms2$ siendo verdaderos:

$\text{T(Señal)}\Delta^{pre_2,pos_{2A},Ms1}\text{(T(ProductorBúfer\_Medio1))} +$

$\text{T(Señal)}\Delta^{pre_2,pos_{2B},Ms2}\text{(T(ConsumidorBúfer\_Medio1))} +$

$\text{T(Señal)}^{pre_2}\!\!\rightarrow^{pos_{2C}}\text{T(Fin)} =$

$\text{T(Señal)}\Delta^{pre_2,pos_{2B},Ms2}\text{(T(ConsumidorBúfer\_Medio1))}$

En la Ecuación 7.2.19 se realiza una transición del proceso T(ConsumidorBúfer_Medio1) al proceso T(ConsumidorBúfer_Medio2) debido a que la precondición pre_7 y la post-condición pos_7 son verdaderas:

$\text{T(ConsumidorBúfer\_Medio1)}^{pre_7}\!\!\rightarrow^{pos_7}\text{T(ConsumidorBúfer\_Medio2)}$

En la Ecuación 7.2.20 no se realiza la transición del proceso T(ConsumidorBúfer_Medio2) al proceso T(Señal) debido a que la precondición pre_8 resulta falsa. La precondición permanecerá como falsa hasta que el trabajo LeGESD Consumidor envíe un mensaje $Me2$ al trabajo LeGESD Búfer, donde el valor del mensaje recibido por Búfer sea diferente de nulo, lo que implica la existencia de un espacio libre en el búfer. Mientras no se reciba este mensaje, Búfer permanecerá en el proceso T(ConsumidorBúfer_Medio2):

$[$T(ConsumidorBúfer_Medio2)/-$]_{Me2}\Delta^{pre_8,pos_8}$(T(Señal))

De 7.2.13 se tiene que para cambiar del proceso T(Búfer) hacia T(Señal) se debe cumplir dentro de la precondición que Ne sea cero, es decir, que el búfer esté vacío inicialmente. Asumiendo en 7.2.14 que Ne==N (búfer lleno), el segundo operador + de la ecuación cambia al proceso T(ConsumidorBúfer_Medio1). De 7.2.19 se tiene que para cambiar del proceso T(ConsumidorBúfer_Medio1) hacia T(ConsumidorBúfer_ Medio2) se debe cumplir dentro de la precondición que Ne sea diferente de cero, lo cual corresponde a la hipótesis (Ne=N). Del proceso T(ConsumidorBúfer_ Medio2) por el operador Δ*, no se realizará una transición hasta que se reciba el mensaje msj-Cons.vDato desde el trabajo Consumidor, es decir, que el trabajo Búfer queda bloqueado por no existir espacio en el búfer para agregar elementos, sin embargo el trabajo Consumidor no se bloquea, de hecho consume y desbloquea al trabajo Búfer cuando envía el mensaje correspondiente avisando que ha consumido un elemento del búfer.*

$\Rightarrow$ El trabajo Consumidor no se bloquea cuando Ne=N.

Con Ne=0 se tiene que:

Partiendo de la Ecuación 7.2.14 se realiza una transición del proceso T(Señal) al proceso T(Consumidor Búfer_ Medio1) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de Ne es igual con cero, este valor es el asumido como hipótesis (Ne=0). Por lo tanto, la expresión a la izquierda del primer operador + es la que se activa, generándose la transición correspondiente con la post-condición pos_{2A} y el mensaje uno de salida $Ms1$ siendo verdaderos:

T(Señal)$\Delta^{pre_2,pos_{2A},Ms1}$(T(ProductorBúfer_Medio1)) +

T(Señal)$\Delta^{pre_2,pos_{2B},Ms2}$(T(ConsumidorBúfer_Medio1)) +

T(Señal)$^{pre_2}\rightarrow^{pos_{2C}}$T(Fin) =

T(Señal)$\Delta^{pre_2,pos_{2A},Ms1}$(T(ProductorBúfer_Medio1))

En la Ecuación 7.2.15 se realiza una transición del proceso T(ProductorBúfer_Medio1) al proceso T(ProductorBúfer_Medio2) debido a que la precondición pre_3 y la post-condición pos_3 son verdaderas:

T(ProductorBúfer_Medio1)$^{pre_3}\rightarrow^{pos_3}$T(ProductorBúfer_Medio2)

En la Ecuación 7.2.16 no se realiza la transición del proceso T(ProductorBúfer_Medio2) al proceso T(Buferizado) debido a que la precondición pre_8 resulta falsa. La precondición permanecerá como falsa hasta que el trabajo LeGESD Productor envíe un mensaje $Me1$ al trabajo LeGESD Búfer, donde el valor del mensaje recibido por Búfer sea diferente de nulo, lo que implica la existencia de un elemento producido en el búfer. Mientras no se reciba este mensaje, Búfer perma-

necerá en el proceso T(ProductorBúfer_Medio2):

$[\text{T(ProductorBúfer\_Medio2)/-}]_{Me1}\Delta^{pre_4,pos_4}(\text{T(Buferizado)})$

Asumiendo en 7.2.14 que Ne==0 (búfer vacio), el operador + de la ecuación cambia al proceso T(Productor Búfer_Medio1) debido a Ne=0 (lo que implica que Ns>0). De 7.2.15 se tiene que para cambiar del proceso T(ProductorBúfer_Medio1) hacia T(ProductorBúfer_Medio2) se debe cumplir dentro de la precondición que Ns sea diferente de cero, lo cual incluye a la hipótesis (Ne=0). De este proceso por el operador Δ*, no se realizará una transición hasta que se reciban los mensajes msjProd1.vDato y msjProd2.vDato desde el trabajo Productor, es decir, que el trabajo Búfer queda bloqueado esperando por el elemento producido a agregar en el búfer, pero el trabajo Productor no se bloquea, de hecho produce y desbloquea al trabajo Búfer cuando envía el mensaje correspondiente avisando que ha producido un elemento a agregar en el búfer.*

$\Rightarrow$ El trabajo Productor no se bloquea cuando Ne=0.

Como se demostró, los conjuntos de ecuaciones de transición para Ne==N y Ne==0 no son posibles al mismo tiempo, además tanto el trabajo Consumidor como Productor nunca se bloquean, la comprobación está completa.

$\rightarrow$ La propiedad 2 ha sido comprobada.

Propiedad 3. El sistema no provoca inanición de algún proceso.

Comprobación. Implica comprobar que las siguientes ecuaciones de transición dadas en un proceso que no provoca inanición de algún proceso se cumplen:

$[\text{T(ProductorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$

$\text{T(Señal)}\Delta^{pre_4,pos_4,Ms}(\text{T(ConsumidorBúfer\_Medio2)})$

Se asume que el trabajo Productor está bloqueado en el proceso T(ProductorBúfer _Medio1) después de activar las Ecuaciones 7.2.7 a 7.2.12:

$\text{T(Productor)}^{pre_1}\!\!\rightarrow^{pos_1}\text{T(Esperar)}$

$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2A}}\text{T(Producir)} + \text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(ProductorBúfer\_Medio1)} =$

$\text{T(Esperar)}^{pre_2}\!\!\rightarrow^{pos_{2A}}\text{T(Producir)}$

$[\text{T(ProductorBúfer\_Medio1)/-}]_{Me}\Delta^{pre_6,pos_6}(\text{T(Esperar)})$

$\text{T(Señal)}\Delta^{pre_4,pos_4,Ms}(\text{T(ProductorBúfer\_Medio2)})$

$T(\text{ProductorBúfer\_Medio2})^{pre_5} \overset{pos_{5A}}{\rightarrow} T(\text{ProductorBúfer\_Medio1}) +$

$T(\text{ProductorBúfer\_Medio2})^{pre_5} \overset{pos_{5B}}{\rightarrow} T(\text{Fin}) =$

$T(\text{ProductorBúfer\_Medio2})^{pre_5} \overset{pos_{5A}}{\rightarrow} T(\text{ProductorBúfer\_Medio1})$

$[T(\text{ProductorBúfer\_Medio1})/\text{-}]_{Me}\Delta^{pre_6,pos_6}(T(\text{Esperar}))$

El trabajo Consumidor interactúa con el trabajo Productor de forma concurrente activando las Ecuaciones 7.2.22 a 7.2.24, las cuales desbloquean al Productor:

$T(\text{Esperar})^{pre_2} \overset{pos_{2A}}{\rightarrow} T(\text{Consumir}) + T(\text{Esperar})^{pre_2} \overset{pos_{2B}}{\rightarrow} T(\text{ConsumidorBúfer\_Medio1}) =$

$T(\text{Esperar})^{pre_2} \overset{pos_{2A}}{\rightarrow} T(\text{Consumir})$

$T(\text{Consumir})^{pre_3} \overset{pos_3}{\rightarrow} T(\text{Señal})$

$T(\text{Señal})\Delta^{pre_4,pos_4,Ms}(T(\text{ConsumidorBúfer\_Medio2}))$

En 7.2.12 se tiene que Ns==0 (búfer lleno), quedando Productor en el proceso T(ProductorBúfer_Medio1). De este proceso por el operador Δ y su precondición pre_6, no se realizará una transición hasta que se reciba el mensaje msjBuf.vDato desde el trabajo Búfer, por lo que el trabajo Productor está bloqueado. En 7.2.24 se cambia al estado T(ConsumidorBúfer_Medio2), de este estado por el operador Δ y su precondición pre_4, se realizará una transición una vez enviado el mensaje msjCons.vDato hacia el trabajo Búfer. Al recibirse este mensaje en el trabajo Búfer, éste envía el mensaje msjBuf.vDato al trabajo Productor desbloqueándolo.

$\Rightarrow$ El trabajo Productor no presenta inanición.

De forma análoga se puede comprobar que el trabajo Consumidor no presenta inanición. Por lo tanto la comprobación está completa.

$\rightarrow$ La propiedad 3 ha sido comprobada.

El análisis de resultados del sistema cena de criptógrafos consiste en la comprobación de la propiedad de seguridad del sistema mediante el álgebra de procesos de LeGESD (ADSD).

Propiedad. Dada la información pública que los criptógrafos comparten, si alguno de ellos pagó la cuenta entonces se mantiene el anonimato del criptógrafo quien pagó la cuenta, o si ninguno de ellos pagó la cuenta entonces se determina que la NSA pagó la cuenta.

Recordando lo mencionado al inicio del capítulo acerca de la verificación de la propiedad de seguridad para la cena de criptógrafos, se tiene que para verificar que el protocolo es seguro (anonimato conservado), es necesario observar las situaciones para las cuales un criptógrafo (para el ejemplo

es el trabajo Maestro), que no ha pagado la cuenta, desea averiguar cuál criptógrafo ha pagado. Lo anterior implica comprobar tres posibles casos para el ejemplo desarrollado del sistema cena de criptógrafos:

- Sean los volados de las dos monedas que Maestro observa los mismos, y Criptógrafo1 anunció 1 (diferentes) y Criptógrafo2 anunció 0 (iguales). El volado de la moneda que Maestro no observa es el mismo que los volados de las monedas que él observa. Para el ejemplo desarrollado, lo anterior significa que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron iguales (rVolado=0 (1) y rVolado=0 (1) respectivamente), y Criptógrafo1 decidió pagar (pague=1) anónimamente la cuenta.
- Sean los volados de las dos monedas que Maestro observa los mismos, y Criptógrafo1 anunció 1 (diferentes) y Criptógrafo2 anunció 0 (iguales). El volado de la moneda que Maestro no observa es distinto al de los volados de las monedas que él observa. Para el ejemplo desarrollado, lo anterior significa que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron diferentes (rVolado=0 y rVolado=1 respectivamente) y Criptógrafo2 decidió pagar (pague=1) anónimamente la cuenta.
- Sean los volados de las dos monedas que Maestro observa distintos. Si los otros dos criptógrafos anunciaron 1 entonces quien pagó la cuenta está más cercano a la moneda cuyo volado es el mismo al que Maestro no observa. Por otra parte, si los otros dos criptógrafos anunciaron 0 entonces quien pagó está más cercano a la moneda cuyo volado es diferente al que Maestro no observa. Para el ejemplo desarrollado, lo anterior significa que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron iguales (rVolado=0 (1) y rVolado=0 (1) respectivamente) y que ninguno de ellos decidió pagar (pague=0) anónimamente la cuenta.

Comprobación. La comprobación consiste en realizar tres trazas del sistema (una para cada caso de los mencionados anteriormente), utilizando las ecuaciones de transición activadas de acuerdo a la traza hecha en cada trabajo LeGESD del ejemplo. La primer traza asume que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron iguales (rVolado=0 (1) y rVolado=0 (1) respectivamente), y que Criptógrafo1 decidió pagar (pague=1) anónimamente la cuenta. Esto implica comprobar que el anonimato de Criptógrafo1 se conserva, es decir que la siguiente ecuación de transición dada en un proceso con las condiciones asumidas se cumple:

$$\text{T(Calcula\_Resultados)} \overset{pre_4}{\rightarrow} \overset{pos_{4B}}{} \text{T(Anuncia\_Resultado)}$$

En la Ecuación 7.3.7 se realiza una transición del proceso T(Criptógrafo1) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas. De forma concurrente, en la Ecuación 7.3.19 se realiza una transición del proceso T(Criptógrafo2) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 también son verdaderas:

$$\text{T(Criptógrafo1)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Volado)}$$

$$\text{T(Criptógrafo2)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Volado)}$$

En la Ecuación 7.3.8 y 7.3.20 se realiza una transición del proceso T(Volado) al proceso T(Compartir) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de *rVolado* es igual con cero (o uno) tanto en Criptógrafo1 como en Criptógrafo2, estos valores son los asumidos como hipótesis (rVolado=0 (1) de Criptógrafo1 y rVolado=0 (1) de Criptógrafo2). Para esta traza se asume que los dos criptógrafos obtuvieron el mismo resultado en el volado, generándose la transición correspondiente con la post-condición pos_2 siendo verdadera:

$$\text{T(Volado)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Compartir)}$$

En la Ecuación 7.3.9 y 7.3.21 se realiza una transición del proceso T(Compartir) al proceso T(Cript1Cript2_Medio) (T(Cript2Cript1_Medio)) debido a que la precondición pre_3 resulta verdadera cuando se verifica que el valor de $msjCript1$ ($msjCript2$) es igual con nulo, entonces se genera la transición correspondiente con la post-condición pos_3 siendo verdadera. En el proceso T(Cript1Cript2_Medio) (T(Crip2Cript1_Medio)) se envía el mensaje $Ms1$ al criptógrafo vecino de Criptógrafo1 (Criptógrafo2), conteniendo el resultado del volado lanzado *rVolado* por Criptógrafo1 (Criptógrafo2).

$$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript1Cript2\_Medio))}$$

$$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript2Cript1\_Medio))}$$

En la Ecuación 7.3.10 y 7.3.22 se realiza una transición del proceso T(Cript1Cript2_Medio) (T(Cript2Cript1_Medio)) al proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) debido a que la precondición pre_4 resulta verdadera, entonces se genera la transición correspondiente con la post-condición pos_4 siendo verdadera.

$$\text{T(Cript1Cript2\_Medio)} \overset{pre_4}{\rightarrow} \overset{pos_4}{} \text{T(Cript2Cript1\_Medio)}$$

$$\text{T(Cript2Cript1\_Medio)} \overset{pre_4}{\rightarrow} \overset{pos_4}{} \text{T(Cript1Cript2\_Medio)}$$

En la Ecuación 7.3.11 y 7.3.23 se realiza una transición del proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) al proceso T(Compartir) debido a que la precondición pre_5 resulta verdadera. Entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera. En el proceso T(Cript2Cript1_Medio) (T(Crip1Cript2_Medio)) se recibe el mensaje $Me1$ del criptógrafo vecino Criptógrafo2 (Criptógrafo1), conteniendo el resultado del volado lanzado *vVolado* por Criptógrafo2 (Criptógrafo1).

$$\text{[T(Cript2Cript1\_Medio)/-]}_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))}$$

$$\text{[T(Cript1Cript2\_Medio)/-]}_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))}$$

En la Ecuación 7.3.12 y 7.3.24 se realiza una transición del proceso T(Compartir) al proceso T(Decisión_Pago) a través de dos posibles caminos, de acuerdo al valor de la precondición pre_6 que resulte verdadera (para la hipótesis se asume rVolado==vVolado); entonces se activa la expresión a la izquierda del operador + de la ecuación de transición, generándose la transición correspondiente

con la post-condición pos_{6A} (anuncio=0) siendo verdadera.

$$T(\text{Compartir})^{pre_6} \overset{pos_{6A}}{\rightarrow} T(\text{Decisión\_Pago}) + T(\text{Compartir})^{pre_6} \overset{pos_{6B}}{\rightarrow} T(\text{Decisión\_Pago})$$

$$= T(\text{Compartir})^{pre_6} \overset{pos_{6A}}{\rightarrow} T(\text{Decisión\_Pago})$$

En la Ecuación 7.3.13 y 7.3.25 se realiza una transición del proceso T(Decisión_Pago) al proceso T(Transmisión) o al proceso T(Cambiar_Resultado), de acuerdo al valor de la precondición pre_7 que resulte verdadera (para la hipótesis se asume pague==0 para Criptógrafo2, y pague==1 para Criptógrafo1). Para Criptógrafo2 se activa la expresión a la izquierda del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{7A} (pague=0) siendo verdadera.

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7A}}{\rightarrow} T(\text{Transmisión}) +$$

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7B}}{\rightarrow} T(\text{Cambiar\_Resultado}) =$$

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7A}}{\rightarrow} T(\text{Transmisión})$$

Mientras que para Criptógrafo1 se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{7B} (pague=1) siendo verdadera.

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7A}}{\rightarrow} T(\text{Transmisión}) +$$

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7B}}{\rightarrow} T(\text{Cambiar\_Resultado}) =$$

$$T(\text{Decisión\_Pago})^{pre_7} \overset{pos_{7B}}{\rightarrow} T(\text{Cambiar\_Resultado})$$

En la Ecuación 7.3.14 se realiza una transición del proceso T(Cambiar_Resultado) al proceso T(Transmisión) de acuerdo al valor de la precondición pre_8 que resulte verdadera (para la hipótesis se asume anuncio==0 para Criptógrafo1); entonces se genera la transición correspondiente con la post-condición pos_8 (anuncio=!anuncio=1) siendo verdadera.

$$T(\text{Cambiar\_Resultado})^{pre_8} \overset{pos_8}{\rightarrow} T(\text{Transmisión})$$

En la Ecuación 7.3.15 y 7.3.27 se realiza una transición del proceso T(Transmisión) al proceso T(Maestro_Medio) o al proceso T(TxSalida_Medio) o al proceso T(Fin), de acuerdo al valor de la precondición pre_8 que resulte verdadera (para la traza msjCript1==null (msjCript2==null)); entonces se activa la expresión a la izquierda del primer operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{9A} (msjCript1.vDato=anuncio=1 (msjCript2.vDato=anuncio=0)) siendo verdadera. En el proceso T(Maestro_Medio) se envía el mensaje $Ms2$ de Criptógrafo1 (Criptógrafo2), conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$T(\text{Transmisión})\Delta^{pre_9, pos_{9A}, Ms2}(T(\text{Maestro\_Medio}))$ +

$T(\text{Transmisión})\Delta^{pre_9, pos_{9B}, Ms3}(T(\text{TxSalida\_Medio}))$ +

$T(\text{Transmisión})^{pre_9} \rightarrow^{pos_{9C}} T(\text{Fin})$ =

$T(\text{Transmisión})\Delta^{pre_9, pos_{9A}, Ms2}(T(\text{Maestro\_Medio}))$

Mientras tanto, en el trabajo LeGESD Maestro que concurrentemente se ejecuta con Criptógrafo1 y Criptógrafo2, en la Ecuación 7.3.33 se realiza una transición del proceso T(Maestro_Medio) al proceso T(Resultados) de acuerdo al valor de la precondición pre_3 que resulte verdadera (para la traza msjCript1!=null (msjCript2!=null)); entonces se genera la transición correspondiente con la post-condición pos_3 (recibido=msjCript1.vDato=1 (recibido=msjCript2.vDato=0), mensajes=mensajes+1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje *Me* tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$[T(\text{Maestro\_Medio})/\text{-}]_{Me}\Delta^{pre_3, pos_3}(T(\text{Resultados}))$

En la Ecuación 7.3.32 se realiza una transición del proceso T(Resultados) al proceso T(Maestro_Medio) o al proceso T(Calcula_Resultados), de acuerdo al valor de la precondición pre_2 que resulte verdadera (para la traza recibido!=-1); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{2B} (recibido!=-1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje *Me* tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2A}} T(\text{Maestro\_Medio})$ + $T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2B}} T(\text{Calcula\_Resultado})$

= $T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2B}} T(\text{Calcula\_Resultado})$

En la Ecuación 7.3.34 se realiza una transición del proceso T(Calcula_Resultado) al proceso T(Maestro_ Medio) o al proceso T(Anuncia_Resultados), de acuerdo al valor de la precondición pre_4 que resulte verdadera (para la traza mensajes==NCriptógrafos); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{4B} (mensajes=NCriptógrafos) siendo verdadera.

$T(\text{Calcula\_Resultado})^{pre_4} \rightarrow^{pos_{4A}} T(\text{Maestro\_Medio})$ +

$T(\text{Calcula\_Resultados})^{pre_4} \rightarrow^{pos_{4B}} T(\text{Anuncia\_Resultado})$ =

$T(\text{Calcula\_Resultados})^{pre_4} \rightarrow^{pos_{4B}} T(\text{Anuncia\_Resultado})$

En la Ecuación 7.3.35 se realiza una transición del proceso T(Anuncia_Resultado) al proceso T(Fin) de acuerdo al valor de la precondición pre_5 que resulte verdadera (para la traza paga-

do_por==’C’); entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera.

$$\text{T(Anuncia\_Resultado)} \overset{pre_5}{\rightarrow} \overset{pos_5}{} \text{T(Fin)}$$

Como resultado final de la primer traza, se observa que tanto Criptógrafo1 como Criptógrafo2 anunciaron públicamente como resultados (anuncio) un valor de uno y de cero respectivamente, después de haber compartido sus respectivos volados. El valor anunciado de uno se debe a que Criptógrafo1 decidió pagar la cuenta de forma anónima. El trabajo LeGESD Maestro funge como anunciante del pago de la cuenta, el cual recibe los anuncios públicos de los criptógrafos y calcula las diferencias, resultando un número impar. Por lo que con la información pública que cada criptógrafo anunció, el Maestro únicamente conoce y anuncia que alguno de los criptógrafos pagó la cuenta pero no puede determinar que Criptógrafo1 pagó la cuenta (anonimato conservado).

⇒ Dada la información pública que los criptógrafos comparten, mediante la primer traza que se realizó del sistema no se tiene forma de conocer cuál criptógrafo pagó la cuenta (anonimato conservado de Criptografo1).

La segunda traza asume que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron diferentes (rVolado=0 y rVolado=1 respectivamente) y que Criptógrafo2 decidió pagar (pague=1) anónimamente la cuenta. Esto implica comprobar que el anonimato de Criptógrafo2 se conserva, es decir que la siguiente ecuación de transición dada en un proceso con las condiciones asumidas se cumple:

$$\text{T(Calcula\_Resultados)} \overset{pre_4}{\rightarrow} \overset{pos_{4B}}{} \text{T(Anuncia\_Resultado)}$$

En la Ecuación 7.3.7 se realiza una transición del proceso T(Criptógrafo1) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas. De forma concurrente, en la Ecuación 7.3.19 se realiza una transición del proceso T(Criptógrafo2) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 también son verdaderas:

$$\text{T(Criptógrafo1)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Volado)}$$

$$\text{T(Criptógrafo2)} \overset{pre_1}{\rightarrow} \overset{pos_1}{} \text{T(Volado)}$$

En la Ecuación 7.3.8 y 7.3.20 se realiza una transición del proceso T(Volado) al proceso T(Compartir) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de *rVolado* es igual con cero en Criptógrafo1 y con uno en Criptógrafo2, estos valores son los asumidos como hipótesis (rVolado=0 de Criptógrafo1 y rVolado=1 de Criptógrafo2). Para esta traza se asume que los dos criptógrafos obtuvieron diferente resultado en el volado, generándose la transición correspondiente con la post-condición pos_2 siendo verdadera:

$$\text{T(Volado)} \overset{pre_2}{\rightarrow} \overset{pos_2}{} \text{T(Compartir)}$$

En la Ecuación 7.3.9 y 7.3.21 se realiza una transición del proceso T(Compartir) al proceso T(Cript1Cript2_Medio) (T(Cript2Cript1_Medio)) debido a que la precondición pre_3 resulta verdadera cuando se verifica que el valor de $msjCript1$ ($msjCript2$) es igual con nulo, entonces se genera la transición correspondiente con la post-condición pos_3 siendo verdadera. En el proceso T(Cript1Cript2_Medio) (T(Crip2Cript1_Medio)) se envía el mensaje $Ms1$ al criptógrafo vecino de Criptógrafo1 (Criptógrafo2), conteniendo el resultado del volado lanzado *rVolado* por Criptógrafo1 (Criptógrafo2).

T(Compartir)$\Delta^{pre_3,pos_3,Ms1}$(T(Cript1Cript2_Medio))

T(Compartir)$\Delta^{pre_3,pos_3,Ms1}$(T(Cript2Cript1_Medio))

En la Ecuación 7.3.10 y 7.3.22 se realiza una transición del proceso T(Cript1Cript2_Medio) (T(Cri pt2Cript1_Medio)) al proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) debido a que la precondición pre_4 resulta verdadera, entonces se genera la transición correspondiente con la post-condición pos_4 siendo verdadera.

T(Cript1Cript2_Medio)$^{pre_4}\!\!\rightarrow^{pos_4}$T(Cript2Cript1_Medio)

T(Cript2Cript1_Medio)$^{pre_4}\!\!\rightarrow^{pos_4}$T(Cript1Cript2_Medio)

En la Ecuación 7.3.11 y 7.3.23 se realiza una transición del proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) al proceso T(Compartir) debido a que la precondición pre_5 resulta verdadera, entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera. En el proceso T(Cript2Cript1_Medio) (T(Crip1Cript2_Medio)) se recibe el mensaje $Me1$ del criptógrafo vecino Criptógrafo2 (Criptógrafo1), conteniendo el resultado del volado lanzado *vVolado* por Criptógrafo2 (Criptógrafo1).

[T(Cript2Cript1_Medio)/-]$_{Me1}\Delta^{pre_5,pos_5}$(T(Compartir))

[T(Cript1Cript2_Medio)/-]$_{Me1}\Delta^{pre_5,pos_5}$(T(Compartir))

En la Ecuación 7.3.12 y 7.3.24 se realiza una transición del proceso T(Compartir) al proceso T(Decisión_Pago) a través de dos posibles caminos, de acuerdo al valor de la precondición pre_6 que resulte verdadera (para la hipótesis se asume rVolado!=vVolado); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{6B} (anuncio=1) siendo verdadera.

T(Compartir)$^{pre_6}\!\!\rightarrow^{pos_{6A}}$T(Decisión_Pago) + T(Compartir)$^{pre_6}\!\!\rightarrow^{pos_{6B}}$T(Decisión_Pago) =

T(Compartir)$^{pre_6}\!\!\rightarrow^{pos_{6B}}$T(Decisión_Pago)

En la Ecuación 7.3.13 y 7.3.25 se realiza una transición del proceso T(Decisión_Pago) al proceso T(Transmisión) o al proceso T(Cambiar_Resultado), de acuerdo al valor de la precondición pre_7 que resulte verdadera (para la hipótesis se asume pague==0 para Criptógrafo1, y pague==1

para Criptógrafo2). Para Criptógrafo1 se activa la expresión a la izquierda del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{7A} (pague=0) siendo verdadera.

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7A}}$T(Transmisión) +

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7B}}$T(Cambiar_Resultado) =

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7A}}$T(Transmisión)

Mientras que para Criptógrafo2 se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{7B} (pague=1) siendo verdadera.

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7A}}$T(Transmisión) +

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7B}}$T(Cambiar_Resultado) =

T(Decisión_Pago)$^{pre_7}\rightarrow^{pos_{7B}}$T(Cambiar_Resultado)

En la Ecuación 7.3.26 se realiza una transición del proceso T(Cambiar_Resultado) al proceso T(Transmisión) de acuerdo al valor de la precondición pre_8 que resulte verdadera (para la hipótesis se asume anuncio==1 para Criptógrafo2); entonces se genera la transición correspondiente con la post-condición pos_8 (anuncio=!anuncio=0) siendo verdadera.

T(Cambiar_Resultado)$^{pre_8}\rightarrow^{pos_8}$T(Transmisión)

En la Ecuación 7.3.15 y 7.3.27 se realiza una transición del proceso T(Transmisión) al proceso T(Maestro_Medio) o al proceso T(TxSalida_Medio) o al proceso T(Fin), de acuerdo al valor de la precondición pre_8 que resulte verdadera (para la traza msjCript1==null (msjCript2==null)); entonces se activa la expresión a la izquierda del primer operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{9A} (msjCript1.vDato=anuncio=1 (msjCript2.vDato=anuncio=0)) siendo verdadera. En el proceso T(Maestro_Medio) se envía el mensaje $Ms2$ de Criptógrafo1 (Criptógrafo2), conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

T(Transmisión)$\Delta^{pre_9,pos_{9A},Ms2}$(T(Maestro_Medio)) +

T(Transmisión)$\Delta^{pre_9,pos_{9B},Ms3}$(T(TxSalida_Medio)) +

T(Transmisión)$^{pre_9}\rightarrow^{pos_{9C}}$T(Fin) =

T(Transmisión)$\Delta^{pre_9,pos_{9A},Ms2}$(T(Maestro_Medio))

Mientras tanto, en el trabajo LeGESD Maestro que concurrentemente se ejecuta con Criptógrafo1 y Criptógrafo2, en la Ecuación 7.3.33 se realiza una transición del proceso T(Maestro_Medio) al proceso T(Resultados) de acuerdo al valor de la precondición pre_3 que resulte verdadera (para la traza msjCript1!=null (msjCript2!=null)); entonces se genera la transición correspondiente con la post-condición pos_3 (recibido=msjCript1.vDato=1 (recibido=msjCript2.vDato=0), mensajes=mensajes+1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje *Me* tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$$[\text{T(Maestro\_Medio)/-}]_{Me}\Delta^{pre_3,pos_3}(\text{T(Resultados)})$$

En la Ecuación 7.3.32 se realiza una transición del proceso T(Resultados) al proceso T(Maestro_Medio) o al proceso T(Calcula_Resultados), de acuerdo al valor de la precondición pre_2 que resulte verdadera (para la traza recibido!=-1); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{2B} (recibido!=-1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje *Me* tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$$\text{T(Resultados)}^{pre_2}\!\!\rightarrow^{pos_{2A}}\text{T(Maestro\_Medio)} + \text{T(Resultados)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(Calcula\_Resultado)} =$$

$$\text{T(Resultados)}^{pre_2}\!\!\rightarrow^{pos_{2B}}\text{T(Calcula\_Resultado)}$$

En la Ecuación 7.3.34 se realiza una transición del proceso T(Calcula_Resultado) al proceso T(Maestro_ Medio) o al proceso T(Anuncia_Resultados), de acuerdo al valor de la precondición pre_4 que resulte verdadera (para la traza mensajes==NCriptógrafos); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{4B} (mensajes=NCriptógrafos) siendo verdadera.

$$\text{T(Calcula\_Resultado)}^{pre_4}\!\!\rightarrow^{pos_{4A}}\text{T(Maestro\_Medio)} +$$

$$\text{T(Calcula\_Resultados)}^{pre_4}\!\!\rightarrow^{pos_{4B}}\text{T(Anuncia\_Resultado)} =$$

$$\text{T(Calcula\_Resultados)}^{pre_4}\!\!\rightarrow^{pos_{4B}}\text{T(Anuncia\_Resultado)}$$

En la Ecuación 7.3.35 se realiza una transición del proceso T(Anuncia_Resultado) al proceso T(Fin) de acuerdo al valor de la precondición pre_5 que resulte verdadera (para la traza pagado_por=='C'); entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera.

$$\text{T(Anuncia\_Resultado)}^{pre_5}\!\!\rightarrow^{pos_5}\text{T(Fin)}$$

Como resultado final de la segunda traza, se observa que tanto Criptógrafo1 como Criptógrafo2 anunciaron públicamente como resultados (anuncio) un valor de uno y de cero respectivamente, después de haber compartido sus respectivos volados. El valor anunciado de cero se debe a

que Criptógrafo2 decidió pagar la cuenta de forma anónima. El trabajo LeGESD Maestro funge como anunciante del pago de la cuenta, el cual recibe los anuncios públicos de los criptógrafos y calcula las diferencias, resultando un número impar. Por lo que con la información pública que cada criptógrafo anunció, el Maestro únicamente conoce y anuncia que alguno de los criptógrafos pagó la cuenta pero no puede determinar que Criptógrafo2 pagó la cuenta (anonimato conservado).

⇒ Dada la información pública que los criptógrafos comparten, mediante la segunda traza que se realizó del sistema no se tiene forma de conocer cuál criptógrafo pagó la cuenta (anonimato conservado de Criptografo2).

La tercer traza asume que el resultado de los volados lanzados por Criptógrafo1 y Criptógrafo2 fueron iguales (rVolado=0 (1) y rVolado=0 (1) respectivamente) y que ninguno de ellos decidió pagar (pague=0) anónimamente la cuenta. Esto implica comprobar que la NSA (Maestro) pagó la cuenta, es decir, que la siguiente ecuación de transición dada en un proceso con las condiciones asumidas se cumple:

$$\text{T(Calcula\_Resultados)}^{pre_4} \rightarrow^{pos_{4B}} \text{T(Anuncia\_Resultado)}$$

En la Ecuación 7.3.7 se realiza una transición del proceso T(Criptógrafo1) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 son verdaderas. De forma concurrente, en la Ecuación 7.3.19 se realiza una transición del proceso T(Criptógrafo2) al proceso T(Volado) debido a que la precondición pre_1 y la post-condición pos_1 también son verdaderas:

$$\text{T(Criptógrafo1)}^{pre_1} \rightarrow^{pos_1} \text{T(Volado)}$$

$$\text{T(Criptógrafo2)}^{pre_1} \rightarrow^{pos_1} \text{T(Volado)}$$

En la Ecuación 7.3.8 y 7.3.20 se realiza una transición del proceso T(Volado) al proceso T(Compartir) debido a que la precondición pre_2 resulta verdadera cuando se verifica que el valor de $rVolado$ es igual con cero (o uno) tanto en Criptógrafo1 como en Criptógrafo2, este valor es el asumido como hipótesis (rVolado=0 (1) y rVolado=0 (1)). Para esta traza se asume que los dos criptógrafos obtuvieron el mismo resultado en el volado, generándose la transición correspondiente con la post-condición pos_2 siendo verdadera:

$$\text{T(Volado)}^{pre_2} \rightarrow^{pos_2} \text{T(Compartir)}$$

En la Ecuación 7.3.9 y 7.3.21 se realiza una transición del proceso T(Compartir) al proceso T(Cript1Cript2_Medio) (T(Cript2Cript1_Medio)) debido a que la precondición pre_3 resulta verdadera cuando se verifica que el valor de $msjCript1$ ($msjCript2$) es igual con nulo, entonces se genera la transición correspondiente con la post-condición pos_3 siendo verdadera. En el proceso T(Cript1Cript2_Medio) (T(Crip2Cript1_Medio)) se envía el mensaje $Ms1$ al criptógrafo vecino de Criptógrafo1 (Criptógrafo2), conteniendo el resultado del volado lanzado $rVolado$ por Criptógrafo1 (Criptógrafo2).

$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript1Cript2\_Medio))}$

$\text{T(Compartir)}\Delta^{pre_3,pos_3,Ms1}\text{(T(Cript2Cript1\_Medio))}$

En la Ecuación 7.3.10 y 7.3.22 se realiza una transición del proceso T(Cript1Cript2_Medio) (T(Cript2Cript1_Medio)) al proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) debido a que la precondición pre_4 resulta verdadera, entonces se genera la transición correspondiente con la post-condición pos_4 siendo verdadera.

$\text{T(Cript1Cript2\_Medio)}^{pre_4}\!\!\rightarrow^{pos_4}\text{T(Cript2Cript1\_Medio)}$

$\text{T(Cript2Cript1\_Medio)}^{pre_4}\!\!\rightarrow^{pos_4}\text{T(Cript1Cript2\_Medio)}$

En la Ecuación 7.3.11 y 7.3.23 se realiza una transición del proceso T(Cript2Cript1_Medio) (T(Cript1Cript2_Medio)) al proceso T(Compartir) debido a que la precondición pre_5 resulta verdadera, entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera. En el proceso T(Cript2Cript1_Medio) (T(Crip1Cript2_Medio)) se recibe el mensaje $Me1$ del criptógrafo vecino Criptógrafo2 (Criptógrafo1), conteniendo el resultado del volado lanzado *vVolado* por Criptógrafo2 (Criptógrafo1).

$[\text{T(Cript2Cript1\_Medio)/-}]_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))}$

$[\text{T(Cript1Cript2\_Medio)/-}]_{Me1}\Delta^{pre_5,pos_5}\text{(T(Compartir))}$

En la Ecuación 7.3.12 y 7.3.24 se realiza una transición del proceso T(Compartir) al proceso T(Decisión_Pago) a través de dos posibles caminos, de acuerdo al valor de la precondición pre_6 que resulte verdadera (para la hipótesis se asume rVolado==vVolado); entonces se activa la expresión a la izquierda del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{6A} (anuncio=0) siendo verdadera.

$\text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6A}}\text{T(Decisión\_Pago)} + \text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6B}}\text{T(Decisión\_Pago)} =$

$\text{T(Compartir)}^{pre_6}\!\!\rightarrow^{pos_{6A}}\text{T(Decisión\_Pago)}$

En la Ecuación 7.3.13 y 7.3.25 se realiza una transición del proceso T(Decisión_Pago) al proceso T(Transmisión) o al proceso T(Cambiar_Resultado), de acuerdo al valor de la precondición pre_7 que resulte verdadera (para la hipótesis se asume pague==0); entonces se activa la expresión a la izquierda del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{7A} (pague=0) siendo verdadera.

$\text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7A}}\text{T(Transmisión)} +$

$\text{T(Decisión\_Pago)}^{pre_7}\!\!\rightarrow^{pos_{7B}}\text{T(Cambiar\_Resultado)} =$

$T(\text{Decisión\_Pago})^{pre_7} \rightarrow^{pos_{7A}} T(\text{Transmisión})$

En la Ecuación 7.3.15 y 7.3.27 se realiza una transición del proceso T(Transmisión) al proceso T(Maestro_Medio) o al proceso T(TxSalida_Medio) o al proceso T(Fin), de acuerdo al valor de la precondición pre_8 que resulte verdadera (para la traza msjCript1==null (msjCript2==null)); entonces se activa la expresión a la izquierda del primer operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{9A} (msjCript1.vDato=anuncio=0 (msjCript2.vDato=anuncio=0)) siendo verdadera. En el proceso T(Maestro_Medio) se envía el mensaje $Ms2$ de Criptógrafo1 (Criptógrafo2), conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$T(\text{Transmisión})\Delta^{pre_9, pos_{9A}, Ms2}(T(\text{Maestro\_Medio})) +$

$T(\text{Transmisión})\Delta^{pre_9, pos_{9B}, Ms3}(T(\text{TxSalida\_Medio})) +$

$T(\text{Transmisión})^{pre_9} \rightarrow^{pos_{9C}} T(\text{Fin}) =$

$T(\text{Transmisión})\Delta^{pre_9, pos_{9A}, Ms2}(T(\text{Maestro\_Medio}))$

Mientras tanto, en el trabajo LeGESD Maestro que concurrentemente se ejecuta con Criptógrafo1 y Criptógrafo2, en la Ecuación 7.3.33 se realiza una transición del proceso T(Maestro_Medio) al proceso T(Resultados) de acuerdo al valor de la precondición pre_3 que resulte verdadera (para la traza msjCript1!=null (msjCript2!=null)); entonces se genera la transición correspondiente con la post-condición pos_3 (recibido=msjCript1.vDato=0 (recibido=msjCript2.vDato=0), mensajes=mensajes+1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje Me tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$[T(\text{Maestro\_Medio})/\text{-}]_{Me}\Delta^{pre_3, pos_3}(T(\text{Resultados}))$

En la Ecuación 7.3.32 se realiza una transición del proceso T(Resultados) al proceso T(Maestro_Medio) o al proceso T(Calcula_Resultados), de acuerdo al valor de la precondición pre_2 que resulte verdadera (para la traza recibido!=-1); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{2B} (recibido!=-1) siendo verdadera. En el proceso T(Maestro_Medio) se recibe el mensaje Me tanto del Criptógrafo1 como del Criptógrafo2, conteniendo el resultado a anunciar *anuncio* de los volados lanzados por Criptógrafo1 y Criptógrafo2.

$T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2A}} T(\text{Maestro\_Medio}) + T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2B}} T(\text{Calcula\_Resultado}) =$

$T(\text{Resultados})^{pre_2} \rightarrow^{pos_{2B}} T(\text{Calcula\_Resultado})$

En la Ecuación 7.3.34 se realiza una transición del proceso T(Calcula_Resultado) al proceso T(Maestro_Medio) o al proceso T(Anuncia_Resultados), de acuerdo al valor de la precondición

pre_4 que resulte verdadera (para la traza mensajes==NCriptógrafos); entonces se activa la expresión a la derecha del operador + de la ecuación de transición, generándose la transición correspondiente con la post-condición pos_{4B} (mensajes=NCriptógrafos) siendo verdadera.

$$\text{T(Calcula\_Resultado)} \overset{pre_4}{\rightarrow} \overset{pos_{4A}}{} \text{T(Maestro\_Medio) +}$$

$$\text{T(Calcula\_Resultados)} \overset{pre_4}{\rightarrow} \overset{pos_{4B}}{} \text{T(Anuncia\_Resultado) =}$$

$$\text{T(Calcula\_Resultados)} \overset{pre_4}{\rightarrow} \overset{pos_{4B}}{} \text{T(Anuncia\_Resultado)}$$

En la Ecuación 7.3.35 se realiza una transición del proceso T(Anuncia_Resultado) al proceso T(Fin) de acuerdo al valor de la precondición pre_5 que resulte verdadera (para la traza pagado_por=='N'); entonces se genera la transición correspondiente con la post-condición pos_5 siendo verdadera.

$$\text{T(Anuncia\_Resultado)} \overset{pre_5}{\rightarrow} \overset{pos_5}{} \text{T(Fin)}$$

Como resultado final de la tercera traza, se observa que tanto Criptógrafo1 como Criptógrafo2 anunciaron cada uno como resultado (anuncio) un valor de cero, después de haber compartido sus respectivos volados. Lo anterior fue debido a que ninguno de ellos decidió pagar la cuenta, siendo el trabajo LeGESD Maestro (NSA) quien pagó. El trabajo LeGESD Maestro funge como anunciante del pago de la cuenta, el cual recibe los anuncios públicos de los criptógrafos y calcula las diferencias, resultando un número par. Por lo que con la información pública que cada criptógrafo anunció, el Maestro conoce y anuncia que él ha sido quien pagó la cuenta (anonimato conservado debido a que ninguno de lo criptógrafos pagó).

$\Rightarrow$ Dada la información pública que los criptógrafos comparten, mediante la tercer traza que se realizó del sistema sólo es posible conocer que ningún criptógrafo pagó la cuenta (anonimato conservado), habiéndola pagado el Maestro.

La comprobación está completa.

$\rightarrow$ La propiedad ha sido comprobada.

De acuerdo al análisis de resultados, se han verificado las propiedades de seguridad tanto del sistema productor-consumidor como del sistema cena de criptógrafos especificados con LeGESD, a partir de su especificación realizada con ADSD. La verificación de las propiedades de seguridad realizada en el dominio algebraico de ADSD, puede extenderse al dominio gráfico de LeGESD debido a la herramienta matemática desarrollada en la presente tesis, la cual establece que la especificación gráfica es equivalente a la especificación algebraica, permitiendo la verificación de la especificación gráfica a partir de la especificación algebraica.

A lo largo de este trabajo se ha presentado el lenguaje LeGESD, desarrollando tanto su sintaxis en el capítulo 5 como su semántica en el capítulo 6. Además se han especificado dos sistemas completos utilizando LeGESD, verificando sus propiedades de seguridad a través del álgebra de procesos ADSD del lenguaje. Basándose en los resultados obtenidos, y recordando la hipótesis de

trabajo planteada, se puede decir que la hipótesis ha quedado comprobada, pudiéndose afirmar que:

En efecto, es posible definir un lenguaje gráfico para la especificación de sistemas distribuidos que incorpore como su semántica formal un álgebra de procesos, de tal manera que la especificación gráfica de un sistema distribuido sea equivalente a su especificación algebraica permitiendo la verificación de propiedades de seguridad de la especificación gráfica a partir de su especificación algebraica.

Con esto concluye la presentación del Lenguaje Gráfico para la Especificación de Sistema Distribuidos nombrado LeGESD, el cual ha sido el tema de la presente tesis. En el siguiente capítulo, se muestra la estructura de una herramienta computacional desarrollada como parte del trabajo de tesis que implementa a LeGESD.

8 Herramienta computacional de LeGESD

8.1. Introducción

Dentro del trabajo realizado en la presente tesis, se ha desarrollado una herramienta computacional a través de la cual es posible realizar la especificación de sistemas distribuidos utilizando a LeGESD. La herramienta está programada en el lenguaje de programación C#, y en este capítulo se describe de manera general la estructura de la misma.

8.2. Elementos de la herramienta computacional

La herramienta computacional de LeGESD cuenta con los siguientes elementos como parte de su interfaz gráfica de usuario:

a) Elementos del menú.

b) Símbolos gráficos para los enlaces LeGESD.

c) Símbolos gráficos para los estados LeGESD.

d) Ventana de proyecto.

e) Ventana de diagramación.

En la Figura 8.1 se muestran cada uno de los elementos anteriores. Con un recuadro rojo se señalan los elementos del menú (a)), con un recuadro azul los símbolos gráficos para los enlaces LeGESD (b)), con un recuadro morado los símbolos gráficos para los estados LeGESD (c)), con un recuadro verde la ventana de proyecto (d)), y con un recuadro naranja la ventana de diagramación (e)).

8.2.1. Elementos del menú

Los elementos del menú se pueden observar en la Figura 8.1 dentro del recuadro rojo (a)), y son los siguientes:

- Archivo: Permite manipular los archivos generados con la herramienta computacional.
- Proyecto: Permite agregar nuevos diagramas dentro de un proyecto LeGESD.
- Diagrama: Permite manipular los diagramas que se crean dentro de un proyecto LeGESD.
- Ayuda: Permite consultar una breve ayuda sobre el uso de la herramienta computacional.

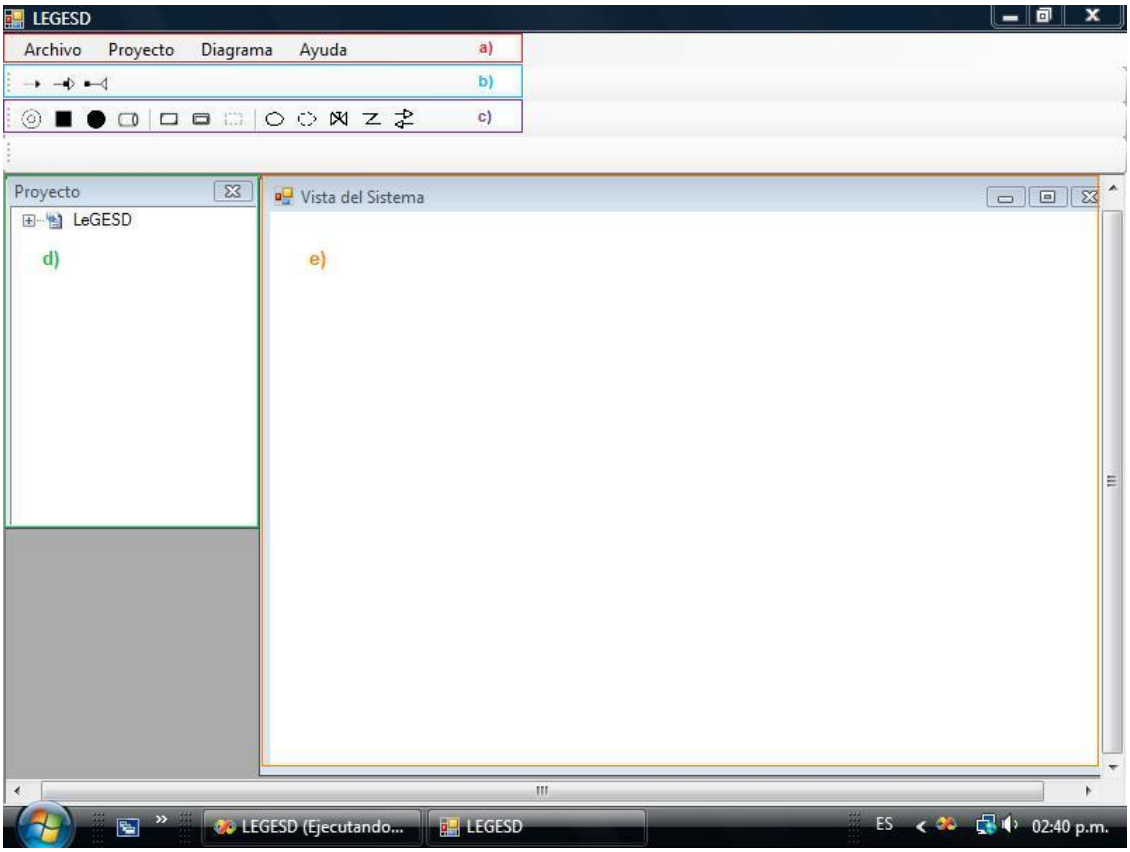


Figura 8.1: Elementos de la herramienta computacional.

Los elementos que integran al submenú Archivo pueden observarse en la Figura 8.2, y a continuación se describen:

Figura 8.2: Elementos del submenú Archivo.

- Abrir: Permite abrir un proyecto creado en la herramienta computacional de LeGESD, como puede observarse en la Figura 8.3.
- Nuevo: Permite crear un nuevo proyecto LeGESD; para crear el nuevo proyecto se escribe el nombre del proyecto en el componente respectivo de la interfaz de usuario, y finalmente se agrega el nuevo proyecto tanto a la Ventana de proyecto como de diagramación. Lo descrito anteriormente puede observarse en las Figuras 8.4, 8.5, 8.7 y 8.8.
- Guardar: Permite guardar un proyecto creado en la herramienta computacional de LeGESD bajo la extensión leg, como puede observarse en la Figura 8.6.

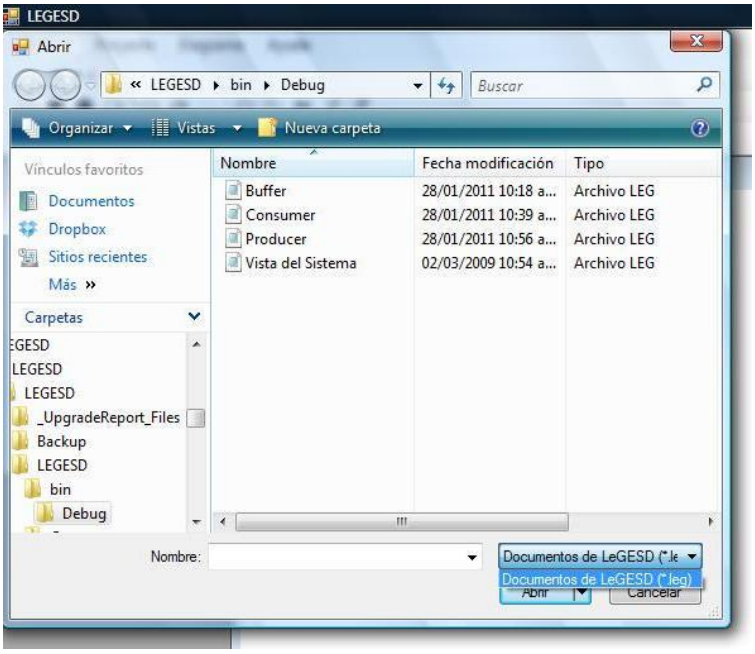


Figura 8.3: Opción Abrir del submenú Archivo.

Figura 8.4: Opción Nuevo del submenú Archivo.

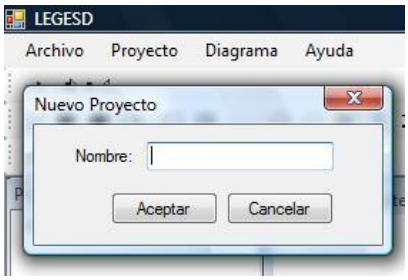


Figura 8.5: Nombre del nuevo proyecto.

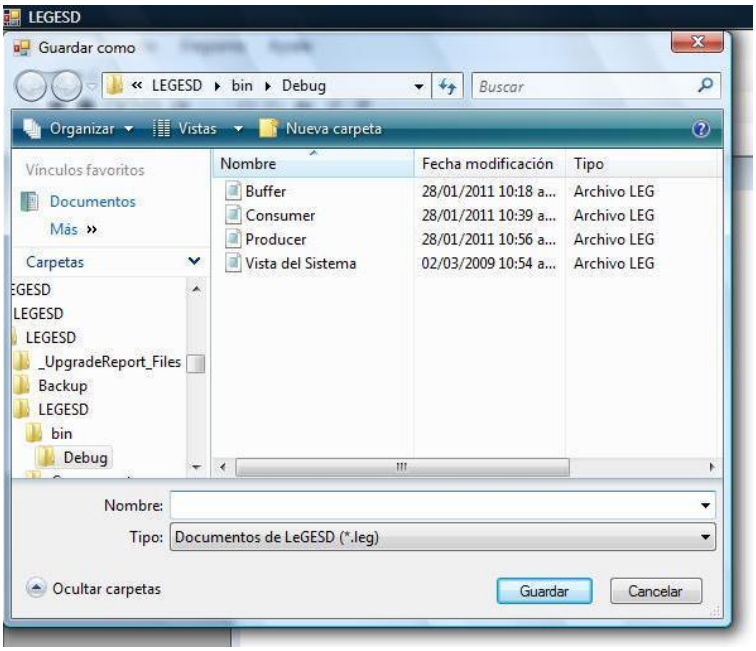


Figura 8.6: Opción Guardar del submenú Archivo.

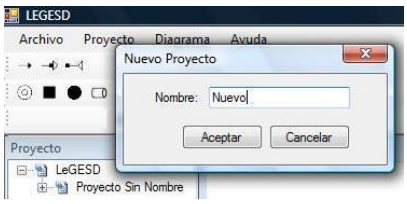


Figura 8.7: Captura del nombre del nuevo proyecto.

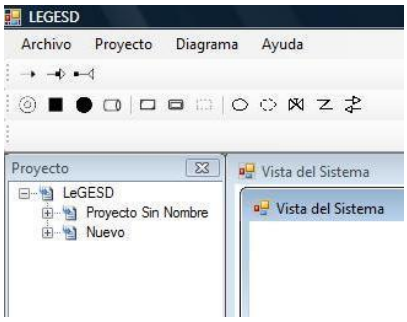


Figura 8.8: Nuevo proyecto agregado a la Ventana de proyecto y de diagramación.

El elemento que integra al submenú Proyecto puede observarse en la Figura 8.9, y a continuación se describe:

- Agregar: Permite agregar un nuevo diagrama al proyecto, como puede observarse en la Figura 8.10.

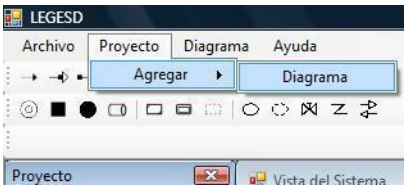


Figura 8.9: Elemento del submenú Proyecto.

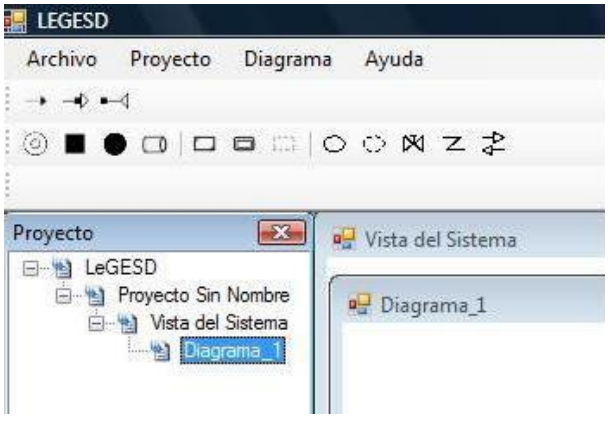


Figura 8.10: Nuevo diagrama agregado al proyecto.

Los elementos que integran al submenú Diagrama pueden observarse en la Figura 8.11, y a continuación se describen:

Figura 8.11: Elementos del submenú Diagrama.

- Ocultar: Permite ocultar temporalmente el diagrama activo y que no vaya a utilizarse durante el trabajo con otros diagramas, como puede observarse en la Figura 8.12 y 8.13.
- Ver: Permite visualizar el diagrama que se marque como activo en la Ventana de proyecto para que pueda utilizarse en ese momento, como puede observarse en la Figura 8.14 y 8.15.
- Borrar: Permite eliminar el diagrama que se marque como activo en la Ventana de proyecto, como puede observarse en la Figura 8.16 y 8.17.

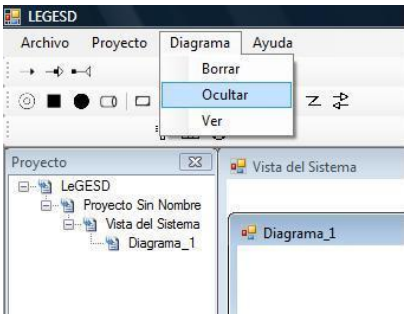


Figura 8.12: Opción Ocultar del submenú Diagrama.

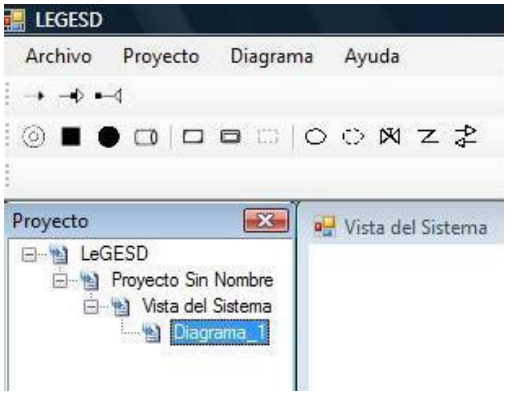


Figura 8.13: Ocultamiento del diagrama activo Diagrama_1.

Figura 8.14: Opción Ver del submenú Diagrama.

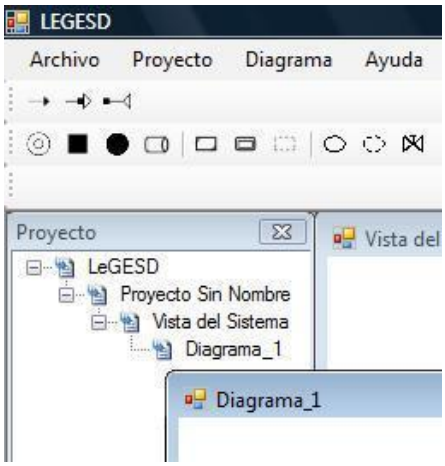


Figura 8.15: Visualización del diagrama activo Diagrama_1.

Figura 8.16: Opción Borrar del submenú Diagrama.

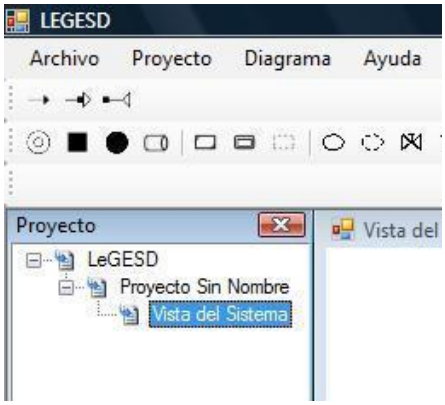


Figura 8.17: Eliminación del diagrama activo Diagrama_1.

El elemento que integra al submenú Ayuda puede observarse en la Figura 8.18, y a continuación se describe:

- Ayuda de LeGESD: Permite consultar una breve ayuda sobre el uso de la herramienta computacional, como puede observarse en la Figura 8.19.

8.2.2. Símbolos gráficos para los enlaces LeGESD

Los símbolos gráficos para los enlaces LeGESD se pueden observar en la Figura 8.1 dentro del recuadro azul (b)), y son los siguientes:

- Enlace simple: Permite unir dos estados LeGESD dando clic en el icono y seleccionando los estados que unirá en la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el enlace es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar la precondición y la post condición asociadas a dicho enlace. Las Figuras 8.20, 8.21, 8.22 y 8.23 muestran lo descrito anteriormente.

- Enlace mensaje de entrada: Permite unir un estado Transición con un estado Punto de acceso para declarar mensajes de entrada. Dando clic con el botón derecho del ratón sobre el enlace es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar los mensajes de entrada asociados a dicho enlace. Las Figuras 8.24, 8.25, 8.26 y 8.27 muestran lo descrito anteriormente.

- Enlace mensaje de salida: Permite unir un estado Transición con un estado Punto de acceso para declarar mensajes de salida. Dando clic con el botón derecho del ratón sobre el enlace es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar los mensajes de salida asociados a dicho enlace. Las Figuras 8.28, 8.29, 8.30 y 8.31 muestran lo descrito anteriormente.

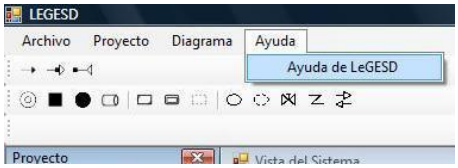


Figura 8.18: Elemento del submenú Ayuda.

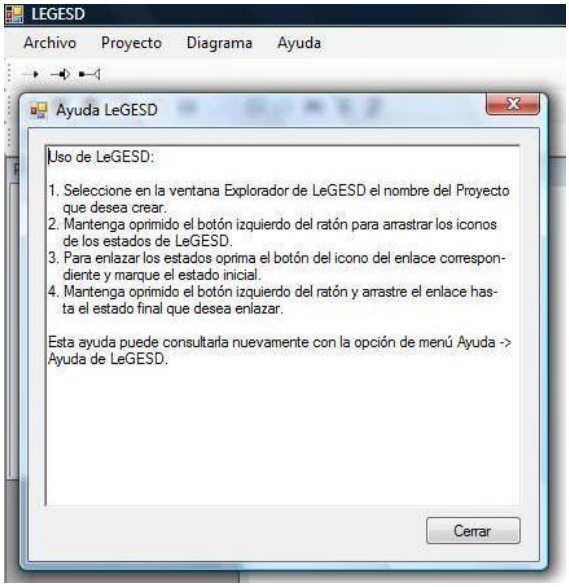


Figura 8.19: Ayuda breve sobre el uso de la herramienta computacional.

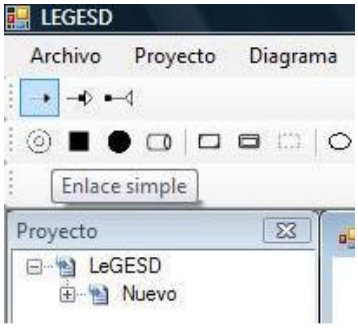


Figura 8.20: Símbolo para un Enlace simple.

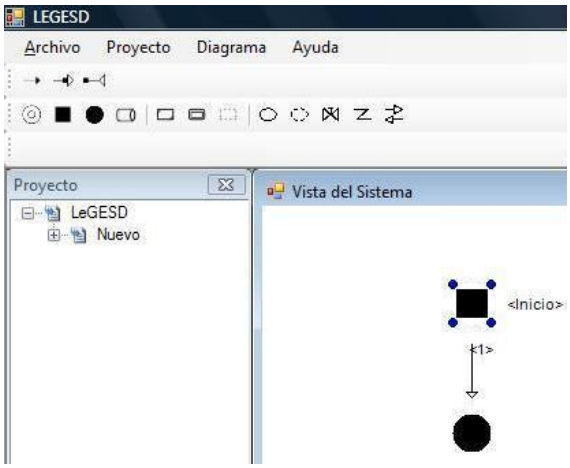


Figura 8.21: Enlace simple uniendo dos estados LeGESD en la Ventana de diagramación.

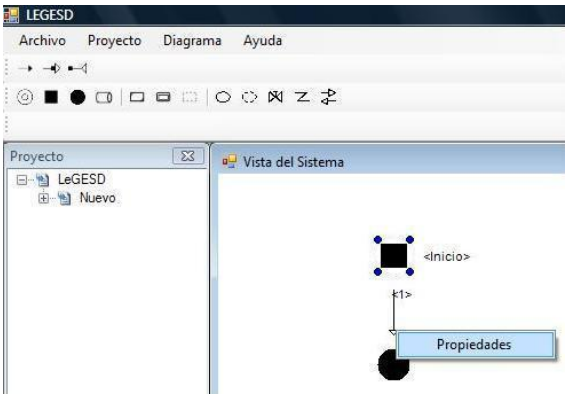


Figura 8.22: Propiedades del Enlace simple.

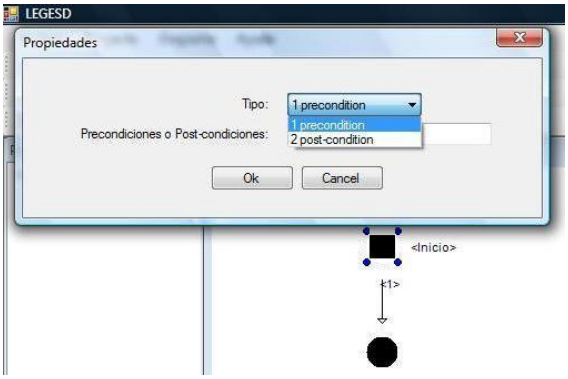


Figura 8.23: Precondición y post condición del Enlace simple.

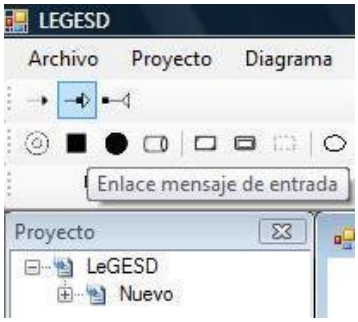


Figura 8.24: Símbolo para un Enlace mensaje de entrada.

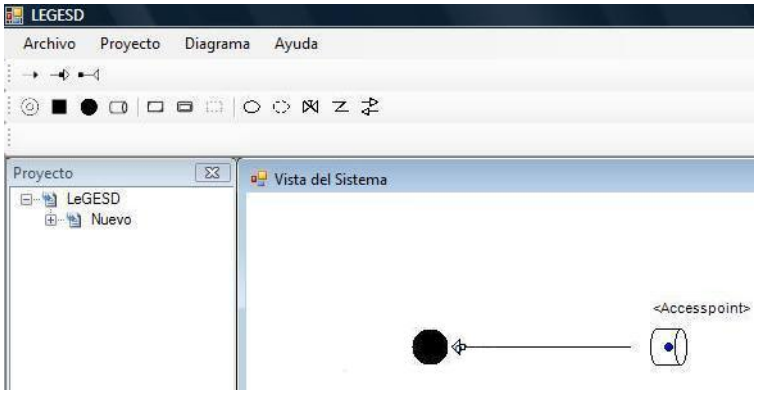


Figura 8.25: Enlace mensaje de entrada uniendo un estado Transición con un estado Punto de acceso en la Ventana de diagramación.

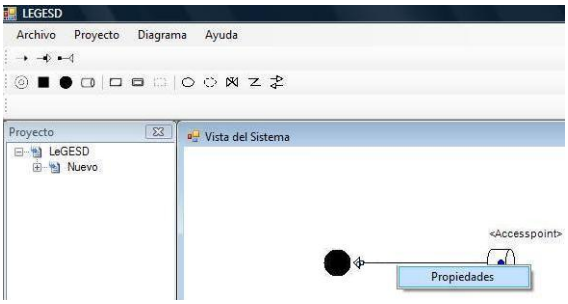


Figura 8.26: Propiedades del Enlace mensaje de entrada.

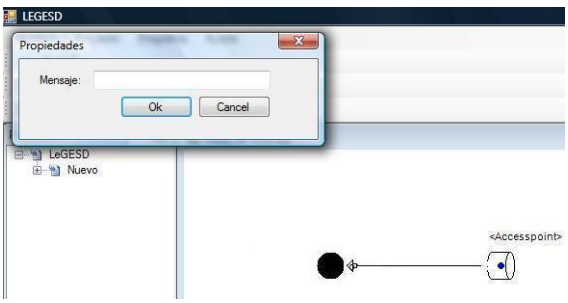


Figura 8.27: Mensajes a declarar como entrada del Enlace mensaje de entrada.

Figura 8.28: Símbolo para un Enlace mensaje de salida.

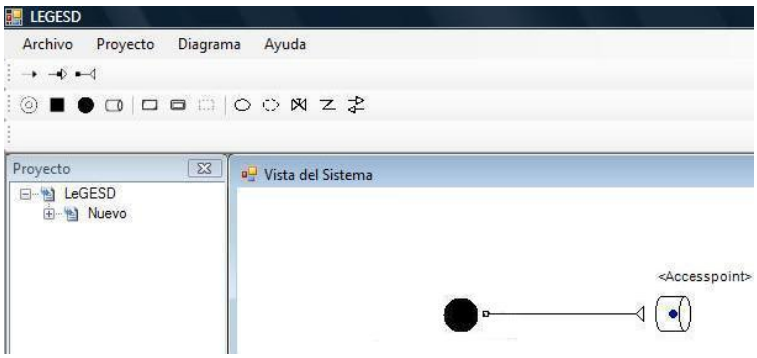


Figura 8.29: Enlace mensaje de salida uniendo un estado Transición con un estado Punto de acceso en la Ventana de diagramación.

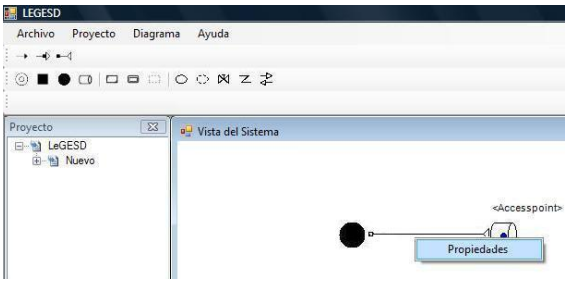


Figura 8.30: Propiedades del Enlace mensaje de salida.

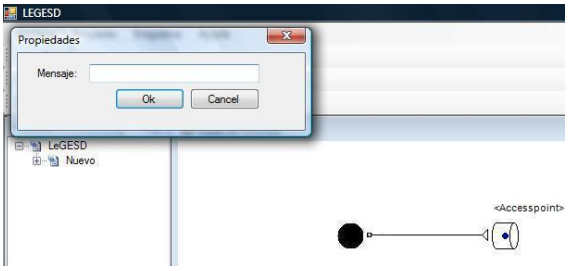


Figura 8.31: Mensajes a declarar como salida del Enlace mensaje de salida.

8.2.3. Símbolos gráficos para los estados LeGESD

Los símbolos gráficos para los estados LeGESD se pueden observar en la Figura 8.1 dentro del recuadro morado (c)), y son los siguientes:

- Inicio: Permite dibujar un estado Inicio dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.32, 8.33 y 8.34 muestran lo descrito anteriormente.
- Fin: Permite dibujar un estado Fin dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.35, 8.36 y 8.37 muestran lo descrito anteriormente.
- Transición: Permite dibujar un estado Transición dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Este estado no tiene propiedades asociadas como se muestra en la Figura 8.38.
- Punto de acceso: Permite dibujar un estado Punto de acceso dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el

estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado y los mensajes a intercambiar. Las Figuras 8.39, 8.40 y 8.41 muestran lo descrito anteriormente

Figura 8.32: Símbolo para un estado Inicio.

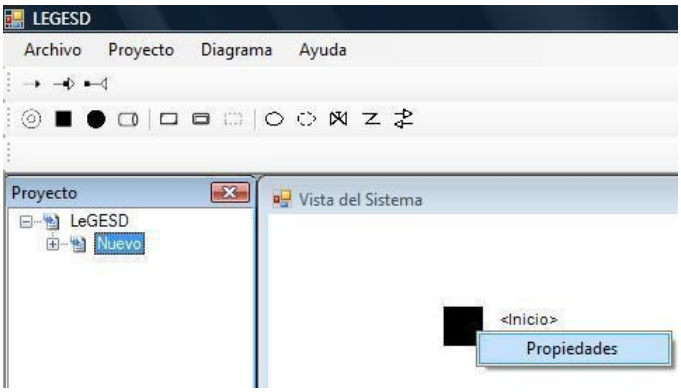


Figura 8.33: Propiedades del estado Inicio.

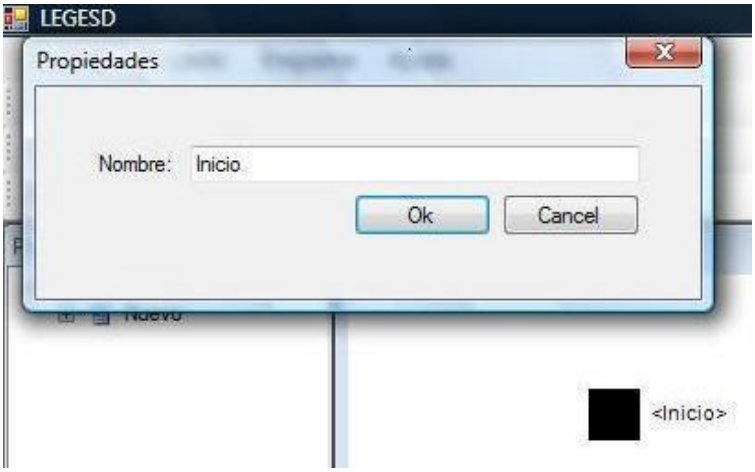


Figura 8.34: Nombre del estado Inicio.

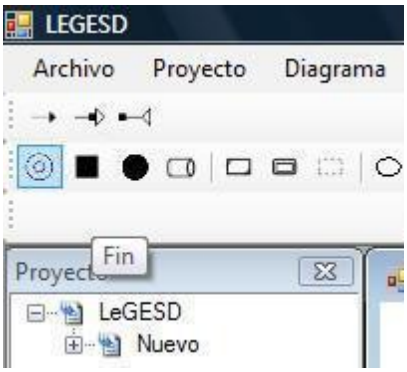


Figura 8.35: Símbolo para un estado Fin.

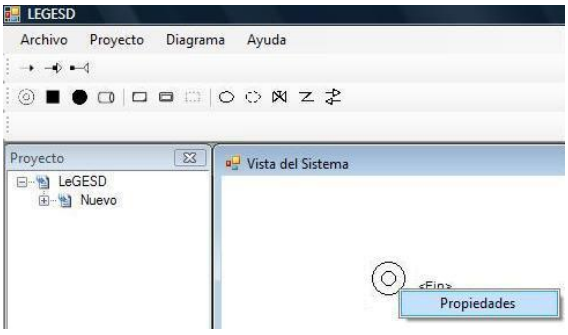


Figura 8.36: Propiedades del estado Fin.

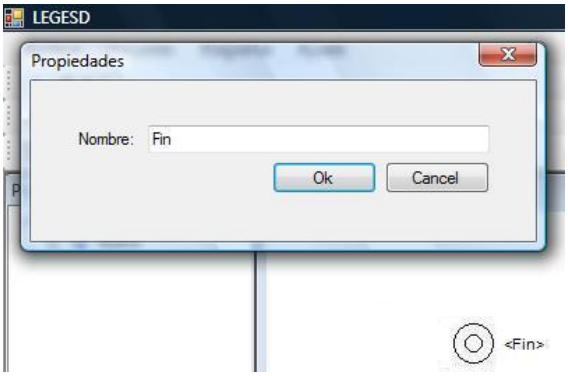


Figura 8.37: Nombre del estado Fin.

Figura 8.38: Símbolo para un estado Transición.

Figura 8.39: Símbolo para un estado Punto de acceso.

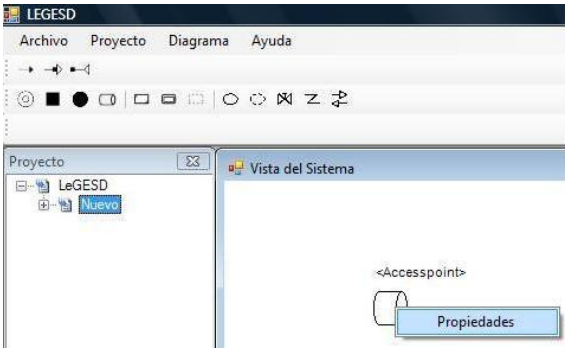


Figura 8.40: Propiedades del estado Punto de acceso.

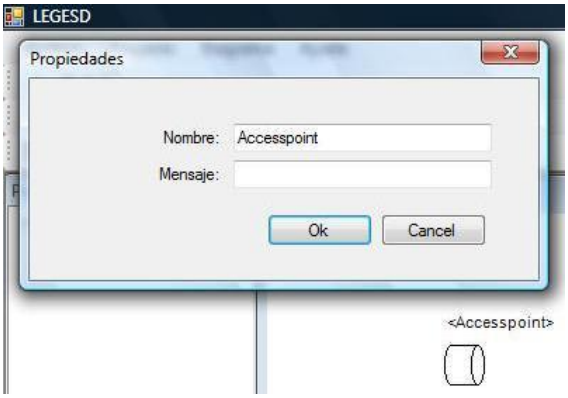


Figura 8.41: Nombre del estado Punto de acceso y los mensajes a intercambiar.

- Interno: Permite dibujar un estado Interno dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado y los datos del método asociado al estado. Las Figuras 8.42, 8.43 y 8.44 muestran lo descrito anteriormente.

- Comunicación: Permite dibujar un estado Comunicación dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.45, 8.46 y 8.47 muestran lo descrito anteriormente.

- Compuesto: Permite dibujar un estado Compuesto dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar

el nombre del estado, así como los eventos y mensajes asociados a dicho estado. Las Figuras 8.48, 8.49 y 8.50 muestran lo descrito anteriormente.

- Medio simple: Permite dibujar un estado Medio simple dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.51, 8.52 y 8.53 muestran lo descrito anteriormente.

Figura 8.42: Símbolo para un estado Interno.

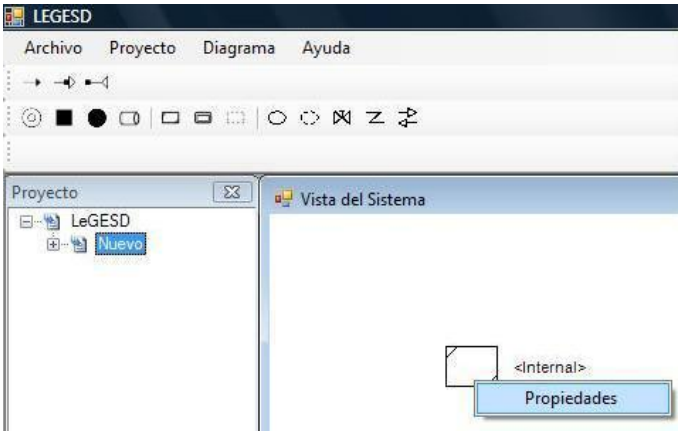


Figura 8.43: Propiedades del estado Interno.

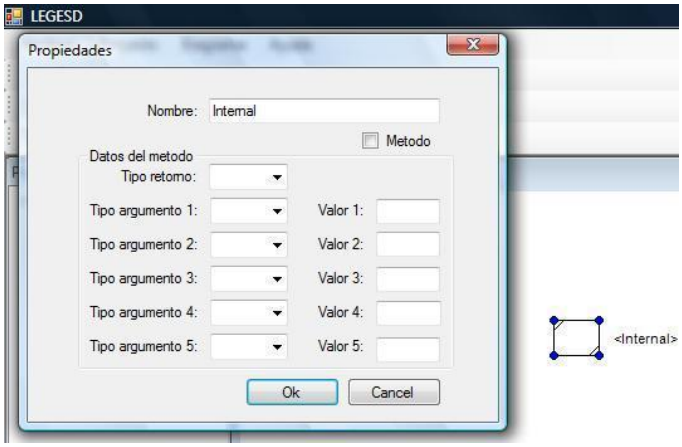


Figura 8.44: Nombre del estado Interno y datos del método.

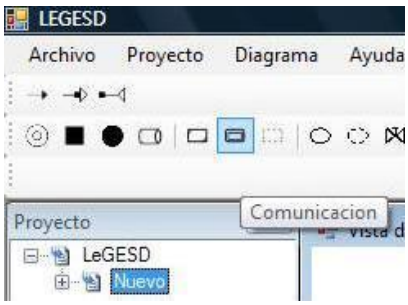


Figura 8.45: Símbolo para un estado Comunicación.

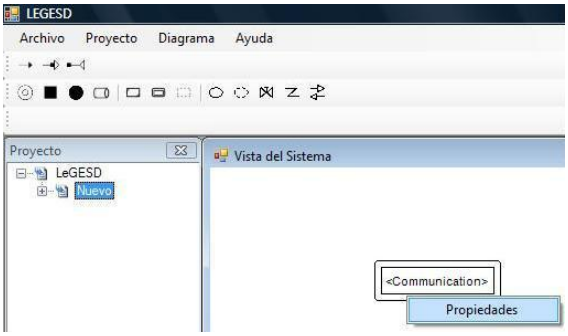


Figura 8.46: Propiedades del estado Comunicación.

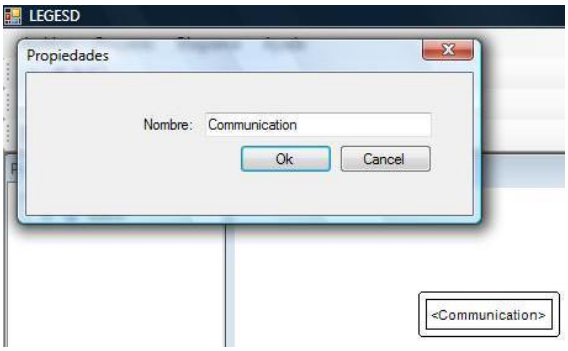


Figura 8.47: Nombre del estado Comunicación.

Figura 8.48: Símbolo para un estado Compuesto.

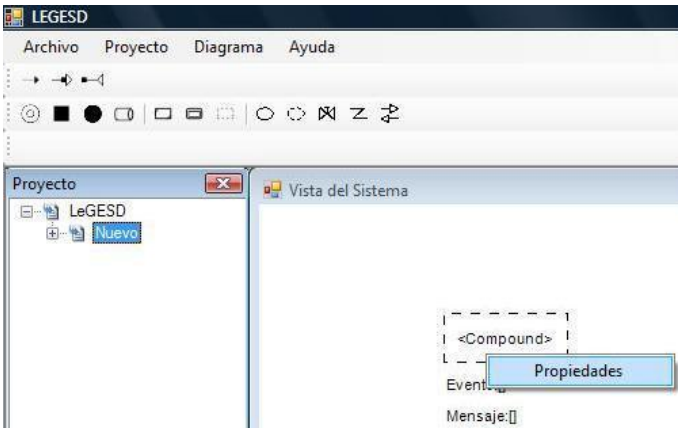


Figura 8.49: Propiedades del estado Compuesto.

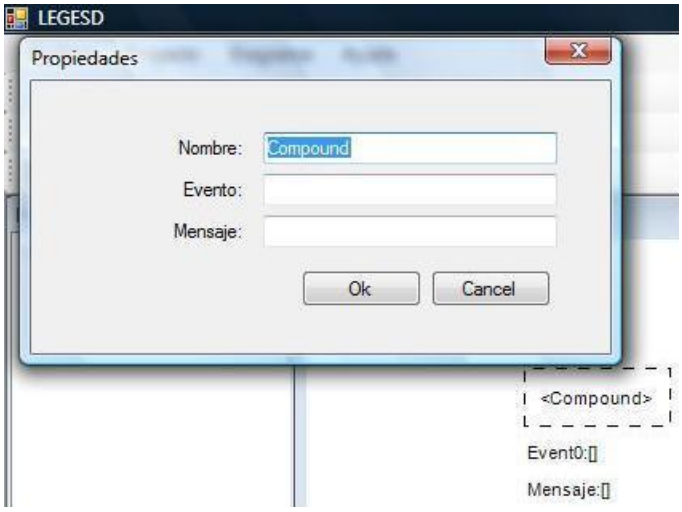


Figura 8.50: Nombre del estado Compuesto, eventos y mensajes asociados.

Figura 8.51: Símbolo para un estado Medio simple.

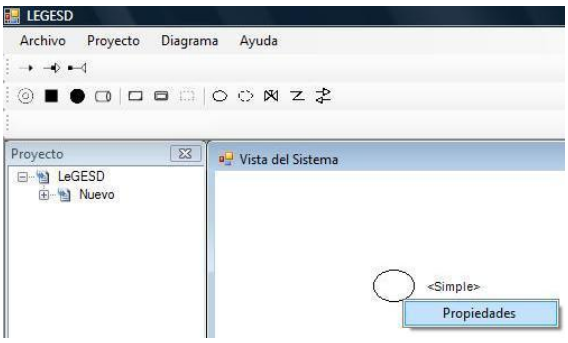


Figura 8.52: Propiedades del estado Medio simple.

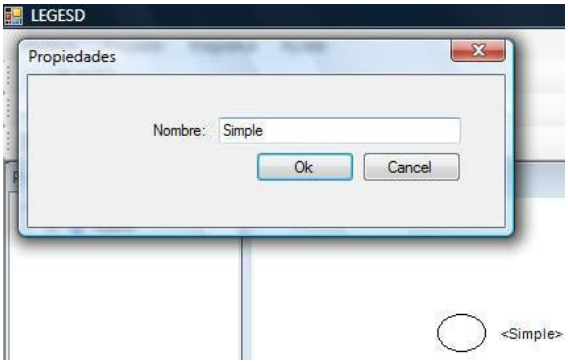


Figura 8.53: Nombre del estado Medio simple.

- Medio compuesto: Permite dibujar un estado Medio compuesto dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.54, 8.55 y 8.56 muestran lo descrito anteriormente.

- Apertura medio: Permite dibujar un estado Apertura dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.57, 8.58 y 8.59 muestran lo descrito anteriormente.

- Enlace medio: Permite dibujar un estado Atado dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.60, 8.61 y 8.62 muestran lo descrito anteriormente.

- Inicialización medio: Permite dibujar un estado Comienzo dando clic en el icono y arrastrándolo hasta la Ventana de diagramación. Dando clic con el botón derecho del ratón sobre el estado es posible acceder a las propiedades del mismo. Dentro de las propiedades es posible declarar el nombre del estado. Las Figuras 8.63, 8.64 y 8.65 muestran lo descrito anteriormente.

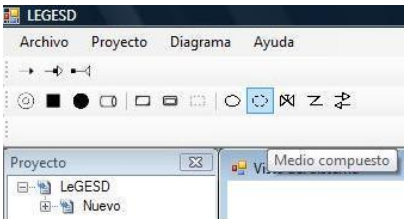


Figura 8.54: Símbolo para un estado Medio compuesto.

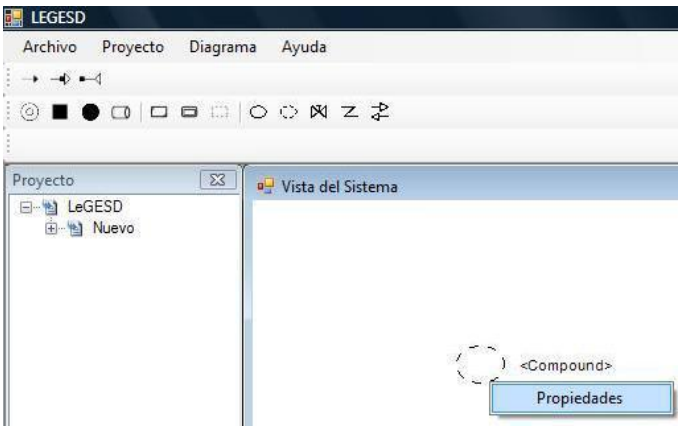


Figura 8.55: Propiedades del estado Medio compuesto.

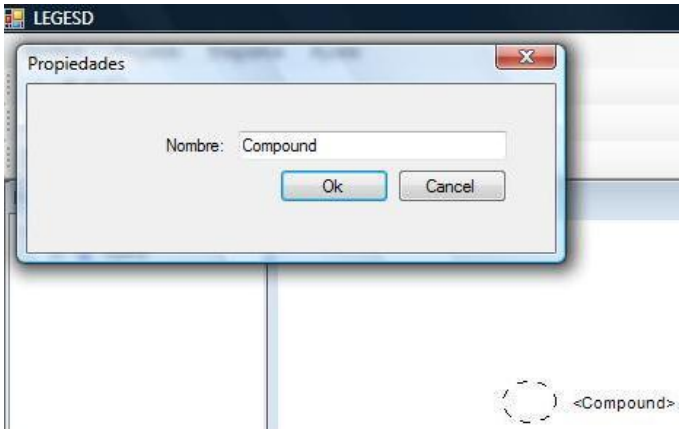


Figura 8.56: Nombre del estado Medio compuesto.

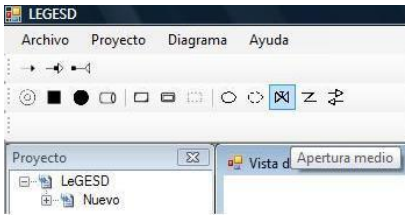


Figura 8.57: Símbolo para un estado Apertura.

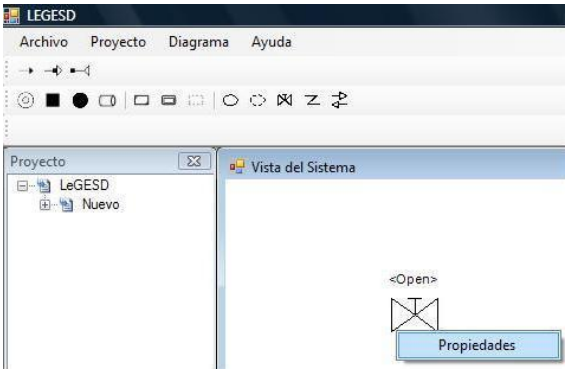


Figura 8.58: Propiedades del estado Apertura.

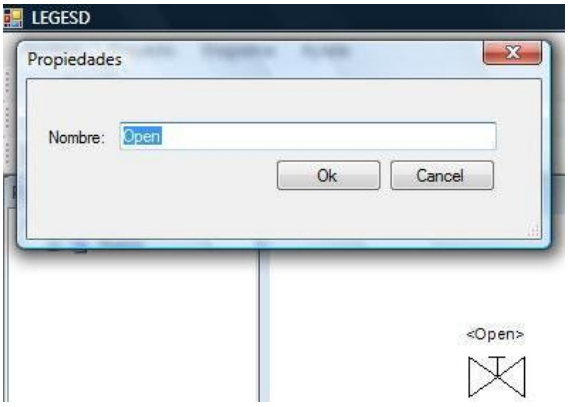


Figura 8.59: Nombre del estado Apertura.

Figura 8.60: Símbolo para un estado Atado.

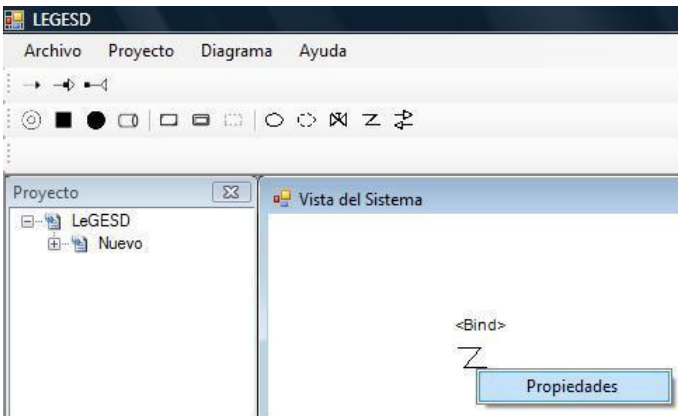


Figura 8.61: Propiedades del estado Atado.

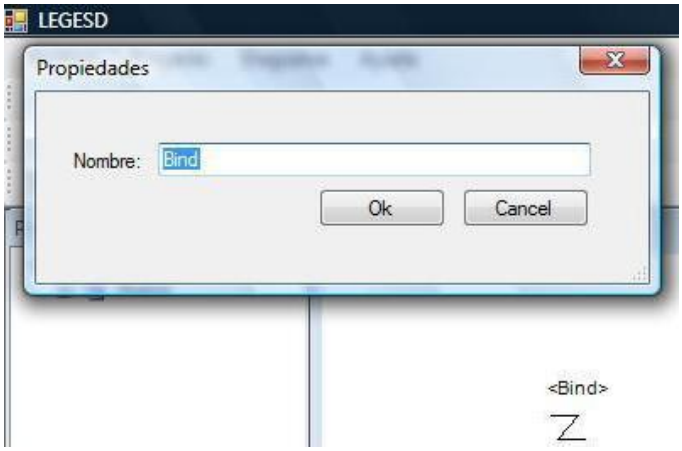


Figura 8.62: Nombre del estado Atado.

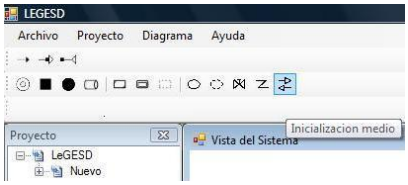


Figura 8.63: Símbolo para un estado Comienzo.

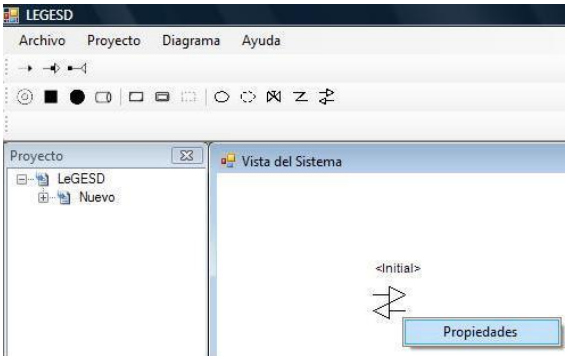


Figura 8.64: Propiedades del estado Comienzo.

Figura 8.65: Nombre del estado Comienzo.

8.2.4. Ventana de proyecto

La Ventana de proyecto se puede observar en la Figura 8.1 dentro del recuadro verde (d)). En esta ventana se muestran en forma de árbol los proyectos LeGESD que se estén desarrollando. Cada proyecto está conformado por la Vista de sistema y uno o más diagramas, todos ellos agrupados dentro de la Ventana de proyecto. Desde la ventana de proyecto es posible renombrar tanto al proyecto como a los distintos diagramas que lo integran; esto se realiza mediante un doble clic sobre el nombre del elemento que se desee cambiar el nombre. Lo descrito anteriormente se muestra en las Figuras 8.66, 8.67, 8.68, 8.69 y 8.70.

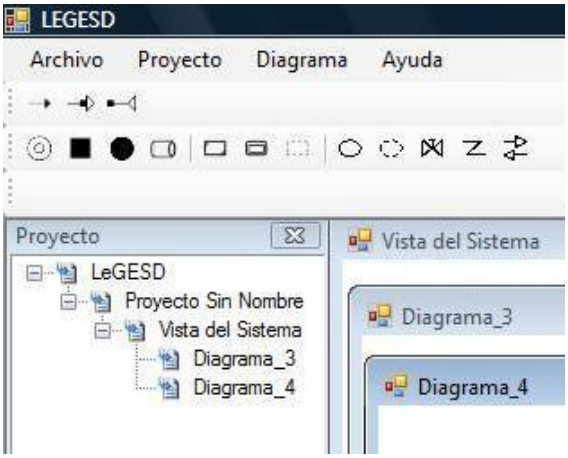


Figura 8.66: Elementos de un proyecto visualizados en la Ventana de proyecto.

Figura 8.67: Selección del proyecto para cambiar su nombre.

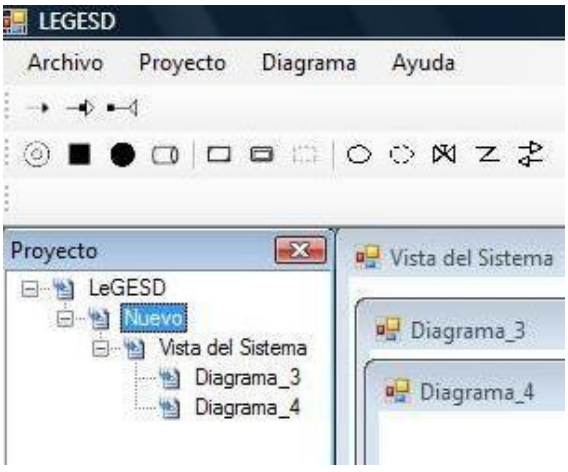


Figura 8.68: Nombre del proyecto cambiado.

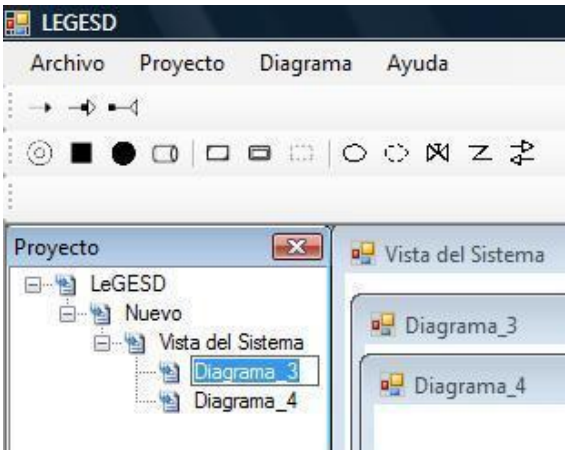


Figura 8.69: Selección del diagrama para cambiar su nombre.

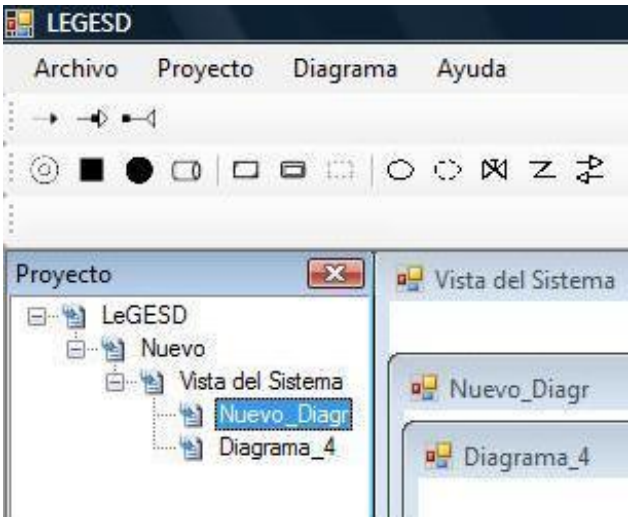


Figura 8.70: Nombre del diagrama cambiado.

8.3. Verificación de las reglas sintácticas de LeGESD utilizando la herramienta computacional

En esta sección se muestra el uso de la herramienta computacional de LeGESD para la verificación de algunas de las reglas sintácticas del lenguaje. Estas reglas fueron presentadas en el capítulo 5, en la sección 5.4.3. La figura 8.71 muestra la revisión, por parte de la herramienta, de la regla sintáctica 3 de LeGESD, la cual establece que: *dos estados cualesquiera, excepto el estado Transición, deben estar conectados a través de un estado Transición utilizando enlaces simples o de mensajes*.

En la figura 8.71 se observa el intento de conectar el estado llamado *Interno* con el estado llamado *Fin*, sin utilizar un estado Transición para la conexión. La figura 8.72 muestra el mensaje de error generado por la herramienta, cuando revisa que la regla sintáctica correspondiente no es válida.

La figura 8.73 muestra la revisión de la regla sintáctica 5 de LeGESD, la cual establece que: *la precondición está siempre declarada antes de un estado Transición, y la post-condición está siempre declarada después de un estado Transición*.

En la figura 8.73 se observa el intento de declarar una post-condición como parte de la conexión entre el estado llamado *Inicio* y el estado Transición. La figura 8.74 muestra el mensaje de error generado por la herramienta, cuando revisa que la regla sintáctica correspondiente no es válida, debido a que una post-condición no puede declararse antes que un estado Transición.

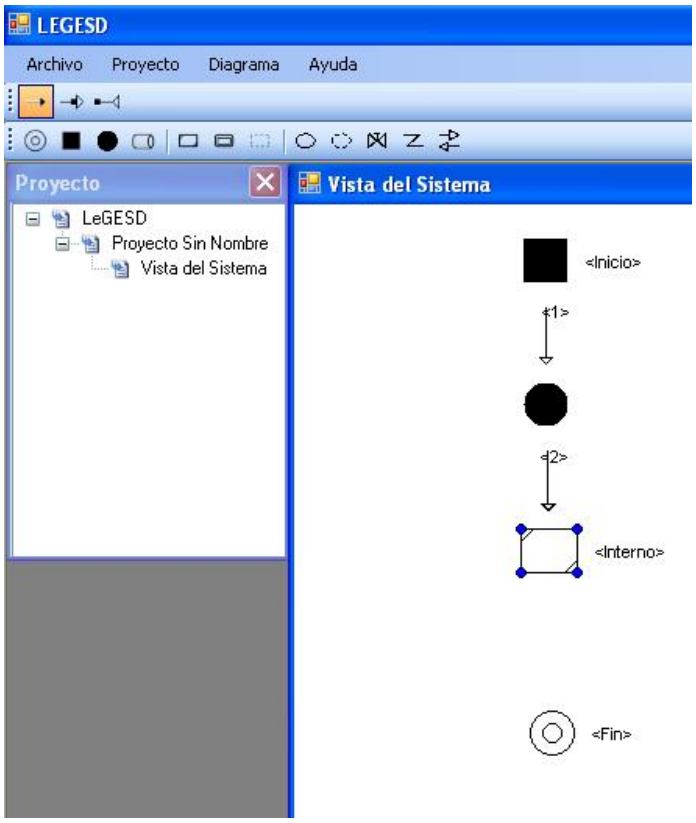


Figura 8.71: Verificación de la regla sintáctica 3 de LeGESD por la herramienta computacional.

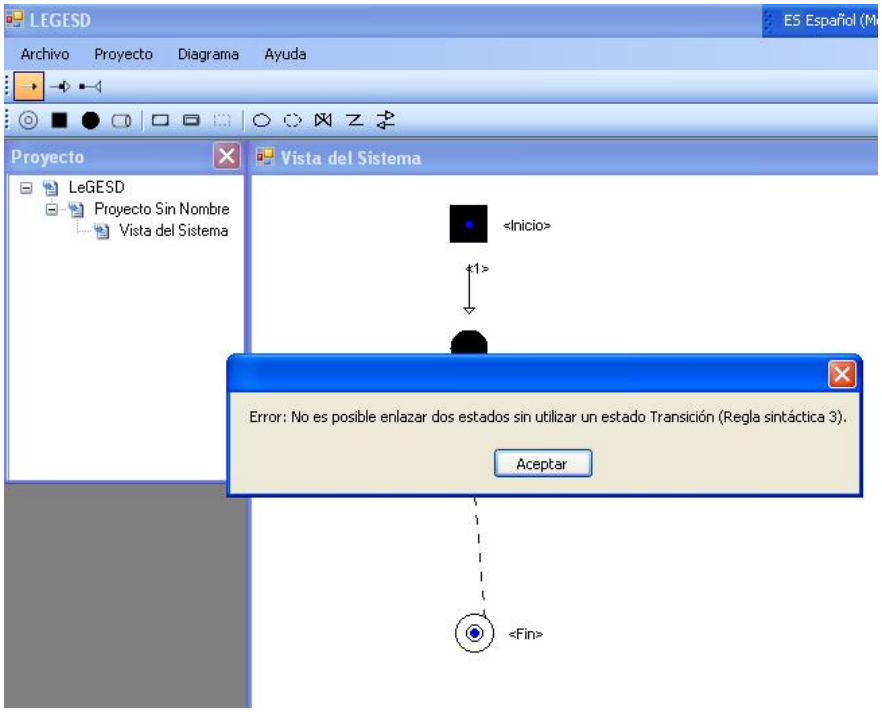


Figura 8.72: Error resultante de la verificación de la regla sintáctica 3.

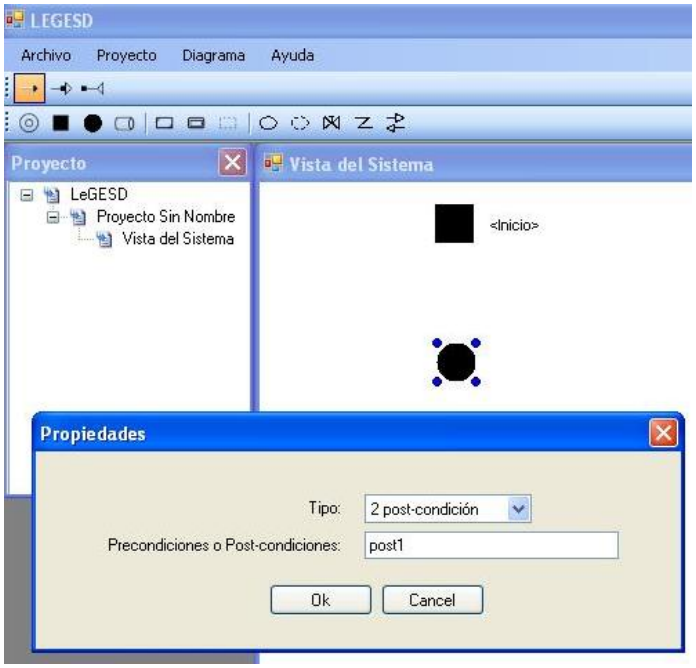


Figura 8.73: Verificación de la regla sintáctica 5 de LeGESD por la herramienta computacional para la precondición.

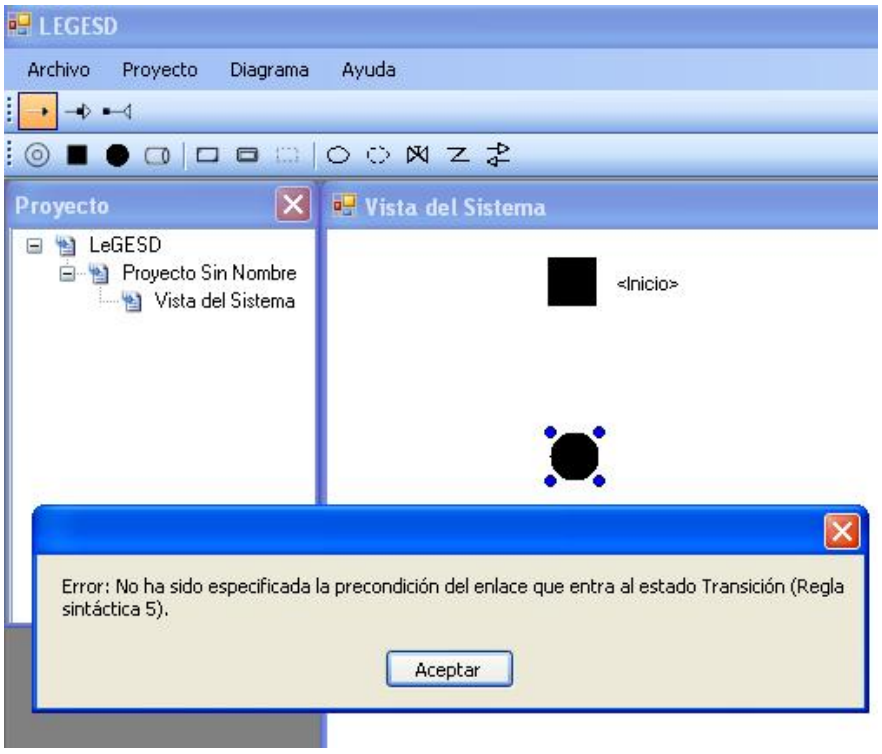


Figura 8.74: Error resultante de la verificación de la regla sintáctica 5 para la precondición.

De manera similar, la figura 8.75 y la figura 8.76 muestran el caso correspondiente para el

intento de declaración de una precondición después de un estado Transición.

La figura 8.77 muestra la revisión de la regla sintáctica 8 de LeGESD, la cual establece que: *un estado Punto de acceso y un estado Transición están conectados entre ellos con un enlace de mensaje etiquetado-entrada y/o etiquetado-salida, debiendo estar asociados a un estado Comunicación o Simple o Compuesto del medio*.

En la figura 8.77 se observa el intento de conectar un estado Transición con un estado Punto de acceso, haciendo uso de un enlace simple. La figura 8.78 muestra el mensaje de error generado por la herramienta, cuando revisa que la regla sintáctica correspondiente no es válida, debido a que la conexión entre un estado Transición y un estado Punto de acceso requiere de un enlace de mensaje.

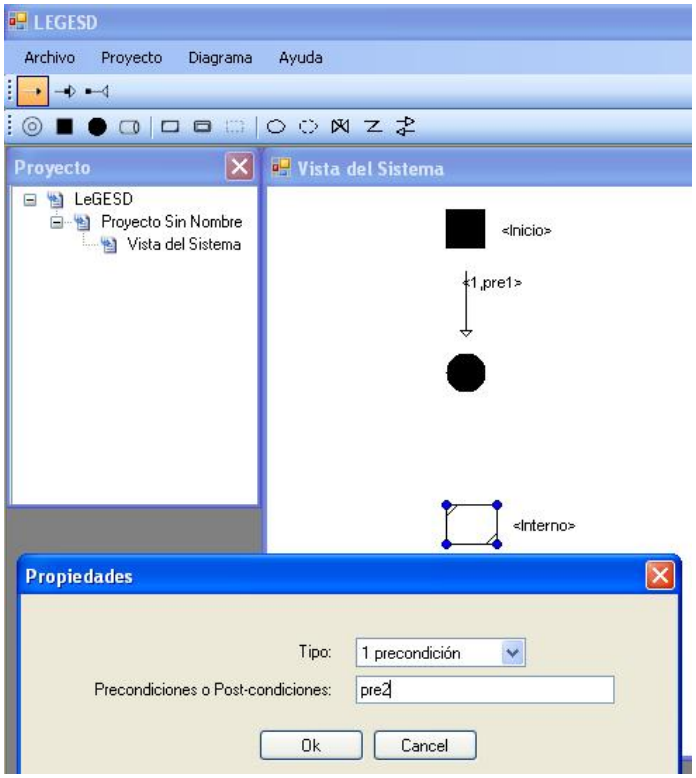


Figura 8.75: Verificación de la regla sintáctica 5 de LeGESD por la herramienta computacional para la post-condición.

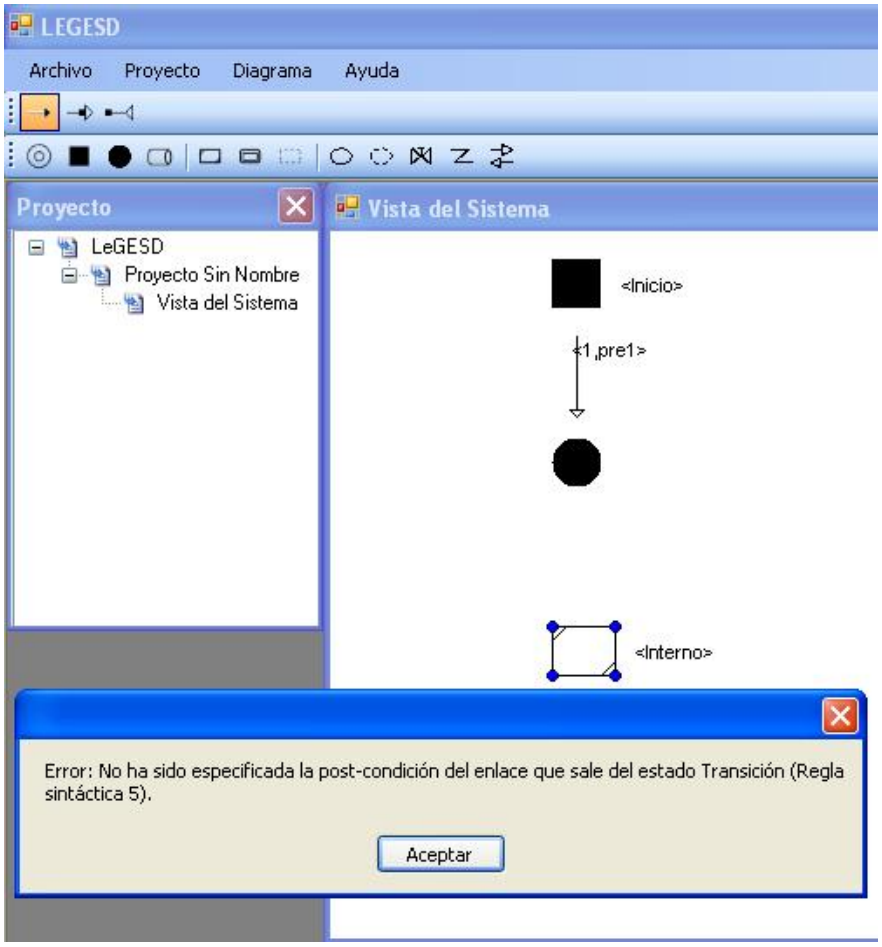


Figura 8.76: Error resultante de la verificación de la regla sintáctica 5 para la post-condición.

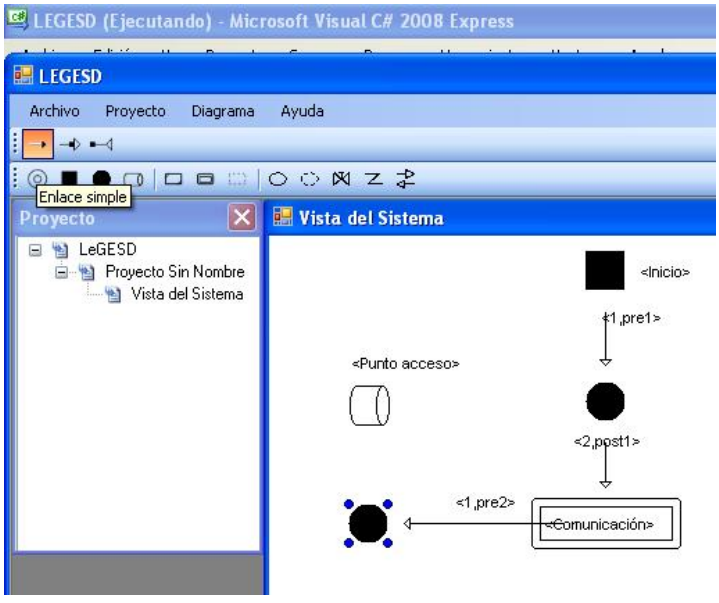


Figura 8.77: Verificación de la regla sintáctica 8 de LeGESD por la herramienta computacional.

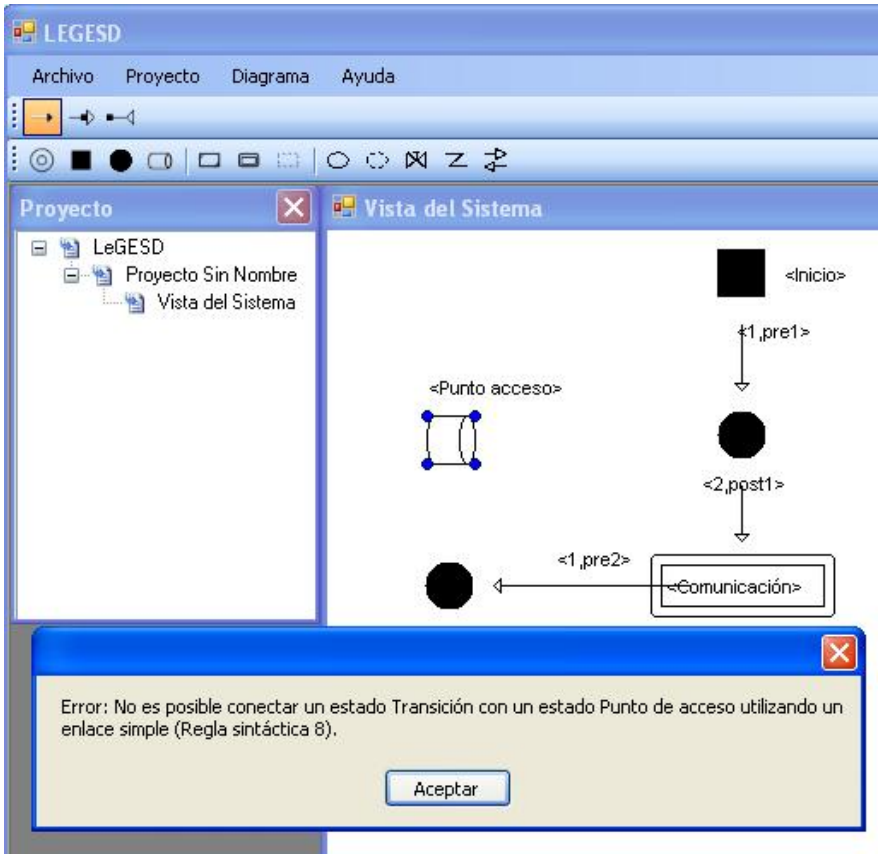


Figura 8.78: Error resultante de la verificación de la regla sintáctica 8.

Con esto concluye la presentación de la herramienta computacional de LeGESD. En el último capítulo, se proporcionan las conclusiones y trabajos a futuro relacionados con esta tesis.

9 Conclusiones y trabajos a futuro

Con la aplicación de LeGESD en la especificación de un sistema distribuido a través de los casos de estudio, y como cierre del presente trabajo de tesis, se obtienen algunas conclusiones y se visualiza a futuro el trabajo potencial a desarrollar con LeGESD. El objetivo del presente capítulo es presentar las conclusiones y trabajos a futuro de esta tesis doctoral.

9.1. Conclusiones

La especificación de sistemas, en general, es usada para describir los componentes de un sistema, así como también para describir todas las relaciones existentes entre dichos componentes. Esta descripción es realizada mediante diferentes técnicas: modelos relacionales de información, modelos orientados a objetos, modelos basados en máquinas de estados finitos, modelos matemáticos dinámicos, o lenguajes de modelado descriptivos auxiliados por herramientas de modelado y metodologías.

Es justamente en la técnica de los lenguajes de modelado descriptivos auxiliados por herramientas de modelado y metodologías, donde se incorpora el trabajo que ha sido desarrollado en estos capítulos.

A lo largo de este trabajo de tesis doctoral se ha presentado el Lenguaje Gráfico de Especificación de Sistemas Distribuidos nombrado LeGESD, el cual está integrado por un lenguaje visual (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 jerárquica, modular y escalable. El lenguaje considera tanto los requerimientos funcionales como de comunicación del sistema, haciéndolo idóneo para la especificación de sistemas distribuidos.

La semántica que sustenta al lenguaje LeGESD se ha denominado Análisis y Diseño de Sistemas Distribuidos (ADSD), la cual utiliza el álgebra de procesos para proporcionar una semántica precisa que describe la interacción entre los componentes que integran a LeGESD: trabajos y medios. De acuerdo al estudio presentado en esta tesis, se ha encontrado que ADSD tiene características útiles para la verificación de propiedades de seguridad en los sistemas distribuidos

especificados con LeGESD. Estas características son:

- Uso del álgebra de procesos.
- Aplicación de un algoritmo que permite obtener la transformación y equivalencia gráfico-algebraica del lenguaje.
- Generación de expresiones en ADSD relacionadas con los procesos derivados de la especificación gráfica, y que son reflejadas en las ecuaciones de transición.
- Apoyo en una herramienta matemática basada en un teorema de equivalencia gráfico-algebraica, el cual establece que una especificación gráfica es equivalente a su especificación algebraica, permitiendo la verificación de propiedades de seguridad en la especificación gráfica a partir de la especificación algebraica.

Con la semántica ADSD basada en un álgebra de procesos se tiene una semántica precisa del lenguaje basada en procesos, la cual puede apoyar al análisis mediante la verificación de propiedades de seguridad de sistemas distribuidos, apoyándose en las características de modularidad y jerarquía de LeGESD.

Recordando los objetivos específicos planteados al inicio de este trabajo de tesis, se tiene que éstos son:

- Definir una notación gráfica (símbolos) para la especificación de sistemas distribuidos.
- Definir los elementos del lenguaje gráfico y sus reglas sintácticas para la especificación de sistemas distribuidos.
- Desarrollar una herramienta para validar la sintaxis del lenguaje a través de sus reglas sintácticas definidas para la construcción de estructuras sintácticas válidas.
- Definir una semántica algebraica para el lenguaje gráfico.
- Diseñar y verificar un algoritmo de transformación que tome como entrada una especificación gráfica en LeGESD y devuelva una especificación algebraica en ADSD.
- Desarrollar una herramienta matemática consistente de un teorema de equivalencia gráfico-algebraica para establecer la equivalencia entre la especificación gráfica y su especificación algebraica.
- Mostrar la aplicación del lenguaje propuesto y del algoritmo de transformación por medio de dos casos de estudio.
- Verificar las propiedades de seguridad de los dos casos de estudio.

Después de haber desarrollado y presentado los resultados obtenidos de esta tesis podemos establecer que:

- Se ha definido una notación gráfica para la especificación de sistemas distribuidos, la cual fue presentada en el capítulo 5.

- Se han definido los elementos del lenguaje gráfico LeGESD y sus reglas sintácticas para la especificación de sistemas distribuidos (sintaxis del lenguaje), los cuales fueron presentados también en el capítulo 5.
- Se han propuesto quince reglas sintácticas las cuales permiten la construcción de estructuras sintácticas válidas en los diagramas utilizados en LeGESD. Las reglas sintácticas fueron presentadas también en el capítulo 5.
- Se ha definido una semántica algebraica que establece una equivalencia entre la especificación gráfica y su especificación algebraica. La semántica algebraica fue presentada en el capítulo 6.
- Se ha diseñado el Algoritmo de transformación de estados LeGESD a procesos ADSD (TLA), el cual permite realizar la transformación de una especificación gráfica en LeGESD a su especificación algebraica en ADSD. El diseño del algoritmo fue presentado también en el capítulo 6.
- Se han propuesto cinco teoremas auxiliares y el Teorema de equivalencia gráfico-algebraica para establecer la equivalencia entre la especificación gráfica y su especificación algebraica. La herramienta matemática fue presentada también en el capítulo 6.
- Se ha sustentado la semántica de LeGESD a través de una herramienta matemática consistente del Algoritmo de transformación de estados LeGESD a procesos ADSD (TLA) y del Teorema de equivalencia gráfico-algebraica.
- Se ha demostrado experimentalmente la aplicación de LeGESD para la especificación y comprobación de propiedades de seguridad de sistemas distribuidos a través de dos casos de estudios, el sistema Productor-Consumidor y el sistema Cena de criptógrafos, los cuales fueron presentados en el capítulo 7.

Al cumplir con cada uno de estos objetivos específicos, es posible afirmar que el objetivo general de la tesis se ha alcanzado, estableciendo a LeGESD como un lenguaje gráfico para la especificación de sistemas distribuidos que posee una semántica formal que permite demostrar propiedades de seguridad de los sistemas especificados. Entre las características principales que presenta LeGESD, y que resuelven los problemas puntuales que se señalaron en el capítulo 1, se pueden mencionar las siguientes:

- Permite la especificación de elementos gráficos complejos (diagramas de comportamiento y comunicación), resultantes de la agrupación de diversos símbolos gráficos, como una composición jerárquica de los diversos diagramas que pueden crearse. Esto forma parte de la sintaxis de LeGESD, la cual fue definida en el capítulo 5 (sección 5.4.1). Concretamente, se puede observar en la definición de los componentes principales de LeGESD: trabajos y medios, y en sus quince reglas sintácticas. Los trabajos y medios se pueden agrupar de manera jerárquica, modular y escalable mediante sus diversos diagramas de comportamiento y de comunicación, lo cual quedó de manifiesto en el desarrollo de los casos de estudio en el capítulo 7 (secciones 7.2 y 7.3).

- Permite mantener relaciones gráficas y sintácticas entre los símbolos gráficos encontrados en los diversos niveles jerárquicos de la especificación. Esto lo realiza mediante las reglas sintácticas que fueron establecidas para la construcción de estructuras sintácticas válidas y con un significado preciso. Las reglas sintácticas de LeGESD fueron presentadas en el capítulo 5 (sección 5.4.3), la aplicación de estas reglas se presentó en los casos de estudio del capítulo 7 (secciones 7.2 y 7.3).

- Tiene una semántica formal algebraica llamada ADSD, la cual establece la equivalencia local entre los símbolos gráficos y sus procesos asociados en ADSD, de manera tal que posibilita a la especificación gráfica ser transformada a su especificación algebraica equivalente. Esto se logra definiendo una especificación composicional, una semántica operacional y un análisis de comportamiento, las cuales se definieron en el capítulo 6 (sección 6.2.4 y 6.3) y su aplicación se presentó en el capítulo 7 durante el desarrollo de los casos de estudio (secciones 7.2 y 7.3).

    - Su especificación composicional se define mediante la gramática de operadores válidos en ADSD. La definición de estos cinco operadores de ADSD fue hecha en el capítulo 6 (sección 6.2.4).
    - Su semántica operacional define cinco reglas de transición las cuales indican cómo desde un proceso x se puede alcanzar un proceso z mendiante una acción a. La definición de estas reglas de ADSD fue hecha en el capítulo 6 (también en la sección 6.2.4).
    - Su análisis de comportamiento (capítulo 6, sección 6.3) define la equivalencia entre los términos algebraicos de la semántica operacional de ADSD (procesos ADSD) y los elementos sintácticos de LeGESD (estados LeGESD), la definición de este análisis de comportamiento integra un algoritmo de transformación de estados LeGESD a procesos ADSD (llamado TLA), el cual transforma cada estado LeGESD a su proceso ADSD equivalente (capítulo 6, sección 6.3.1.1). Este algoritmo utiliza cinco teoremas auxiliares que se asocian con las cinco reglas de transformación establecidas en la semántica operacional. Estos teoremas auxiliares transforman las estructuras gráficas principales de LeGESD con sus correspondientes procesos y operadores de ADSD (capítulo 6, sección 6.3.1.2). Para establecer la equivalencia entre la especificación gráfica y la algebraica, se propone el Teorema de equivalencia gráfico-algebraica, el cual se apoya en el algoritmo diseñado (capítulo 6, sección 6.3.2). Tanto el algoritmo TLA como el Teorema de equivalencia gráfico-algebraica forman la herramienta matemática propuesta en LeGESD.

Finalmente, se puede concluir que con LeGESD es posible especificar de manera gráfica un sistema distribuido, y realizar la verificación de sus propiedades de seguridad a través del álgebra de procesos ADSD que incorpora el lenguaje, y que esta verificación realizada en el dominio algebraico de ADSD es igualmente válida en el dominio gráfico de LeGESD.

9.1.1. Aportaciones de la tesis

Las principales aportaciones de la presente tesis son:

- La sintaxis del lenguaje gráfico LeGESD, la cual consiste en la definición de una notación gráfica, de los elementos principales del lenguaje, y de sus reglas sintácticas para la especificación de sistemas distribuidos. Esta notación gráfica consiste de un conjunto de símbolos que fueron explicados en el capítulo 5, sección 5.4.1, mediante los cuales es posible construir diagramas que representen la especificación del sistema distribuido. La construcción de los diagramas se lleva a cabo aplicando las reglas sintácticas establecidas en LeGESD, las cuales fueron presentadas en el capítulo 5, sección 5.4.3. Estas reglas establecen la forma correcta de relacionar los símbolos gráficos para la construcción correcta de los diagramas.

- La semántica formal del lenguaje gráfico LeGESD, la cual consiste en la definición de una semántica algebraica llamada ADSD. ADSD está basado en álgebra de procesos, definiéndose para esta: una especificación composicional, una semántica operacional (las cuales fueron explicadas en el capítulo 6, sección 6.2.4), y un análisis de comportamiento (el cual fue explicado en el capítulo 6, sección 6.3).

- El desarrollo de una herramienta matemática compuesta por el algoritmo de transformación TLA y por el Teorema de equivalencia gráfico-algebraica que establece la equivalencia entre las especificaciones gráficas en LeGESD y el álgebra ADSD. Esta herramienta matemática es el andamiaje necesario para establecer una equivalencia entre la especificación gráfica y la especificación algebraica de un sistema distribuido, y viceversa. Con ello, es posible realizar demostraciones sobre propiedades de seguridad en la especificación algebraica, las cuales son igualmente válidas para la especificación gráfica. La herramienta matemática fue presentada en el capítulo 6, sección 6.3.

- La demostración experimental en dos casos de estudios de la aplicación de LeGESD para la especificación de sistemas distribuidos. Los dos casos de estudios son: el sistema productor-consumidor, el cual fue presentando en el capítulo 7, sección 7.2; el sistema cena de criptógrafos, el cual fue presentado en el capítulo 7, sección 7.3.

- La verificación de propiedades de seguridad de los sistemas distribuidos especificados como casos de estudio, utilizando la especificación algebraica de ADSD, con la característica de que dicha verificación es igualmente válida para la especificación gráfica. Para llevar a cabo esta verificación se partió de la especificación algebraica de los sistemas especificados en los casos de estudio, dicha especificación algebraica fue obtenida aplicando la herramienta matemática de LeGESD, la cual fue presentada en el capítulo 6, sección 6.3. Esta herramienta permite realizar demostraciones sobre seguridad en la especificación algebraica, las cuales son igualmente válidas para la especificación gráfica de los sistemas correspondientes a los casos de estudio.

9.1.2. Diferenciación de LeGESD con trabajos relacionados

En la tabla 9.1 se muestra una comparativa de LeGESD con los trabajos relacionados que fueron presentados en el capítulo 4, sección 4.2:

Características a comparar	LeGESD	IOA	PEDS	LfP
Tipo de notación utilizada para la especificación de sistemas.	Gráfica.	Textual.	Gráfica.	Gráfica.
Tipo de formalismo que sustenta a la especificación.	Álgebra de procesos ADSD.	Autómata IOA.	No tiene.	Redes de Petri.
Herramientas de apoyo al formalismo.	Algoritmo de Transformación de estados LeGESD a procesos ADSD (TLA) y Teorema de equivalencia gráfico-algebraica.	Teoremas de IOA.	No tiene.	No tiene.

Tabla 9.1: Comparativa de LeGESD con trabajos relacionados.

De la tabla 9.1 se puede observar que la principal diferencia que tiene LeGESD con respecto a los otros lenguajes es la herramienta de apoyo al formalismo, la cual consiste tanto en el Algoritmo de Transformación de estados LeGESD a procesos ADSD (TLA) como en el Teorema de equivalencia gráfico-algebraica. Esta herramienta matemática permite relacionar bidireccionalmente la especificación gráfica con su equivalente especificación algebraica, permitiendo aseverar que propiedades demostrables sobre la especificación algebraica son igualmente válidas sobre la especificación gráfica, y viceversa.

9.1.3. Limitantes de la tesis

La limitante de LeGESD es que no es posible realizar demostraciones de vitalidad sobre las especificaciones gráficas, únicamente es posible realizar demostraciones de seguridad. Lo anterior se debe a que bajo el desarrollo propuesto en la tesis, no se tiene forma de demostrar que hay estados que por fuerza deben ser alcanzados (las demostraciones de vitalidad dicen que hay estados deseables que siembre son alcanzables). Lo que se ha realizado propiamente en el desarrollo de la presente tesis, es suponer que si un estado es alcanzable, entonces dicho estado debe cumplir con un conjunto de predicados sobre las variables que definen el estado, pero no se tiene forma de demostrar que el sistema va a llegar a un estado dado.

9.2. Trabajos a futuro

En la creación de lenguajes visuales es de trascendencia el definir claramente su sintaxis y semántica. Esto se debe a que si la sintaxis y la semántica son formalmente especificadas, pueden crearse herramientas de soporte, ya que todas las herramientas de desarrollo deben estar basadas en definiciones inequívocas proporcionadas por el lenguaje.

LeGESD al contar con una sintaxis y semántica claramente definida, posibilita la creación de herramientas de soporte al lenguaje. Recordando las etapas necesarias para la construcción automática de sistemas distribuidos:

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 dos primeras etapas son la base necesaria para la adecuada construcción de todo sistema distribuido, mientras que las dos últimas etapas son útiles tanto para la contrsucción automática de estos sistemas así como para realizar análisis de desempeño y experimentos preliminares. Con la creación de LeGESD, se han cubierto las dos primeras etapas para la construcción de sistemas distribuidos, quedando como trabajos a futuro el desarrollar:

1. Un generador de código fuente para uno o varios paradigmas de programación que utilice a LeGESD como lenguaje de especificación.
   La implementación de un generador de código fuente permitirá implementar de manera automatizada parte del código del sistema especificado a través de LeGESD. Esta implementación podrá realizarse en uno o varios paradigmas de programación, de acuerdo a las necesidades funcionales del sistemas especificado. Lo anterior es posible debido a que la especificación hecha en LeGESD tiene independencia sobre el ambiente de ejecución en el cual se piensa implementar el sistema distribuido. Para lograr esta implementación, el generador de código fuente deberá procesar las estructuras algebraicas generadas por ADSD (ecuaciones de transición), y mapearlos a estructuras programáticas de acuerdo al paradigma y lenguaje de programación en los que se desee implementar el sistema especificado. Una posible forma de realizar este mapeo es utilizando árboles sintácticos.

2. Un simulador de eventos discretos.
   El simulador permitirá caracterizar experimentalmente el desempeño de los sistemas especificados por medio del lenguaje gráfico. La entrada del simulador será el código generado.

3. Una metodología matemática para caracterizar la complejidad temporal, espacial y de mensajes de las especificaciones gráficas.

De estos trabajos a futuro, es posible desprender interesantes líneas de investigación relacionadas con la construcción automática de sistemas distribuidos y su incorporación a diversos ámbitos computacionales, donde los sistemas distribuidos sean requeridos como la base de operación de las aplicaciones a desarrollar.

En relación a trabajos a futuro propiamente involucrados con el crecimiento y depuración de LeGESD se tienen los siguientes:

- Aplicación de LeGESD a la especificación de sistemas distribuidos extensos.
  Los sistemas distribuidos de tipo industrial podrían ser aplicaciones de interés para especificarse haciendo uso de LeGESD. Este tipo de sistemas por sus características de complejidad serán un alto parámetro de prueba sobre la funcionalidad del lenguaje para la especificación de estos sistemas.

- Definición de esquemas algebraicos para la generación de código y la simulación de modelos.
  Los esquemas algebraicos derivados de LeGESD, deberán estar sustentados en el álgebra de procesos ADSD mediante la utilización de la estructura algebraica definida por ADSD. Esta estructura algebraica consiste en la especificación composicional, la semántica operacional

y el análisis de comportamiento que mantiene ADSD. Los esquemas a proponer deberán estar en concordancia con la estructura algebraica de ADSD, lo cual puede ser conseguido a través del establecimiento de un conjunto de reglas de mapeo entre los esquemas y la estructura algebraica.

- Diseño de experimentos con usuarios para validar la facilidad de uso de LeGESD.
  Estos experimentos deberán diseñarse definiendo parámetros e indicadores que permitan crear instrumentos para validar la facilidad de uso de LeGESD, a través de la evaluación de los instrumentos podrá realizarse el ajuste al lenguaje, en caso de que no se haya alcanzado el nivel deseado, para proporcionar la facilidad de uso buscada.

Referencias

[1] L. Arief, M. Little, S. Shrivastava, N. Speirs, and S. Wheater, "Specifying distributed system services," *BT Technical Journal (Special Issue)*, vol. 3, pp. 120–128, 1999.

[2] N. Lynch and M. Tuttle, "An introduction to input/output automata," *CWI-Quarterly*, vol. 3, pp. 219–246, 1989.

[3] J. Cortes and F. Menchaca, "Graphical specification language for distributed systems," in *Proceeding IEEE on 15th International Conference on Computing*, (México D.F., México), pp. 120–126, 2006.

[4] I. Kinchin and 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, pp. 43–57, 2000.

[5] J. Rumbaugh, I. Jacobson, and G. Booch, "The unified modeling language reference manual," *Object Technology Series*, vol. 1, pp. 13–62, 1998.

[6] D. Luckham, "Specification and analysis of system architecture," *IEEE Transactions on Software Engineering*, vol. 21, pp. 336–355, 1995.

[7] A. Rodriguez and C. Killian, "Macedon: Methodology for automatically creating, evaluating, and designing overlay networks," *Proceedings of the NSDI*, vol. 1, pp. 20–34, 2004.

[8] S. Garland and N. Lynch, *The IOA language and toolset: Support for designing, analyzing, and building distributed systems*. MIT Press Technical Report, USA, 1998.

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

[10] D. Zhang and K. Zhang, "A visual programming environment for distributed systems," in *Proceedings of the 11th International IEEE Symposium on Visual Languages*, (Washington, Estados Unidos de America), pp. 310–317, 1995.

[11] J. Cortes and F. Menchaca, "Algebra de procesos aplicada a la especificación formal de sistemas distribuidos," in *1er. Congreso Internacional en Sistemas Computacionales y Electrónicos*, (México D.F., México), pp. 30–38, 2006.

[12] H. Hermanns and U. Herzog, "Process algebra for performance evaluation," *Theoretical Computer Science*, vol. 274, pp. 43–87, 2002.

[13] L. Grunske, K. Winter, and N. Yatapanage, "Defining the abstract syntax of visual languages with advanced graph grammars-a case study based on behavior trees," *Journal of Visual Languages and Computing*, vol. 19, pp. 343–379, 2008.

[14] T. J. R. Chow, *Distributed Operating Systems and Algorithms*. Addison Wesley, USA, 1998.

[15] G. Booch, J. Rumbaugh, and I. Jacobson, *El Proceso Unificado de Desarrollo Software*. Addison Wesley, USA, 1999.

[16] J. Rumbaugh and et al, *Object-Oriented Modeling and Design*. Prentice Hall, USA, 1991.

[17] A. Eliëns, *Principles of Object-Oriented Software Development*. Addison Wesley, USA, 1995.

[18] B. Meyer, *Object-Oriented Software Construction*. Prentice Hall, USA, 1997.

[19] R. Milner, *A Calculus of Communication Systems*. Springer Verlag, USA, 1980.

[20] K. Honda and K. Tokoro, "On asynchronous communication semantics," *Object-Based Concurrent Computing*, vol. 612, pp. 21–51, 1992.

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

[22] N. C. Shu, "Visual programming language: A perspective and a dimensional analysis," *Visual Languages*, vol. 1, pp. 11–34, 1986.

[23] E. J. Golin, *A Method for the Specification and Parsing of Visual Language*. Brown University, USA, 1990.

[24] R. Helm and K. Marriot, "A declarative specification and semantics for visual language," *Journal of Visual Languages and Computing*, vol. 2, pp. 311–331, 1991.

[25] L. Botturi and T. Stubbs, *Handbook of Visual Languages for Instructional Design*. Information Science Reference, USA, 2008.

[26] L. Lockyer, S. Bennett, S. Agostinho, and B. Harper, *Handbook of Research on Learning Design and Learning Objects: Issues Applications and Technologies*. Information Science Reference, USA, 2008.

[27] D. Merrill, J. V. Merrienboer, and M. Driscoll, *Handbook of Research in Instructional Design*. AECT, USA, 2008.

[28] E. Guerra, J. de Lara, and P. Diaz, "Visual specification of measurement and redesign for domain specific visual language," *Journal of Visual Languages and Computing*, vol. 19, pp. 399–425, 2008.

[29] C. Simons and G. Wirtz, "Modeling context in mobile distributed systems with uml," *Journal of Visual Languages and Computing*, vol. 18, pp. 420–439, 2007.

[30] G. D. Plotkin, "A structured approach to operational semantics," Tech. Rep. FM-19, DAIMI, 1998.

[31] R. Milner, "Calculi for synchrony and asynchrony," *Theoretical Computer Science*, vol. 25, pp. 269–310, 1983.

[32] R. Milner, "A complete inference system for a class of regular behaviours," *Journal of Computer and System Science*, vol. 28, pp. 439–466, 1984.

[33] R. Milner, *Process Constructors and Interpretations*. IFIP-WG Information Processing, USA, 1986.

[34] R. Milner, *Communication and Concurrency*. Prentice Hall, USA, 1989.

[35] R. J. V. Glabbeek, "The linear time - branching time spectrum ii: The semantics of sequential systems with silent moves," *Fourth International Conference on Concurrency Theory*, vol. 715, pp. 66–81, 1993.

[36] M. Hennessy and T. Regan, "A process algebra for timed systems," *Information and Computation*, vol. 117, pp. 221–239, 1995.

[37] B. Plateau and K. Atif, "Stochastic automata network for modeling parallel systems," *IEEE Transactions on Software Engineering*, vol. 17, pp. 15–33, 1991.

[38] K. Honda and K. Tokoro, "On asynchronous communication semantics," *Object-Based Concurrent Computing*, vol. 612, pp. 21–51, 1992.

[39] H. Hermanns, U. Herzog, and J. P. Katoen, *Process algebra for performance evaluation*. Theoretical Computer Science, USA, 2001.

[40] G. Coulouris, J. Dollimore, and T. Kindberg, *Sistemas Distribuidos, Conceptos y Diseño*. Addison Wesley, México, 2001.

[41] J. Farley, *Java Distributed Computing*. O Reilly, USA, 1998.

[42] R. Chow and T. Johnson, *Distributed Operating Systems and Algorithms*. Addison Wesley, USA, 1998.

[43] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, USA, 1996.

[44] A. Tanenbaum, *Sistemas Operativos Distribuidos*. Prentice Hall, México, 1996.

[45] O. M. Group, *UML version 1.4*. <http://www.omg.org/technology/documents/formal/uml.htm>. Septiembre, 2001.

[46] J. S. Garland and N. A. Lynch, "The ioa language and toolset: Support for de-signing, analyzing, and building distributed systems," Tech. Rep. MIT/LCS/TR-762, Laboratorio de Ciencias de la Computación, Instituto de Tecnología de Massachusetts, Cambridge, USA, Agosto 1998.

[47] J. K. Goldman, T. B. Swaminathan, P. McCartney, M. D. Anderson, and R. Sethuraman, "The programmers' playground: I/o abstraction for user-configurable distributed applications," *IEEE Transactions on Software Engineering*, vol. 21, pp. 735–746, 1995.

[48] N. A. Lynch and M. R. Tuttle, "Hierarchical correctness proofs for distributed algorithms," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pp. 137–151, 1987.

[49] M. Vaziri, J. A. Tauber, M. J. Tsai, and N. Lynch, "Systematic removal of nondeterminism for code generation in i/o automata," Tech. Rep. MIT/LCS/TR-960, Laboratorio de Ciencias de la Computación, Instituto de Tecnología de Massachusetts, Cambridge, USA, Julio 2004.

[50] D. Regep and F. Kordon, "Lfp: A specification language for rapid prototyping of concurrent systems," in *Proceedings of the 12th International Workshop on Rapid System Prototyping. IEEE Computer Society*, pp. 90–97, 2001.

[51] F. Kordon, I. Mounier, E. Paviot-Adet, and D.Regep, "Formal verification of embedded distributed systems in a prototyping approach," in *Monterey Workshop on Engineering Automation for Software Intensive System Integration*, 2001.

[52] D. Regep, Y. Thierry-Mieg, and F. Kordon, "Modélisation et vérification de systémes répartis: une approche intégrée avec lfp," in *Proceedings of AFADL '03*, 2003.

[53] ITU-T, *Open Distributed Processing, Estándares X.901, X.902, X.903 y X.904*. <http://www.itu.int/itudoc/itu-t/rec/x/x500up>, 2001.

[54] F. Kordon, F. Gilliers, and J. P. Velu, "Generation of distributed programs in their target execution environment," in *Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping (RSP '04). IEEE Computer Society*, 2004.

[55] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, pp. 231–274, 1987.

[56] K. Chu, "The cognitive aspects of chinese character processing," *Visual Languages*, vol. 1, pp. 349–392, 1986.

[57] F. S. Montalvo, *Diagram understanding: The symbolic descriptions behind the scenes in Visual Languages and Application*. Plenum Press, USA, 1990.

[58] D. Ritchie and B. Kernighan, *El Lenguaje de Programación C*. Prentice Hall, México, 1992.

[59] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice Hall, USA, 1985.

[60] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[61] D. Baelde, S. Delaune, and L. Hirschi, "A reduced semantics for deciding trace equivalence using constraint systems," *Principles of Security and Trust. Lecture Notes in Computer Science*, vol. 8414, pp. 1–21, 2014.