COMPOSICIÓN DE SERVICIOS CONVERGENTES MEDIANTE EL USO DE PATRONES DE FLUJO DE EJECUCION EN UN ENTORNO TELCO 2.0

ANDRÉS BENAVIDES PEÑA
GUSTAVO ADOLFO ENRíQUEZ HUERTAS

ANEXOS

Director:
Director: Juan Carlos Corrales
Asesor: Jesús David Ramirez

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Grupo de Ingeniería Telemática
Popayán, Noviembre de 2012

ANEXOS

Contenido

Lista de Figuras

Lista de Tablas

Anexo A

A Patrones de Flujo de Control

A.1 Introducción

Éste anexo presenta en la Seccion A.2 la evaluación realizada por Vander Aalst *et al* [1], [2] de los patrones de flujo de control (CFP) soportados por JBPM y BPEL y en la Seccion A.3, la representación formal en redes de Petri y grafos de los patrones soportados por ambos lenguajes.

A.2 Evaluacion de CFP soportados por JBPM y BPEL

Patrones de Flujo de Control	**Viabilidad de Implementación***	
Patrones Básicos	BPEL	JBPM
Sequence	+	+
Parallel split	+	+
Synchronization	+	+
Exclusive choice	+	+
Simple merge	+	+
Patrones de Ramificación y Sincronización Avanzada		
Multi-choice	+	-
Structured synchronizing merge	+	-
Multi-merge	-	+
Structured discriminator	-	-
Blocking discriminator	-	-
Cancelling discriminator	-	-
Structured partial join	-	-
Blocking partial join	-	-
Cancelling Partial Join	-	-
Generalized AND-join	-	+
Local synchronizing merge	+	-
General synchronizing merge	-	-
Thread merge	+/-	+/-
Thread split	+/-	+/-
Patrones Basados en Estados		
Deferred Choice	-	+
Interleaved Parallel Routing	+/-	-
Milestone	-	-
Critical section	+	-
Interleaved Routing	+	-
Patrones de Múltiples Instancias		
Multiple instances without synchronization	+	+

Multiple instances with a prior design-time knowledge	-	-
Multiple instances with a prior run-time knowledge	-	-
Multiple instances without a prior run-time knowledge	-	-
Static partial join for multiple instances	-	-
Cancelling partial Join for multiple instances	-	-
Dynamic partial Join for multiple instances	-	-
Patrones de Cancelación y Terminación Forzada		
Cancel task	+	+
Cancel case	+	-
Cancel region	+/-	-
Cancel multiple instance activity	-	-
Complete multiple instance activity	-	-
Patrones de Iteración		
Arbitrary cycles	-	+
Structured loop	+	-
Recursion	-	-
Patrones de Terminación		
Implicit termination	+	+
Explicit termination	-	-
Patrones de Activación		
Transient trigger	-	+
Persistent Trigger	+	-
* **+** Si lo soporta; **+/-** Lo soporta con limitantes; **–** No lo soporta		

La lista de los patrones soportados por ambos lenguajes se resume en la Tabla A-1.

Tabla A-1: Patrones soportados por JBPM o BPEL

	Patrón
1	Sequence
2	Parallel Split
3	Synchronization
4	Exclusive Choice
5	Simple Merge
6	Multi-choice
7	Structured synchronizing merge
8	Multi-merge
9	Generalized AND-join
10	Local syncrhonizing merge
11	Deferred Choice
12	Critical Section
13	Interleaved Routing
14	Multiple instances without synchronization
15	Cancel Task
16	Cancel Case
17	Arbitrary Cycles
18	Structured Loop
19	Implicit Termination
20	Transient Trigger
21	Persistent Trigger

A.3 Descripción de los Patrones Seleccionados

A.3.1 Sequence

Este patrón representa el paso del flujo de un lugar a una transición, en el caso de la representación en redes de Petri que se muestra en la Figura A-1 (Izq); y el paso de datos de un nodo a otro en el caso de la representación mediante grafos (Figura A-2 (der)). Para que se considere un patrón de secuencia debe contener al menos 3 lugares en el caso de las redes de Petri y 3 nodos en el caso de los grafos

Figura A-1: Patrón Sequence representado mediante redes de Petri y grafos

A.3.2 Parallel Split

Este patrón representa la divergencia del flujo de control de una rama entrante, en 2 o más ramas salientes. La compuerta ANDS divide el flujo en cada una de las ramas para que cada una de sus actividades se realice de forma simultánea. En la Figura A-2 se ilustra el patrón Parallel Split mediante las representaciones formales de redes de Petri y Grafos.

.

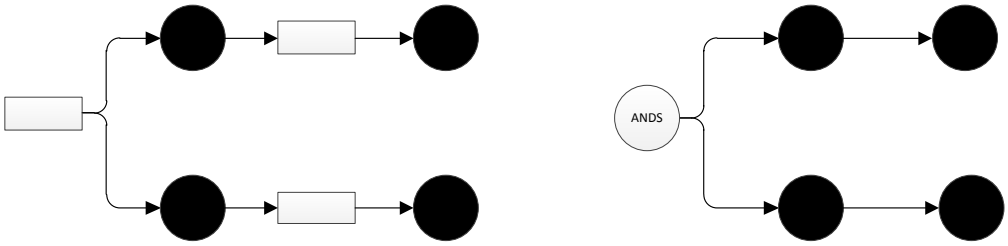


Figura A-2: Patrón Parallel Split representado mediante redes de Petri y grafos

A.3.3 Synchronization

Este patrón representa la convergencia del flujo de control de dos ramas en una sola, tal que el flujo de control de la rama saliente ocurra solo cuando el flujo de las ramas entrantes, converja después de ejecutarse cada una de sus actividades. La compuerta ANDJ, sincroniza en flujo de las ramas entrantes en un solo flujo saliente. En Figura A-3 se ilustra el patrón Synchronization mediante las representaciones formales de redes de Petri y Grafos.

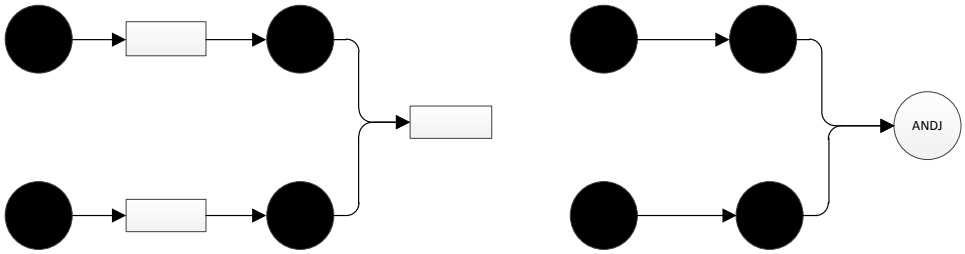


Figura A-3: Patrón Synchronization representado mediante redes de Petri y grafos

A.3.4 Exclusive Choice

Este patrón representa la divergencia del flujo de control, de una rama entrante en dos o más ramas salientes, teniendo en cuenta una condicional previa. A diferencia del patrón *Parallel Split*, el flujo solo continúa por una de las ramas salientes. La compuerta XORS garantiza la divergencia del flujo en una sola rama. En la Figura A-4 se ilustra el patrón Exclusive Choice mediante las representaciones formales de redes de Petri y Grafos.

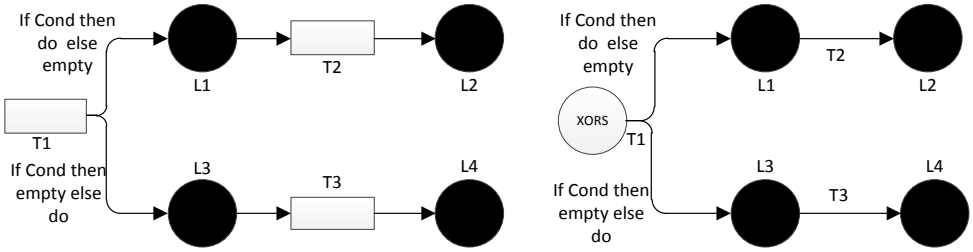


Figura A-4: Patron Exclusive Choice representado mediante redes de Petri y grafos

A.3.5 Simple Merge

Este patrón representa la convergencia del flujo de control, de dos o más ramas entrantes en una rama saliente de manera asíncrona. La compuerta XORJ permite que el flujo proveniente de una de las ramas entrantes, independientemente del momento en que termine su ejecución, pase a la rama saliente de forma asíncrona. En la Figura A-5 se ilustra el patrón Simple Merge mediante las representaciones formales de redes de Petri y Grafos.

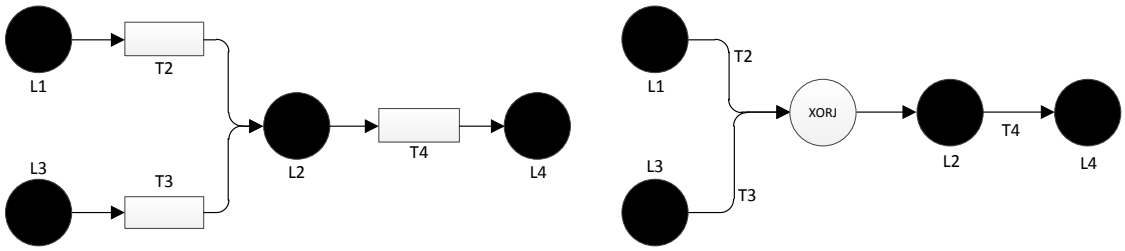


Figura A-5: Patrón Simple Merge representado mediante redes de Petri y grafos

A.3.6 Multi-Choice

Este patrón representa la divergencia del flujo de control, de una rama entrante en dos o más ramas salientes, teniendo en cuenta una condicional previa. Este patrón se diferencia de *Exclusive Choice* ya que el flujo puede continuar por cualquier rama saliente o por todas las ramas salientes simultáneamente, la compuerta ORS representa estas características.
Adicionalmente este patrón puede ser representado con la conjunción de los patrones *Parallel Split* y *Exclusive* Choice. La Figura A-6 ilustra el patrón Multi-Choice

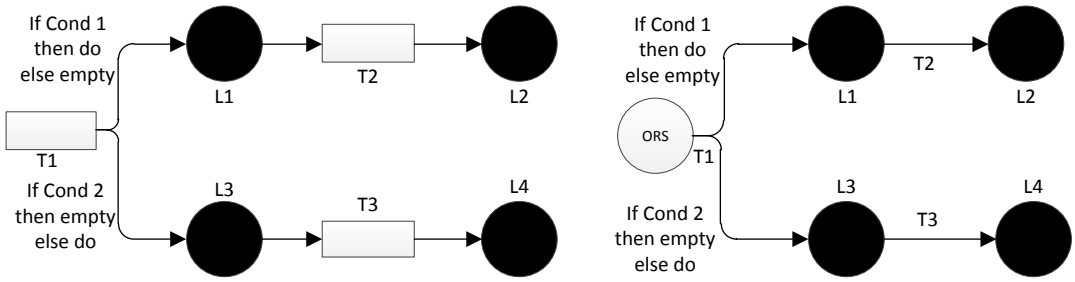


Figura A-6: Patrón Multi Choice representado mediante redes de Petri y Grafos

A.3.7 Structured Synchronizing Merge

Este patrón permite la sincronización del flujo proveniente de dos o más ramas entrantes en una rama saliente. A diferencia del patrón *Synchronization,* este tiene en cuenta el flujo proveniente de las ramas entrantes antes de realizar la sincronización. La Figura A-8 muestra una compuerta ORS la cual diverge el flujo por las ramas superiores o inferiores, o por ambas simultáneamente. En redes de Petri el flujo es valuado en la transición T4 (Figura A-7); si las condiciones 1 y 2 se cumplieron, el flujo ingresa por ambas entradas de T4; si solo una condición se cumplió, el flujo ingresa por solo una rama de T4, mientras que la otra estará en "*empty*" por lo cual el flujo se sincroniza sin tener en cuenta la otra rama. Para representar este patrón son necesarias dos compuertas lógicas: Una de divergencia ORJ y una de sincronización ANDJ.

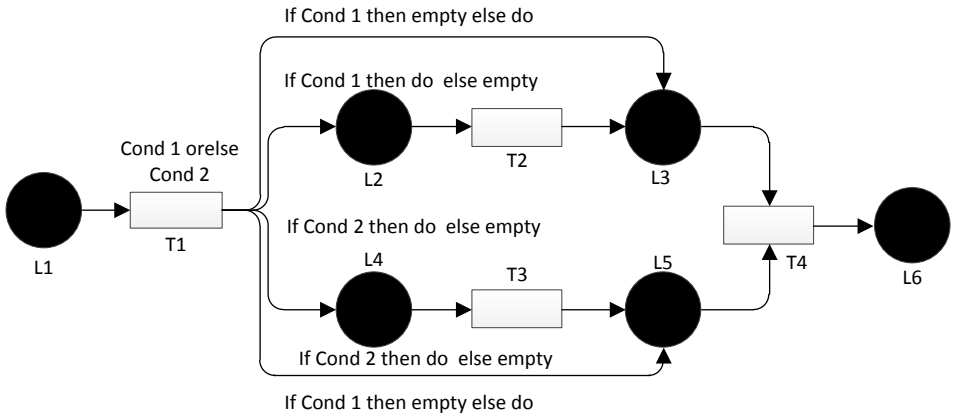


Figura A-7: Patrón Structured Synchronizing Merge representado mediante redes de Petri

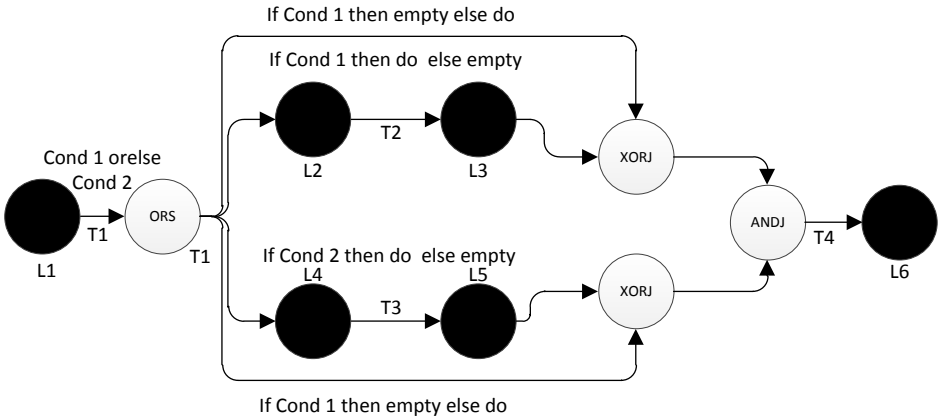


Figura A-8: Figura 8: Patrón Structured Synchronizing Merge representado mediante Grafos

A.3.8 Multi Merge

Este patrón representa la convergencia del flujo de control, de dos o más ramas entrantes en una rama saliente de manera asíncrona. La compuerta ORJ permite que el flujo proveniente de una o ambas ramas entrantes, independientemente del momento en que termine su ejecución, pase a la rama saliente de forma asíncrona. La diferencia con el patrón *Simple Merge* se ve en la compuerta ORJ, la cual permite recibir el flujo por todas las ramas entrantes. En La Figura A-9 ilustra el patrón Multi Merge mediante las representaciones formales de redes de Petri y Grafos.

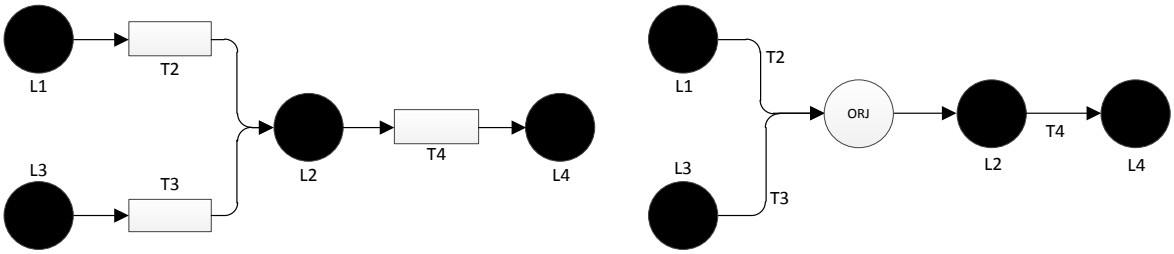


Figura A-9: Patrón Multi Merge representado mediante Grafos y Redes de Petri

A.3.9 Generalized And-join

Éste patrón permite la sincronización de dos o más ramas de entrada en una única rama de salida, independientemente del tiempo de ejecución que se tome cada rama.
Como se observa en la Figura A-10, el flujo diverge en T1 y posteriormente se sincroniza en T4; este comportamiento se observa de forma similar en la Figura A-11, donde la compuerta ANDS se encarga de la divergencia del flujo y la compuerta ANDJ se encarga de la sincronización.

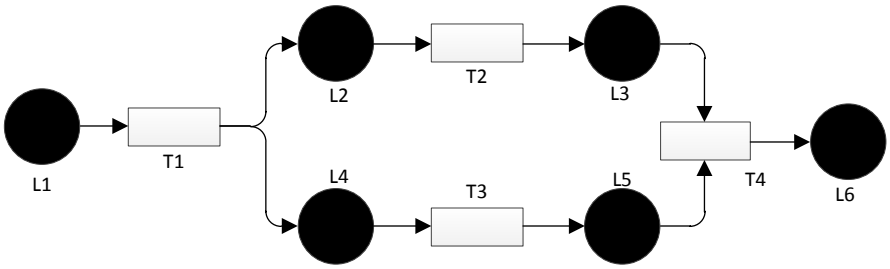


Figura A-10: Patrón Generalized AND-Join representado mediante Redes de Petri

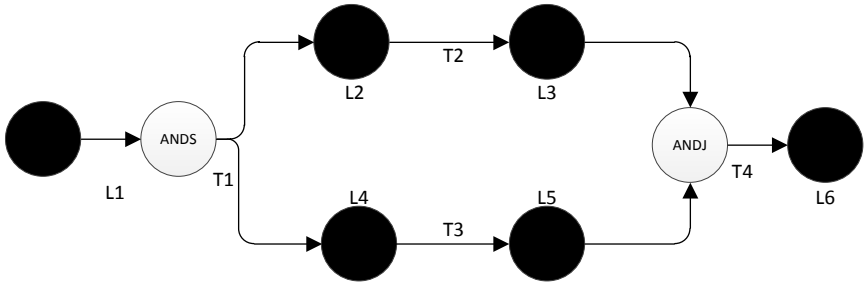


Figura A-11: Patrón Generalized AND-Join representado mediante Grafos

A.3.10 Local syncrhonizing merge

Este patrón es una variación de implementación del patrón *Structured Synchronizing Merge*. La diferencia está en que a lo largo de las ramas se realizan sincronizaciones parciales o locales de flujo. La Figura A-12 muestra como ocurren las sincronizaciones parciales o locales en los lugares L3, L4, L6 y L7; y la Figura A-13 muestra las compuertas necesarias para la representacion mediante grafos.

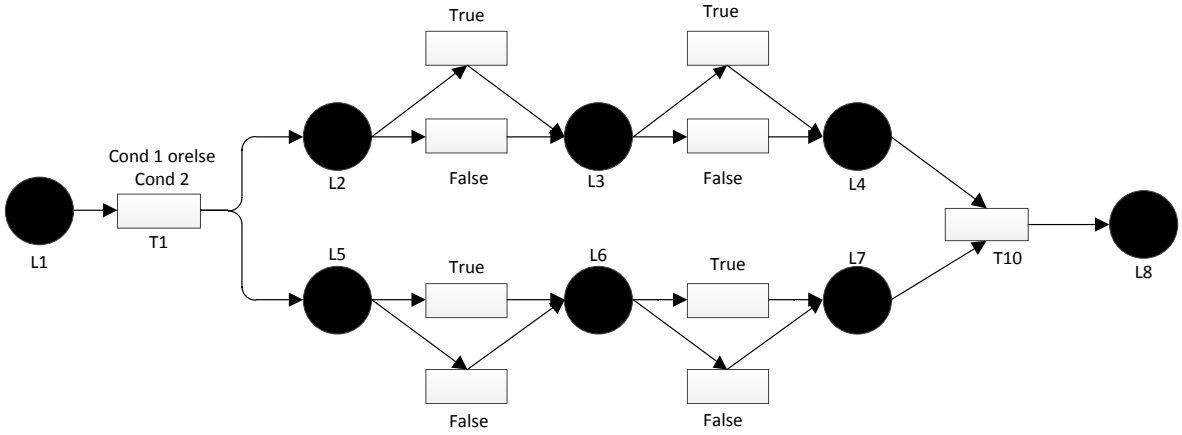


Figura A-12: Patrón Local Synchronizing Merge representado mediante Redes de Petri

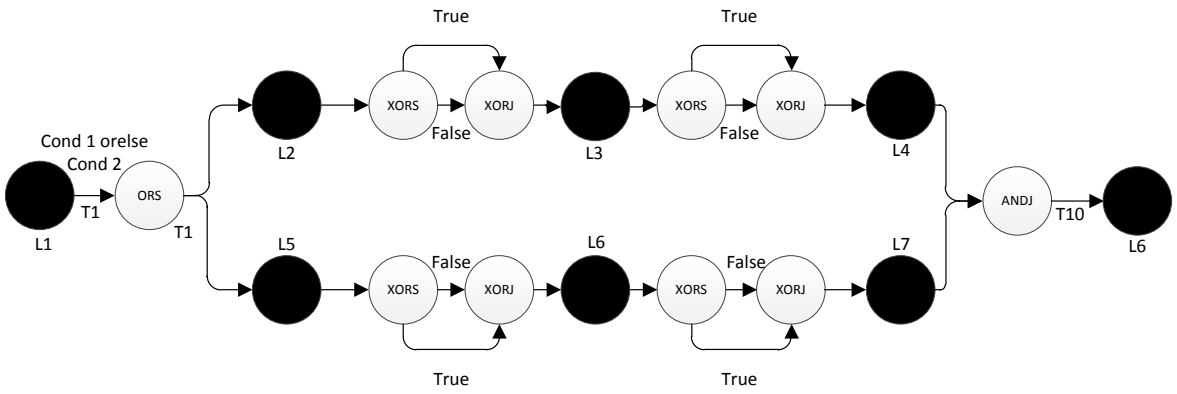


Figura A-13: Patrón Local Synchronizing Merge representado mediante Grafos

A.3.11 Deferred Choice

Este patrón diverge el flujo de control de una rama entrante en dos o más ramas salientes. Éste es clasificado dentro de la categoría: patrones de estado, ya que evalúa el flujo entrante para determinar por cual rama saliente continuara dicho flujo. La Figura A-14 muestra como en la red de Petri, L1 determina si continuará por la rama de la transición T2 o T3 y de forma similar, la compuerta XORS asegura que el flujo pase únicamente una de las dos ramas salientes.

Este patrón es utilizado para modelar los casos en los cuales eventos externos como: mensajes externos, datos de diferentes ambientes, temporizadores etc., afectan el Flujo de Control.

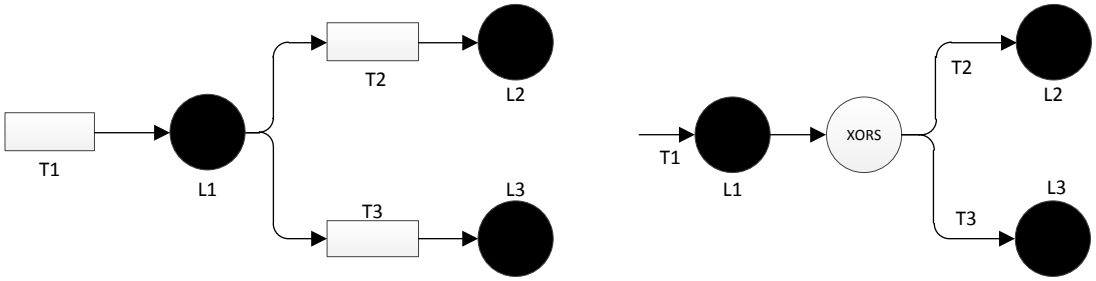


Figura A-14: Patrón Deferred Choice representado mediante Redes de Petri y Grafos

A.3.12 Critical Section

Este patrón diverge y converge el flujo de una manera especial. En la Figura A-15 se observa como una vez evaluada la transición T1 el flujo diverge de manera síncrona por ambas ramas, sin embargo los lugares y transiciones de la rama superior NO deben ejecutarse ni evaluarse simultáneamente a los lugares y transiciones de la rama inferior; este comportamiento se logra comunicando las secciones críticas por medio de temporizadores o multiplexores que controlen el tiempo en el cual son ejecutadas.
La Figura A-16 representa este patrón en grafos mediante las compuertas ANDS encargada de la divergencia del flujo de control de forma síncrona y una compuerta ANDJ encargada de sincronizar el flujo que proviene de las ramas entrantes.

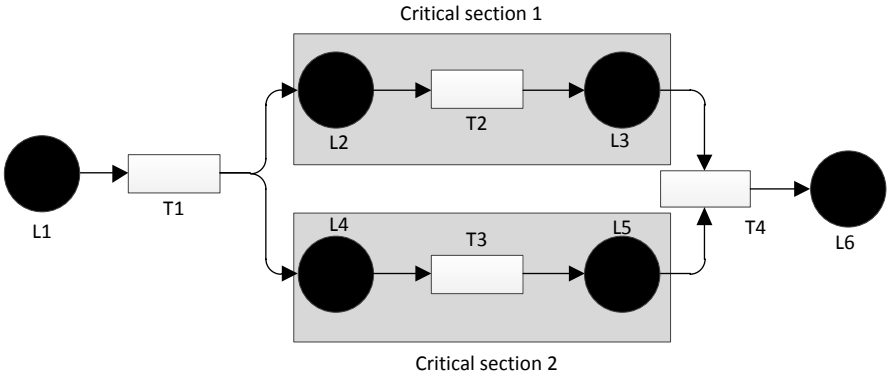


Figura A-15: Patrón Critical Section representado mediante Redes de Petri

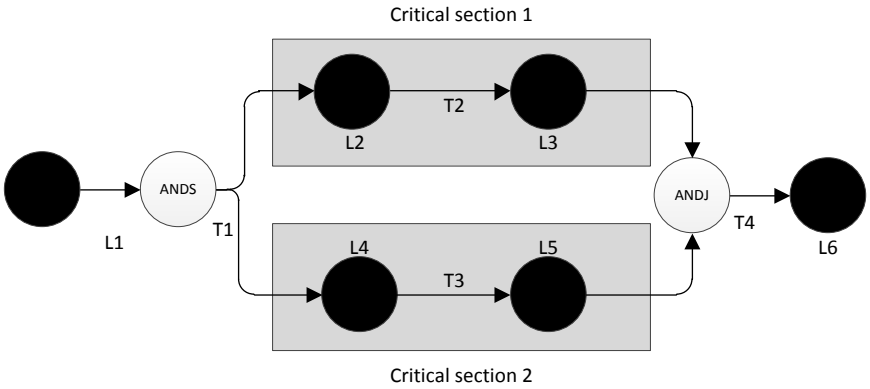


Figura A-16: Patrón Critical Section representado mediante Grafos

A.3.13 Interleaved Routing

Este patrón es muy similar al patrón *Critical Section*. La única diferencia es que no se presentan secciones críticas, pero si es necesario que cada una de las transiciones y lugares de las ramas se ejecute de manera aislada al resto de los lugares y transiciones de las demás ramas. En la figura Figura A-17, la transición T2 NO debe ejecutarse simultáneamente a la transición T3; en el momento en que T2 finalice su ejecución, T3 podrá iniciar. En la Figura A-18 se representa este patrón mediante las compuertas ANDS y ANDJ.

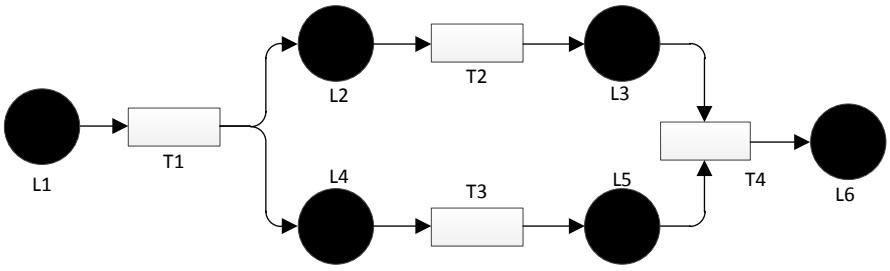


Figura A-17: Patrón Interleaved Routing representado mediante Redes de Petri

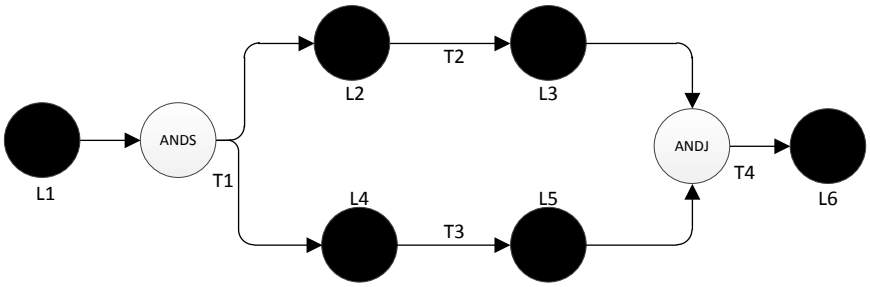


Figura A-18: Patrón Interleaved Routing representado mediante Grafos

A.3.14 Multiple instances without synchronization

Este patrón es utilizado para la creación de instancia de una tarea específica. Es representado en grafos mediante una compuerta XORS ya que según lo que ocurra en T1 es creada o no dicha instancia. La Figura A-19 representa este patrón.

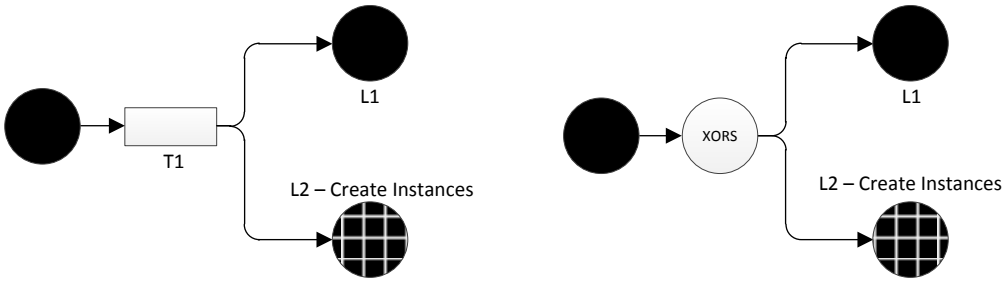


Figura A-19: Patrón Multiple Instances Without Synchronization representado mediante Redes de Petri y Grafos

A.3.15 Cancel Task

Este patrón representa la cancelación de una tarea dentro en el flujo de control. La Figura A-20 muestra el patrón representado mediante redes de Petri y grafos. Es importante aclarar que su representación puede realizarse mediante una compuerta XORJ, si la rama saliente es dirigida al nodo de cancelación; y mediante una compuerta XORS si una de sus ramas salientes se dirige hacia el nodo de cancelación.

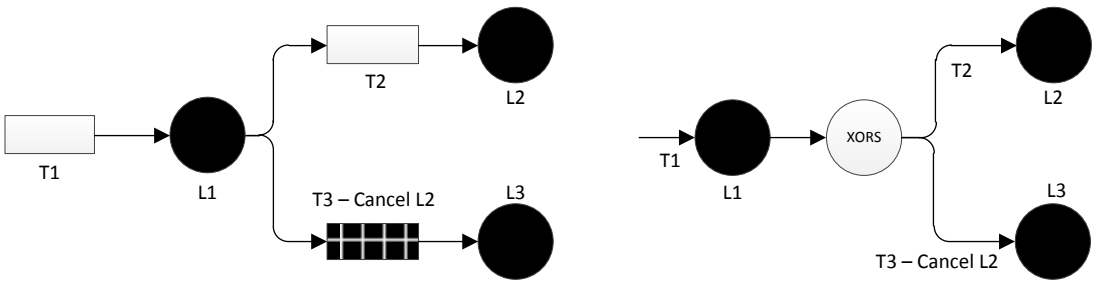


Figura A-20: Patrón Cancel Task representado mediante Redes de Petri y Grafos

A.3.16 Cancel Case

Este patrón representa la cancelación de todo el flujo de control. La Figura A-21 representa este patrón mediante redes de Petri y Grafos. La compuerta XORS diverge el flujo en dos ramas, si el flujo sigue por la rama inferior absolutamente todo el flujo es cancelado sin importar que tarea se está ejecutando.
Su representación puede variar mediante una compuerta XORJ, si su rama saliente se dirige al nodo “end”.

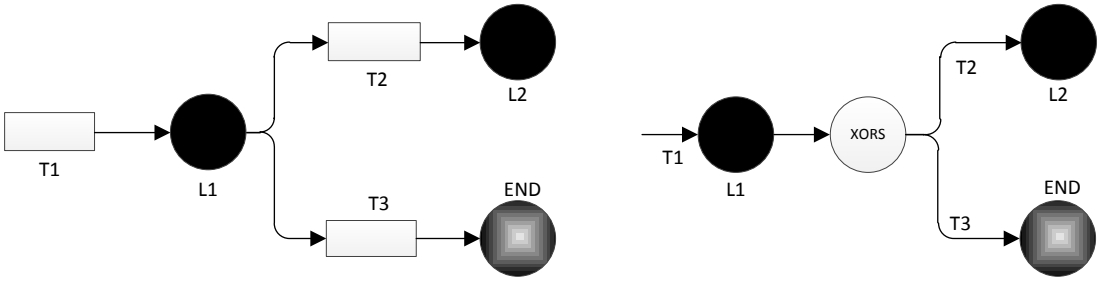


Figura A-21: Patrón Cancel Case representado mediante Redes de Petri y Grafos

A.3.17 Arbitrary Cycles

Este patrón representa ciclos repetitivos de tipo while o repeat presentes en el flujo de control. En la Figura A-22 se observa que L3 es el lugar que se puede ejecutar **n** cantidad de veces dependiendo de la condición evaluada en T3; si 1 y se cumple el flujo se devuelve nuevamente a T2 y L3 es nuevamente ejecutado, en caso de no cumplirse; no se hace el ciclo y el flujo termina en L6. La característica que diferencia este patron de “*Structured Loop*”, es la presencia de multiples entradas o salidas en el lugar que se va a ejecutar repetidamente, para el ejemplo L3.
Para su representación por medio de compuertas lógicas es necesario utilizar compuertas XORJ y XORS, cumpliendo las características nombradas, como se muestra en la Figura A-23.

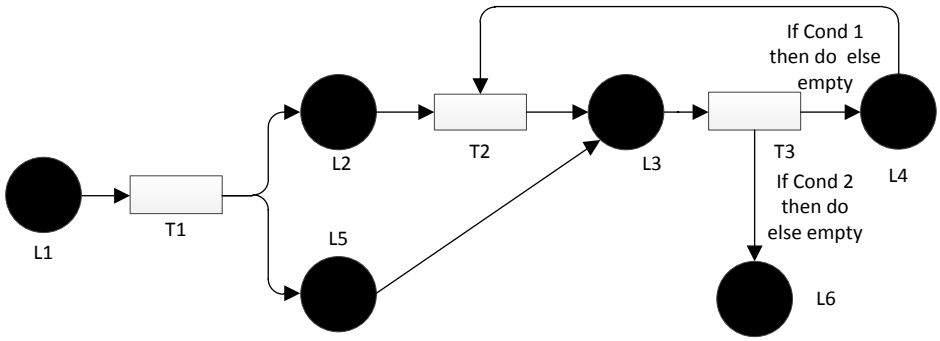


Figura A-22: Patrón Arbitrary Cicles representado mediante Redes de Petri

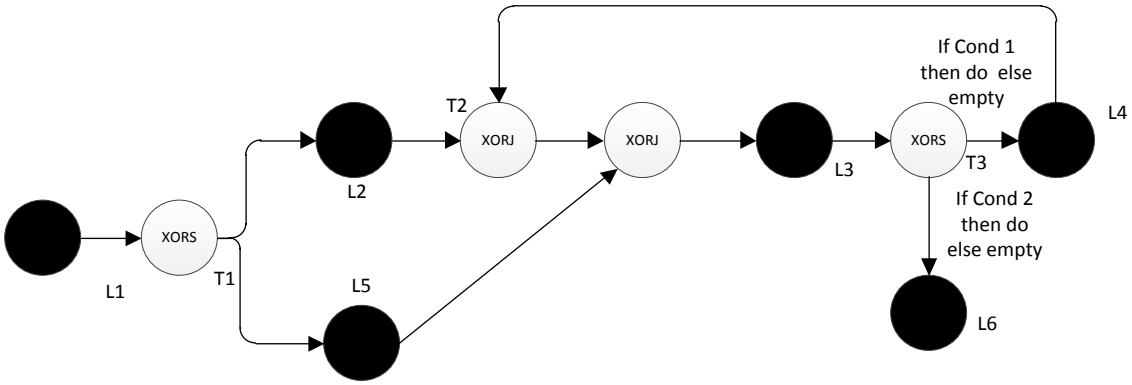


Figura A-23: Patrón Arbitrary Cicles representado mediante Grafos

A.3.18 Structured Loop

Este patrón es similar a *Arbitrary Cicles*, la diferencia está en que este no soporta múltiples entradas y salidas en el lugar que se va a ejecutar repetidamente, como se muestra en la Figura A-24. Este patrón presenta algunas variaciones dependiendo de la transición donde se evalúan las condiciones. Si la condición se evalúa antes del lugar que se repite, en este caso L3, se denomina *Structured loop pre-test,* en el caso contrario de denomina *Structured loop post-test.*
De forma similar al patrón *Arbitrary Cicles* su representación en grafos, como se muestra en la Figura A-25, se hace mediante compuertas XORJ y XORS.

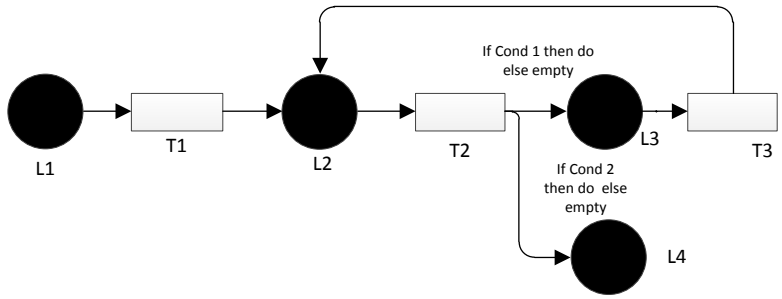


Figura A-24: Patrón Structured Loop representado mediante Redes de Petri

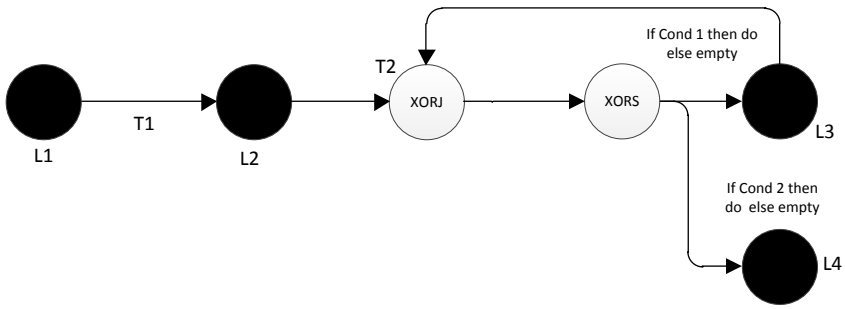


Figura A-25: Patrón Structured Loop representado mediante Grafos

A.3.19 Implicit Termination

Este patrón representa la ejecución de lugares y transiciones en un determinado en un orden determinado. Es similar *Critical Section*, la diferencia está en que permite realizar la ejecución de las tareas de las ramas superiores e inferiores en cualquier orden. Su representación mediante compuertas lógicas, como se muestra en la Figura A-26, se hace a través de ANDS la cual diverge el flujo en dos o más ramas.

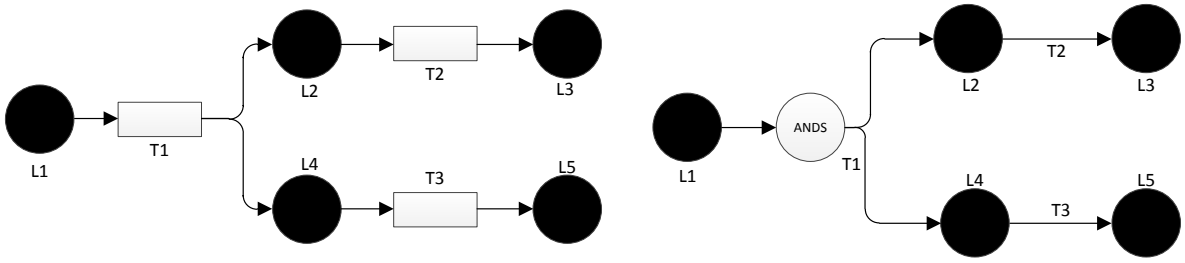


Figura A-26: Patrón Implicit Termination representado mediante redes de Petri y Grafos

A.3.20 Transient Trigger

Este patrón representa un disparador (trigger) que es producido de forma externa, el cual perdura un determinado tiempo. Este debe ser lanzado en un momento determinado cuando la tarea se está ejecutando. Su representación mediante compuertas lógicas, como se muestra en la Figura A-27, se hace a través de ANDJ, la cual converge el flujo de una rama y el flujo proveniente del lanzador.

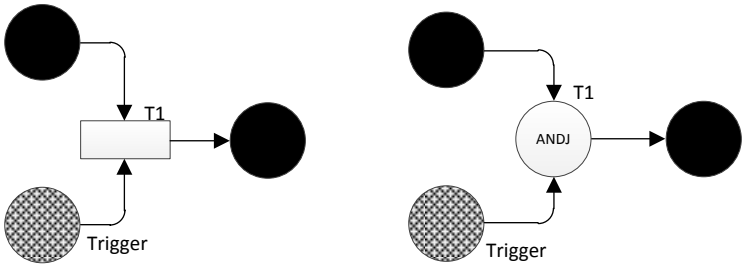


Figura A-27: Patrón Transient Trigger representado mediante Redes de Petri y Grafos

A.3.21 Persistent Trigger

Es una variación de *Transient Trigger*, con la diferencia de que cuando éste es lanzado perdura en el tiempo.

Ejemplo: En un proceso de ensamblaje de autos, se disparo una alarma en el sector de pintura, sin embargo, el auto apenas va en ensamblaje, por lo cual no se detiene el proceso de producción hasta que el carro llegue a pintura, cuando la alarma se desactive, el automóvil puede seguir su recorrido hacia otro sector de la fábrica.

A.4 Relación entre CFP y compuertas lógicas

A continuación se presenta una tabla, la cual relaciona patrones de flujo de control con las compuertas necesarias para su representación.

Tabla A-2: Relación entre patrones de Flujo de Control y compuertas lógicas necesarias para su representación

	Patrón	Compuertas	Comentarios
1	Sequence	-	
2	Parallel Split	ANDS	
3	Synchronization	ANDJ	
4	Exclusive Choice	XORS	
5	Simple Merge	XORJ	
6	Multi-choice	ORS	
7	Structured synchronizing merge	ORS + ANDJ	
8	Multi-merge	ORJ	
9	Generalized AND-join	ANDS + ANDJ	
10	Local syncrhonizing merge	ORS + ANDJ	Se necesita sincronizaciones parciales o locales del flujo mediante compuertas XORJ y XORS. Este patrón es una variación de representación del *Structured synchronizing merge*
11	Deferred Choice	XORS	
12	Critical Section	ANDS + ANDJ	
13	Interleaved Routing	ANDS +ANDJ	
14	Multiple instances without synchronization	XORS	
15	Cancel Task	XORS ó XORJ	Una rama, debe ir a un nodo de cancelación. La rama de destino de XORJ debe ir al nodo de cancelación
16	Cancel Case	XORS o XORJ	Una rama debe ir a un nodo de fin La rama de destino de XORJ debe ir al fin
17	Arbitrary Cycles	XORS + XORJ	Una rama de la compuerta XORS es dirigida a la compuerta XORJ.
18	Structured Loop	XORS +XORJ	Una rama de la compuerta XORS es dirigida a la compuerta XORJ.
19	Implicit Termination	ANDS	
20	Transient Trigger	ANDJ	
21	Persistent Trigger	ANDJ	

Anexo B

B Patrones de Datos

B.1 Introducción

Éste anexo presenta en la Seccion B.2 la evaluación realizada por Vander Aalst *et al* [3]de los patrones de datos (DP) soportados por JBPM y BPEL y en la Seccion A.3, la representación formal en redes de Petri y grafos de los patrones soportados por ambos lenguajes.

B.2 Evaluación de DP soportados por JBPM y BPEL

Patrones de Datos	**Viabilidad de Implementación***	
Data Visibility	BPEL	JBPM
Task data	+/-	+/-
Block data	-	-
Scope Data	+	-
Multiple Instance Data	-	-
Case Data	+	+
WorkFlow Data	-	-
Environment Data	+	+/-
Internal Data Interaction		
Data Interaction between Tasks	+	+
Data Interaction - Block Task to Sub-workflow Decomposition	-	-
Data Interaction - Sub-workflow Decomposition to Block Task	-	-
Data Interaction - to Multiple InstanceTask	-	-
Data Interaction - from Multiple Instance Task	-	-
Data Interaction - Case to Case	+/-	+/-
External Data Interaction		
Data Interaction - Task to Environment - Push-Oriented	+	+/-
Data Interaction - Environment to Task - Pull-Oriented	+	+/-
Data Interaction - Environment to Task - Push-Oriented	+/-	+/-
Data Interaction - Task to Environment - Pull-Oriented	+	-
Data Interaction - Case to Environment - Push-Oriented	-	-
Data Interaction - Environment to Case - Pull-Oriented	-	-
Data Interaction - Environment to Case - Push-Oriented	-	-
Data Interaction - Case to Environment - Pull-Oriented	-	-
Data Interaction - Workflow to Environment - Push-Oriented	-	-

Data Interaction - Environment to Workflow - Pull-Oriented Data Interaction - Environment to Workflow - Push-Oriented Data Interaction - Workflow to Environment - Pull-Oriented	- - -	- - -
Data Transfer		
Data Passing by Value – Incoming Data Passing by Value - Outgoing Data Passing - Copy In/Copy Out Data Passing by Reference - Unlocked Data Passing by Reference - Locked Data Transformation - Input Data Transformation – Output	+ + - + +/- - -	- - + - - + +
Data Routing Patterns		
Task Precondition - Data Existence Task Precondition - Data Value Task Postcondition - Data Existence Task Postcondition - Data Value Event-based Task Trigger Data-based Task Trigger Data-based Routing	+/- + - - + +/- +	- - - - - - +/-
* **+** Si lo soporta; **+/-** Lo soporta con limitantes; **–** No lo soporta		

La lista de los patrones que soportan ambos lenguajes se resume en la Tabla B-1.

Tabla B-1: Patrones soportados por JBPM o BPEL

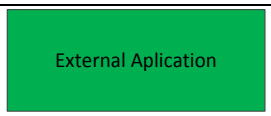
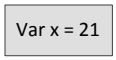
	Patrón
1	Scope Data
2	Case Data
3	Environment Data
4	Data Interaction between Tasks
5	Data Interaction - Task to Environment - Push-Oriented
6	Data Interaction - Environment to Task - Pull-Oriented
7	Data Interaction - Task to Environment - Pull-Oriented
8	Data Passing by Value - Incoming
9	Data Passing by Value - Outgoing
10	Data Passing - Copy In/Copy Out
11	Data Passing by Reference - Unlocked
12	Data Transformation – Input
13	Data Transformation – Output
14	Task Precondition - Data Value
15	Event-based Task Trigger
16	Data-based Routing

B.3 Descripción de los DP seleccionados

Los patrones de datos tienen como objetivo capturar las diversas formas en la cual, los datos son transmitidos entre tareas[1] y entornos externos, independientemente de la tecnología de implementación.

Para la representacion de los 16 DP soportados por ambos lenguajes, se presenta la convención que se muestra en la Tabla B-2**.**

Tabla B-2: Convención para los patrones de Datos

Grafico	Descripción
	Lugar en redes de Petri, representa un estado en el Flujo de Control.
	Transicion en redes de Petri, representa eventos que activan o disparan lugares.
	Arco en redes de Petri, representa la conexión entre lugares y transiciones.
	Flujo de datos entre transiciones y lugares
	Flujo de datos entre lugares, transiciones y aplicaciones externas.
	Flujo de datos entre lugares, transiciones y aplicaciones externas.
	Aplicación externa al Flujo de Control (servicios web, repositorios de datos, etc.)
	Éste rectángulo indica el valor de las variables para un único lugar o transición, o para un grupo de lugares y transiciones.

B.3.1 Scope Data

Éste patrón representa la definición de una variable para un conjunto de lugares y transiciones. En la Figura B-1 se observa que solo L1 y T1 hacen uso de la variable X y ninguno de los otros lugares y transiciones pueden hacer uso de ella.

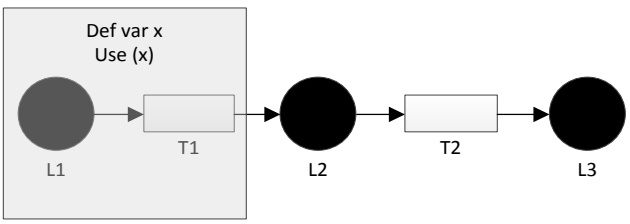


Figura B-1: Patrón Scope Data

[1] Tareas: Se refiere específicamente al Workflow.

B.3.2 Case Data

Este patrón representa la definición y el uso de datos para todo el flujo, es decir, todos los lugares y transiciones utilizan datos establecidos previamente. LaFigura B-2 representa el patrón Case Data.

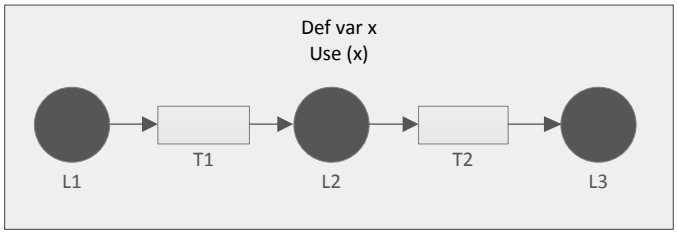


Figura B-2: Patrón Case Data

B.3.3 Environment Data

Los lugares y transiciones hacen uso de datos los cuales son establecidos por entornos externos. La figura Figura B-3 representa el patrón Environment Data.

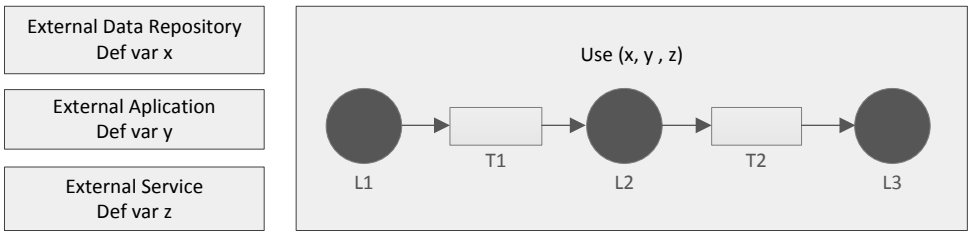


Figura B-3: Patrón Environment Data

B.3.4 Data Interaction between Tasks

Este patrón representa el paso de datos entre lugares y transiciones continuas como es el caso de L1 y T1; y el paso de datos entre lugares-lugares como es el caso de L2 - L3. Lo más importante es que soporta el intercambio de datos sin tener en cuenta el flujo de control. La Figura B-4 representa el patron task to task.

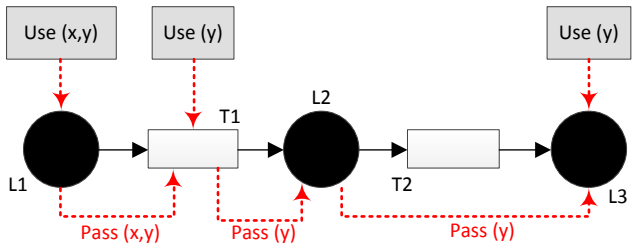


Figura B-4: Patrón Data Interacion between Tasks

B.3.5 Data Interaction - Task to Environment - Push-Oriented

Este patrón representa el intercambio de datos entre un proceso externo al flujo. En el ejemplo que se muestra en la Figura B-5, las transiciones y lugares al ejecutarse establecen el valor de las variables m,z y j en una aplicación externa.

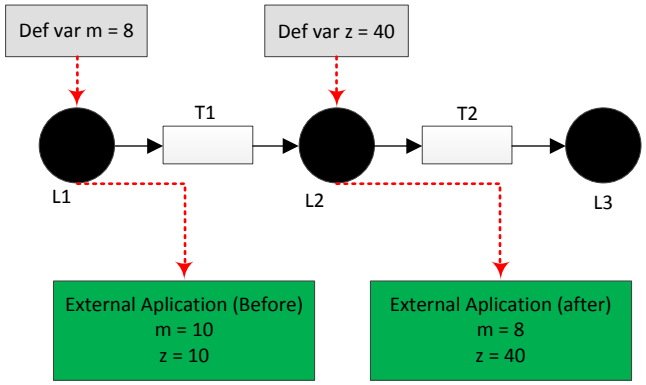


Figura B-5: Patron Data Interaction - Task to Environment – Push - Oriented

B.3.6 Data Interaction - Environment to Task - Pull-Oriented

Este DP es similar al patron *Task to Environment – Push Oriented,* la diferencia está en que la aplicación externa pasa los datos a los lugares y transiciones y puede cambiar las variables que ya estaban predefinidas. La Figura B-6 muestra el paso de datos entre la aplicación externa y los lugares L1 y L2 a través de peticiones y respuestas (request & response).

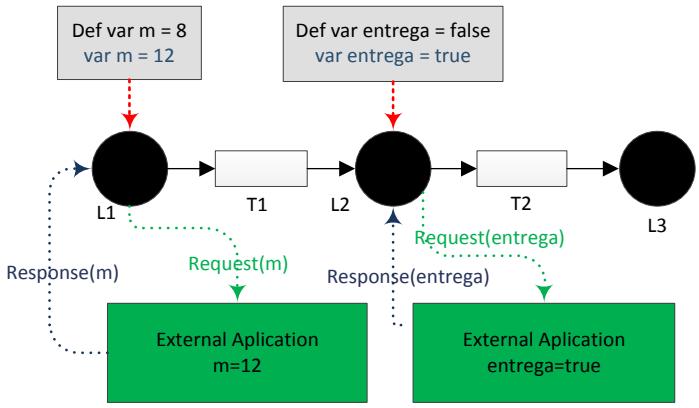


Figura B-6: Patron Data Interaction - Task to Environment – Push - Oriented

B.3.7 Data Interaction - Task to Environment - Pull-Oriented

Es un patrón que permite el intercambio de datos mediante peticiones y respuestas entre los lugares y transiciones de un determinado proceso, y cualquier aplicación externa. La diferencia con los patrones *Task to Environment – Push Oriented y Environment to Task – Pull Oriented* es que permite el paso de datos sin tener en cuenta la definición de las variables. La Figura B-7 muestra un ejemplo del intercambio de datos entre L1, L2 y algunas aplicación externas, para el caso de L1, la aplicación externa hace una petición a éste para que le pase el valor de la variable que está definida, en este caso m, por esto la aplicación externa establece el valor de j en con el mismo valor de m. De forma similar ocurre con L2

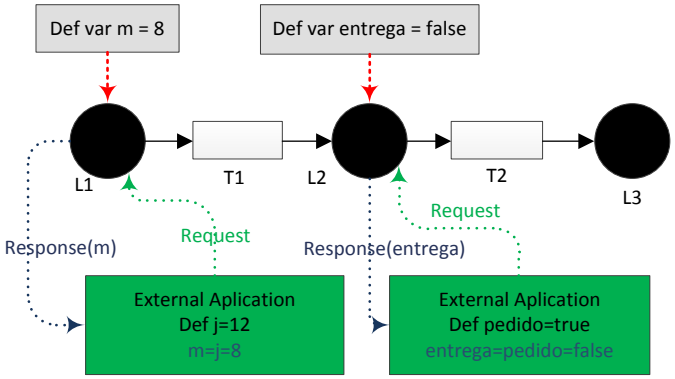


Figura B-7: Patrón Data Interaction -Task to Environment - Pull-Oriented

B.3.8 Data Passing by Value - Incoming & Outgoing

En este patrón se representa la transferencia de datos por valor. La Figura B-8 muestra un ejemplo en donde todas las variables (x,y,z) son establecidas en 21.

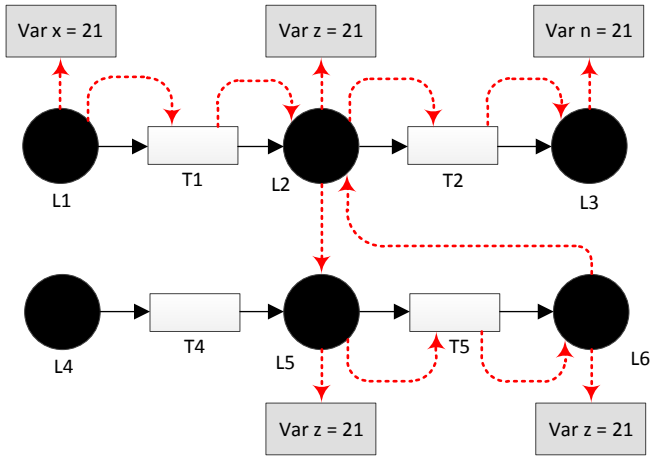


Figura B-8: Patrón Data Passing by Value – Incoming & Outgoing

B.3.9 Data Passing - Copy In/Copy Out

Este patrón representa el paso de datos entre nodos y transiciones y una aplicación externa con características de almacenamiento de datos, por ejemplo, bases de datos y repositorios. La figura Figura B-9 muestra como L1, L2 y L3 obtienen los datos de una base de datos.

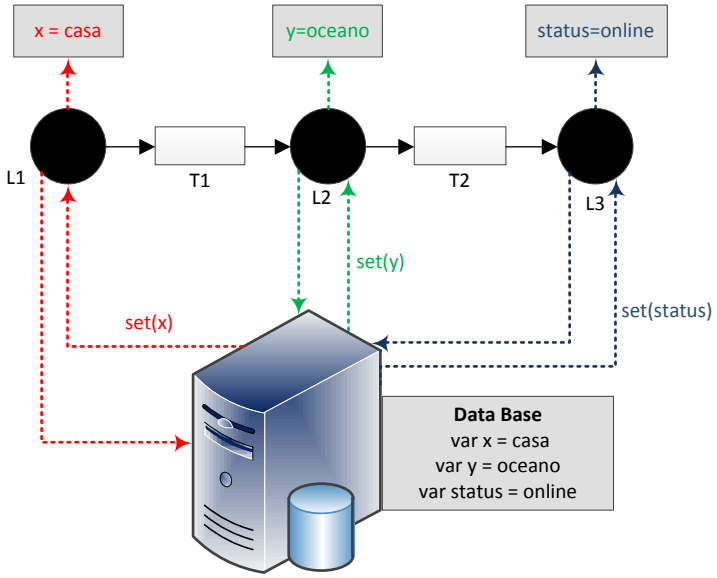


Figura B-9: Patrón Data Passing - Copy In/Copy Out

B.3.10 Data Passing by Reference – Unlocked

Éste patrón representa la transferencia de datos de entornos externos entre lugares y transiciones. Es similar al patrón case data, sin embargo, la diferencia está en que los datos se tienen que sincronizar, es decir, el flujo de control es tenido en cuenta para el intercambio de datos. La Figura B-10 ilustra el patrón Data Transfer by Reference.

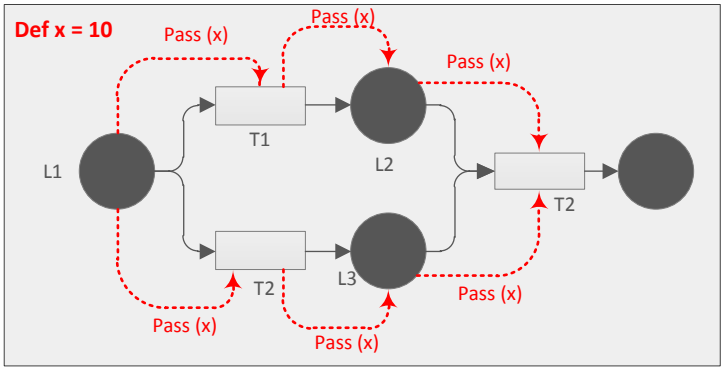


Figura B-10: Patrón Data Passing by Reference – Unlocked

B.3.11 Data Transformation – Input & Output

Este patrón representa la transformación que van sufriendo los datos a medida que estos son transferidos entre lugares y transiciones. La Figura B-11 muestra como la variable x sufre continuas transformaciónes. La diferencia entre el patrón *Data Transformation Input* y O*utput*, simplemente se relaciona con el lugar donde se realiza la transformación; si es a la entrada de un lugar o transición, se dice que es de tipo *Input*, en caso contrario, si se realiza al finalizar la lógica de implementación de cada elemento, se dice que es una transformación de tipo *Output*.

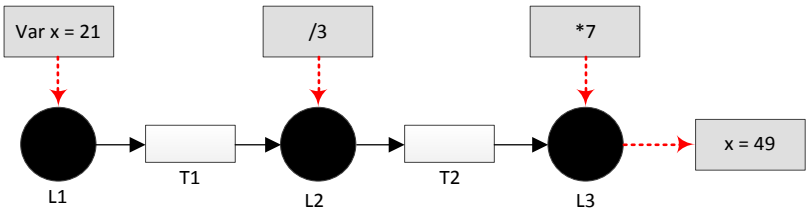


Figura B-11: Patrón Data Transformation – Input & Output

B.3.12 Task Precondition - Data Value

Este patrón representa un condicional previo existente el cual se tiene que cumplir para que el flujo pueda continuar. Para el ejemplo de la Figura B-12 el flujo puede continuar a L2 si se cumple la condicion x > 10.

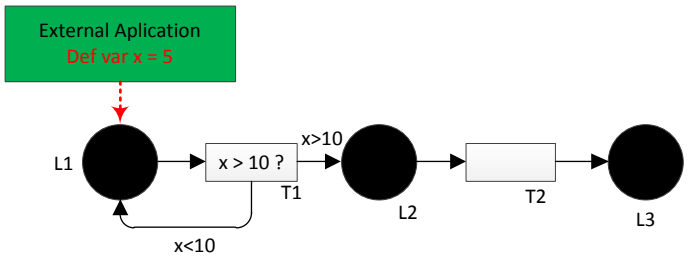


Figura B-12: Patrón Task Precondition - Data Value

B.3.13 Event-based Task Trigger

Este patrón representa el disparo de las actividades del flujo mediente actividades o ambientes externas, como por ejemplo: servicios web, aplicaciones de terceros, etc. La Figura B-13 representa el patron Event – based task Trigger.

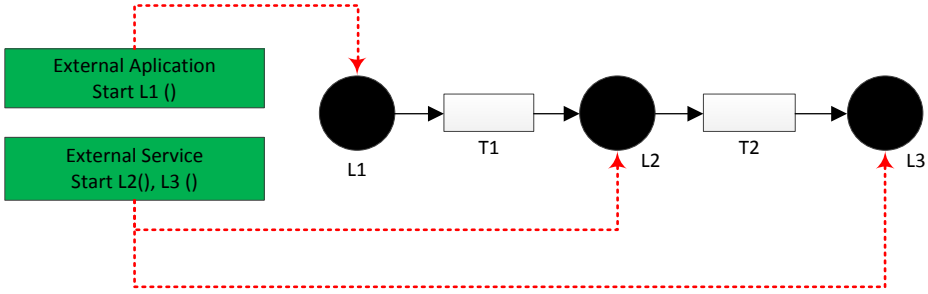


Figura B-13: Patrón Event-based Task Trigger

B.3.14 Data-based Routing

Este patrón tiene especial relación con los CFP representados mediante compuertas lógicas de tipo ORS y XORS, como lo son: *MultiChoice, Exclusive Choice,* etc. ya que la rama por la cual va a continuar el flujo, depende de los datos establecidos previamente. La Figura B-14, representa el patrón Data – based Routing.

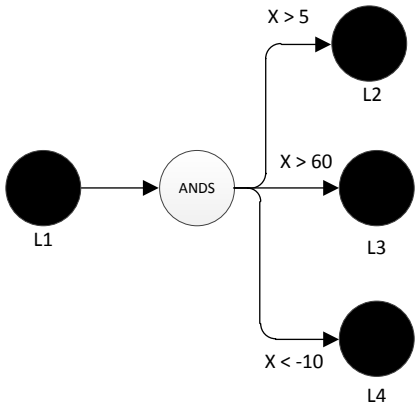


Figura B-14: Patrón Data-based Routing

Anexo C

C Servicios Modelados

C.1.1 Servicios Telco

C.1.1.1 Servicios Básicos

C.1.1.1.1 Llamada Básica

El servicio básico de llamada representado mediante Redes de Petri (Figura C-1) está compuesto por tres Lugares: “Ring”, “Talking” y “Busy Recording”; y cinco transiciones: “invite”, “busy”, “time out”, “bye” y “200 OK”. Para describir el funcionamiento del servicio se identifican tres usuarios: A, B y C. El servicio inicia en el estado Idle, cuando A realiza una invitación (Invite) al usuario B, en ese momento pasa al lugar “Ring”, donde puede ocurrir tres posibilidades: **I)** el usuario B se encuentra como ocupado y no puede contestar la llamada; **II)** después de un tiempo determinado B no responde la invitación a hablar; y **III)** Los usuarios A y B se comunican satisfactoriamente. En **I**, se dispara la transición “Busy”, que activa el lugar “Busy Recording” donde el usuario A escucha una grabación que le indica la imposibilidad de contestar por parte de B, una vez finalizada la grabación se activa la transición “Bye” y el servicio regresa a su estado Inicial. En **II**, el servicio pasa a la transición “time out”, ya que ha transcurrido un tiempo determinado en el lugar “Ring” y la invitación a hablar no ha sido respondida, lo cual, finaliza el servicio. Finalmente, en **III**, la transición “200 OK”, indica que la invitación a hablar es contestada satisfactoriamente, activando así, el lugar “Talking”, donde A y B se comunican; en este lugar puede llegar una invitación a hablar por parte de C, ya sea a A o B, sin embargo, la llamada no puede ser contestada. En las secciones siguientes del documento, la inclusión de servicios suplementarios al servicio básico de llamada, permitirá al usuario A o B contestar la invitación de C.

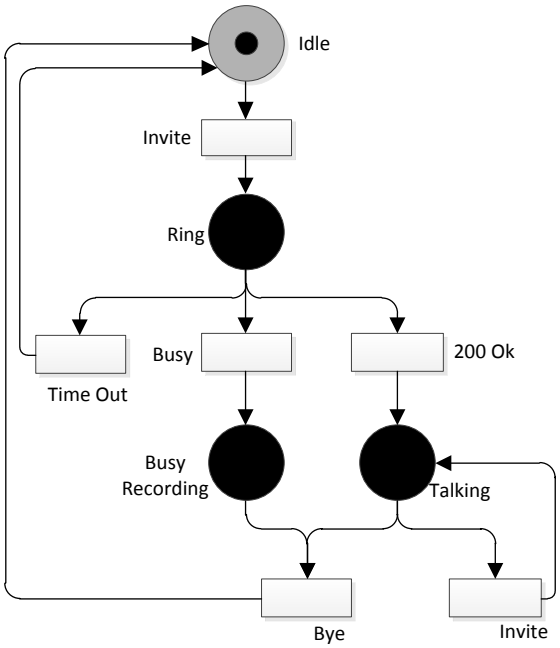


Figura C-1: Servicio básico de llamada representado mediante redes de Petri

Este mismo servicio también fue modelado en grafos, como se muestra en la Figura C-2, teniendo en cuenta la transformación de redes de Petri a Grafos; el funcionamiento del servicio es idéntico.

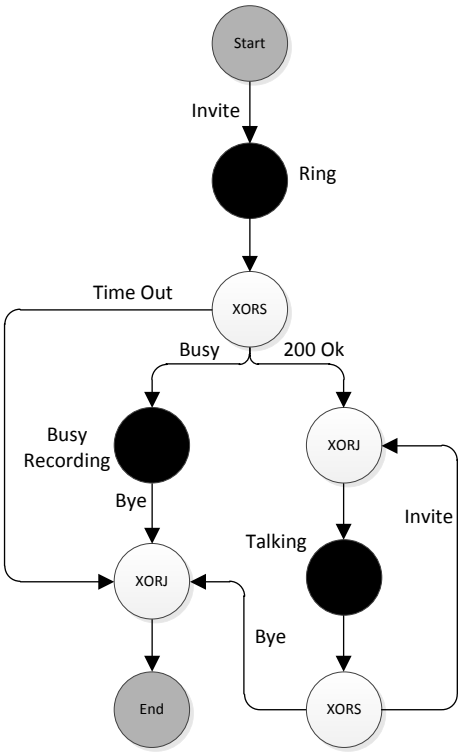


Figura C-2: Servicio básico de llamada representado mediante Grafos

C.1.1.1.2 SMS

El servicio de Mensajeria corta (SMS Short Message service) permite a los usuarios de telefonía enviar cadenas de texto de un tamaño determinado.

Para realizar el modelado de este servicio, no se tienen en cuenta aspectos técnicos propios de la infraestructura de red necesaria, para que el mensaje llegue a su destino, como por ejemplo el SMSC (Short Message Service Center) el cual se encarga de recibir el mensaje que viene del dispositivo que envía el mensaje y luego de realizar algunas verificaciones propias de cada operador, envía el mensaje al destino.

La Figura C-3 y Figura C-4 ilustran el modelado en redes de Petri y Grafos respectivamente, teniendo en cuenta una solicitud entrante de envió de mensaje representado mediante la transición/arista “Solicitud SMS” la cual activa el lugar/Nodo “Estado del terminal”, encargado de verificar si el dispotivo de destino se encuentra alcanzable o no alcanzable por la red de comunicaciones, para enviar el mensaje. Adicionalmente se representa el periodo de validez que tiene un SMS, ya que si el terminal no se encuentra disponible, el mensaje perdurara por un tiempo y luego caducara según las políticas de cada operador. El servicio termina cuando el mensaje ha llegado a su destino o cuando el tiempo de validez del mensaje expira.

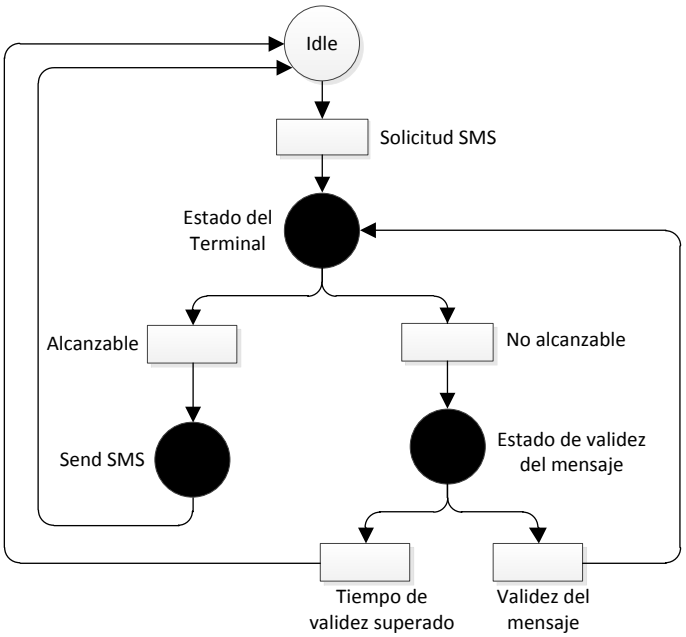


Figura C-3: Servicio de mensajería corta SMS representado mediante redes de Petri

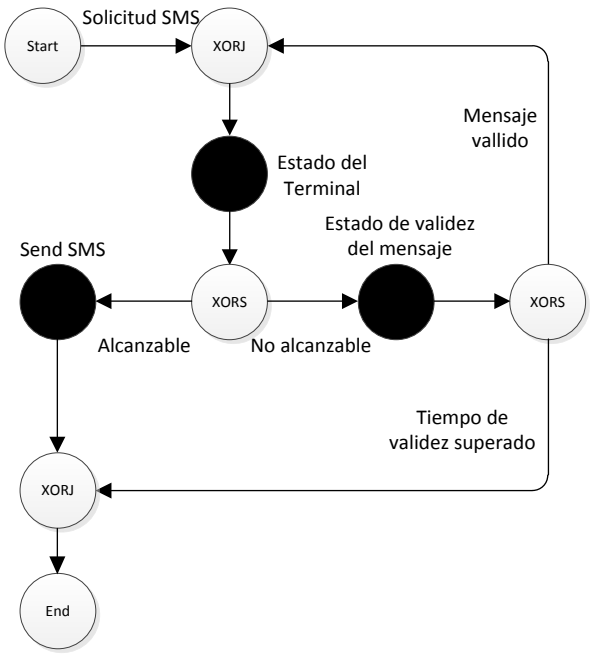


Figura C-4: Servicio de mensajería corta SMS representado mediante Grafos

C.1.1.2 Servicios Suplementarios

Como se explicó en el Capitulo 3 de la Monografia, los servicios suplementarios como desvío de llamada, transferencia de llamada, etc., deben estar acompañados de servicios básicos, para que puedan ser ofrecidos a los usuarios; por esta razón, la combinación entre servicios básicos y suplementarios se explica en la Sección C.1.1.3.

C.1.1.3 Servicios Compuestos

Los servicios compuestos son el resultado de las posibles combinaciones entre servicios básicos y suplementarios. A continuación se presenta un conjunto de servicios compuestos, algunos de estos resultan de incluir uno o más servicios suplementarios con uno o más servicios compuestos.

C.1.1.3.1 Llamada + desvío incondicional

En servicio *llamada + desvío incondicional,* que se muestra en la Figura C-5 a) y b) representado mediante redes de Petri y Grafos respectivamente, se identifican cuatro usuarios A, B, C y D. El servicio comienza cuando el usuario A realiza una invitación a hablar a B, si este tiene activado el servicio de *desvío de llamada,* el usuario A escuchara una grabación que le notificara que su llamada está siendo redirigida, en este caso al usuario C.

Una vez la llamada es redirigida, el servicio se comporta de forma idéntica al servicio de *llamada básica* de la Seccion: C.1.1.1, donde el usuario D es aquel que trata de comunicarse con A o C, pero ninguno de estos puede responder su invitación a hablar.

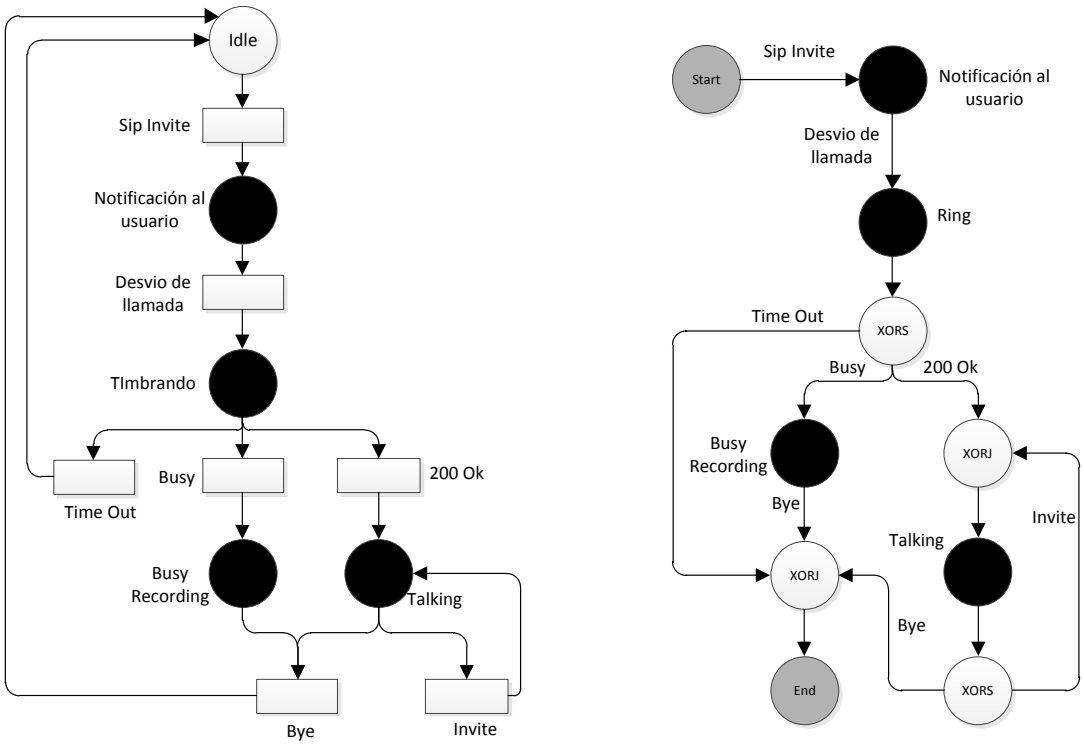


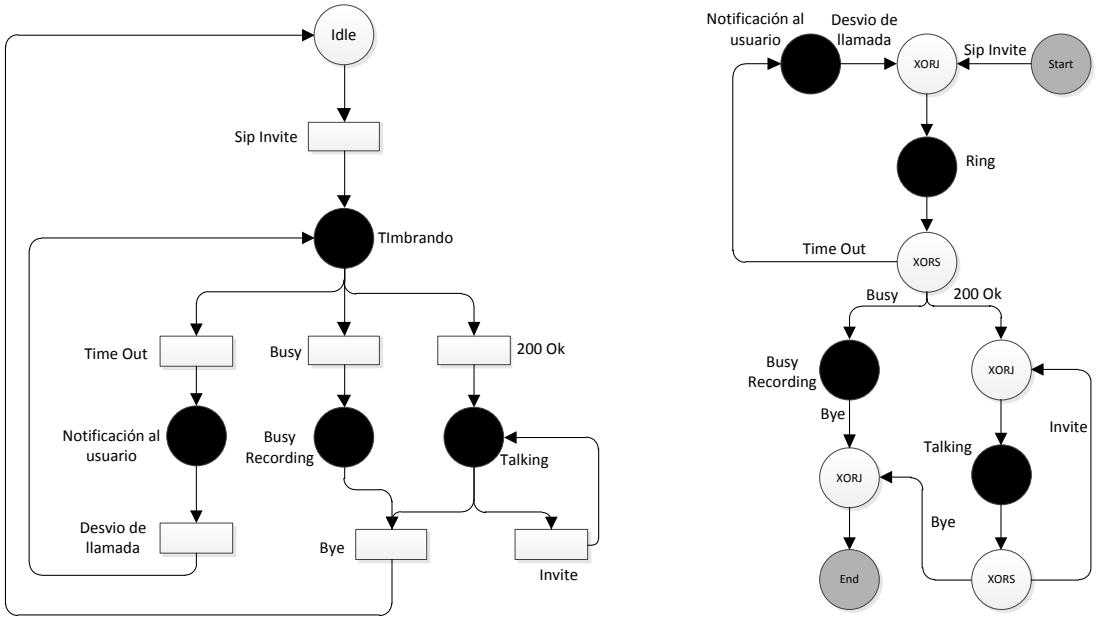
Figura C-5: Servicio de llamada + desvío incondicional representado mediante a) Grafos b) redes de Petri

C.1.1.3.2 Llamada + Desvío en caso de no respuesta (CFNR, Call Forwarding – No Reply)

En este servicio se identifican 3 usuarios A, B y C. A realiza una invitación a hablar a B, sin embargo, B no contesta la invitación, por esta razón el servicio redirige la llamada a un usuario C. El usuario C puede estar configurado ya sea como número fijo o un buzón de mensaje. La Figura C-6, muestra la representación del servicio en redes de Petri y grafos.

C.1.1.3.3 Llamada + Llamada en Espera (Call Hold)

En este servicio se identifican tres usuarios: A, quien realiza la primera invitación a hablar; B, al cual el usuario A desea llamar; y C, quien realiza una invitación a hablar a A.
La Figura C-7 y Figura C-8 muestran la representación del servicio en redes de Petri y Grafos respectivamente; éste inicia cuando A, hace un Sip_Invite a B, si este no contesta el servicio finaliza, pero si este contesta, A y B pasan a establecer comunicación. Cuando estos dos usuarios se encuentran hablando puede llegar una invitación a hablar de C, en este caso A tiene la posibilidad de contestar la llamada a C. si esto ocurre, A deja en espera (Hold) a B y establece comunicación con C, sin embargo si B todavía se encuentra disponible, A puede restablecer la comunicación con B dejando en espera a C.
Si A está hablando con B y tiene en espera a C, puede restablecer la comunicación con C en cualquier momento, sin embargo, si C y B cuelgan el servicio finaliza. Si A está hablando con C, y B está en espera, A puede restablecer la comunicación con B en cualquier momento, sin embargo, si C y B se desconectan el servicio finaliza. Por otra parte si A se desconecta sin importar si tiene en espera a B o C, el servicio finaliza ya que el A siempre tiene que estar presente en la comunicación.



a) b)

Figura C-6: Llamada + Desvío en caso de no respuesta (CFNR, Call Forwarding – No Reply) representado mediante redes de Petri a) y Grafos b)

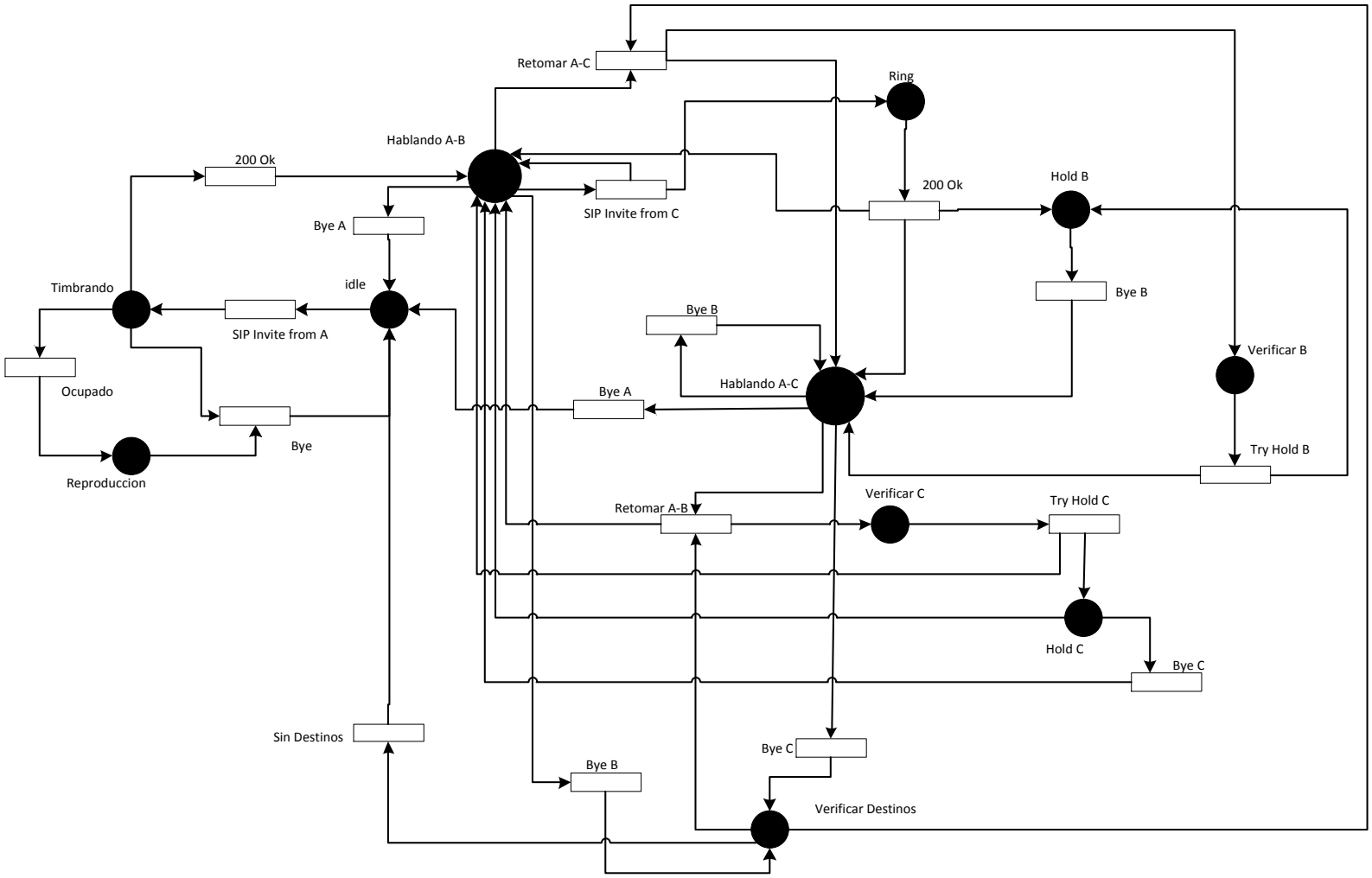


Figura C-7 Servicio de Llamada + Llamada en Espera (Call Hold) representado mediante redes de Petri

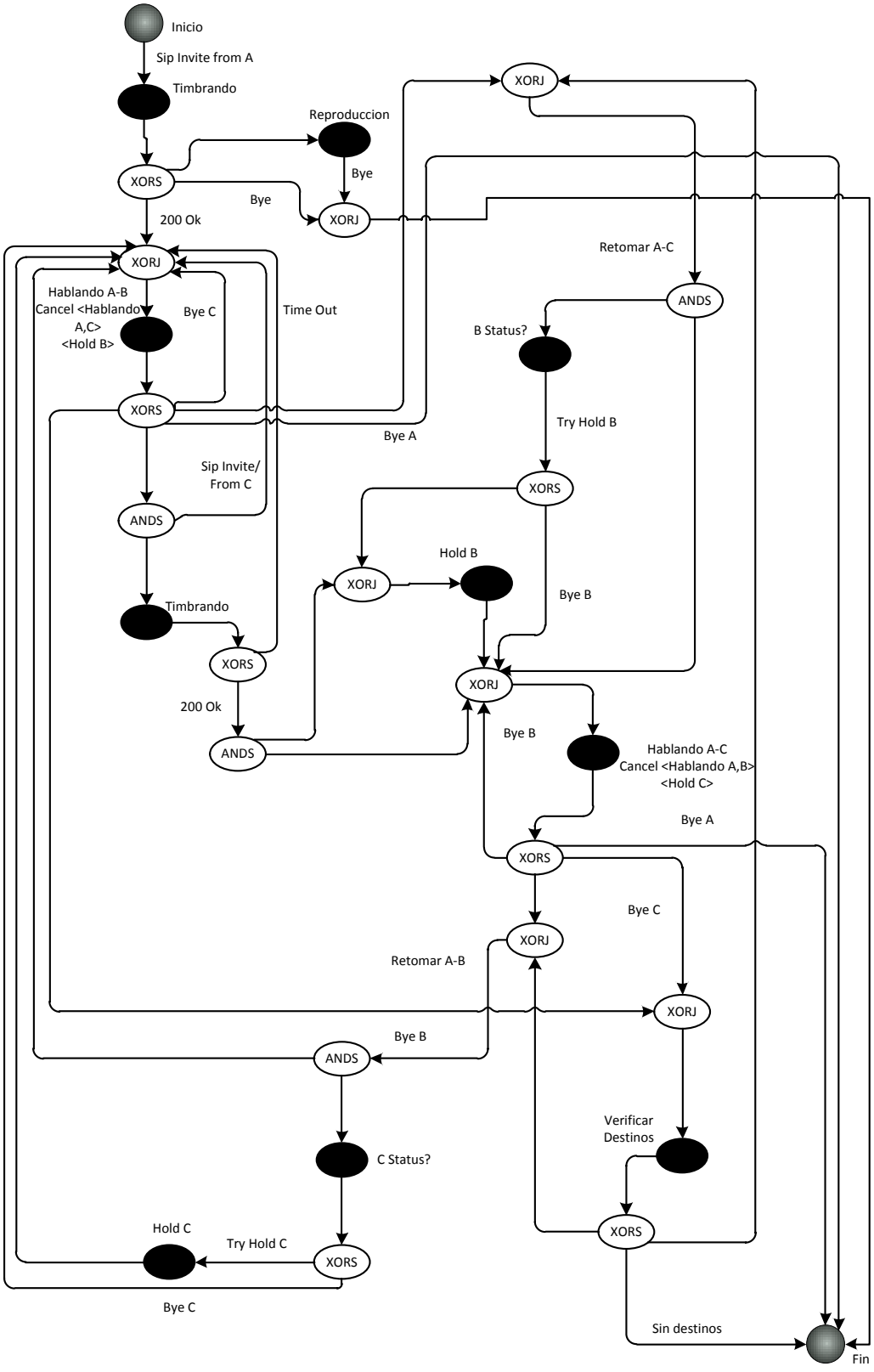


Figura C-8: Servicio de Llamada + Llamada en Espera (Call Hold) representado mediante Grafos

C.1.1.3.4 Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer)

Este servicio se diferencia del servicio de *llamada + llamada en espera*, ya que permite a B y C establecer comunicación. La única condición es que A tiene que ser la parte que se encarga de comunicar a B y C, una vez establecida la llamada entre estos, A puede abandonar el servicio sin que las partes B y C se desconecten. La Figura C-9 y Figura C-10 muestra la representación del servicio en redes de Petri y Grafos respectivamente, las líneas punteadas (---) indican la diferencia que existe ente el servicio de llamada en espera (Figura C-7 y Figura C-8) y el servicio de *llamada en espera + transferencia de llamada.*

C.1.1.3.5 Llamada + Llamada en Espera (Call Hold) + Conference

Este servicio tiene las capacidades del servicio de *llamada + llamada en espera*, la diferencia está en que tanto A, B y C pueden establecer comunicación simultáneamente. La representación de este servicio mediante redes de Petri y Grafos se muestra en la Figura C-11 y Figura C-12 respectivamente donde la línea punteada (…) indica la diferencia que existe ente el servicio de *llamada + llamada en espera* y el servicio de *llamada + llamada en espera + conferencia.*

C.1.1.3.6 Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) Conference

El servicio representado mediante redes de Petri y Grafos en la Figura C-13 y Figura C-14 respectivamente, reúne las capacidades de los servicios mostrados en la Figura C-7, Figura C-9 y Figura C-11, ya que tanto A B y C pueden establecer comunicación, de la misma forma B y C sin necesidad de que A este presente. Las líneas punteadas (…) muestran el servicio de conferencia, las líneas punteadas (---) muestran el servicio de transferencia de llamada y las líneas continuas representa el servicio de llamada en espera. En conjunto representan el servicio compuesto de *Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) + Conference.*

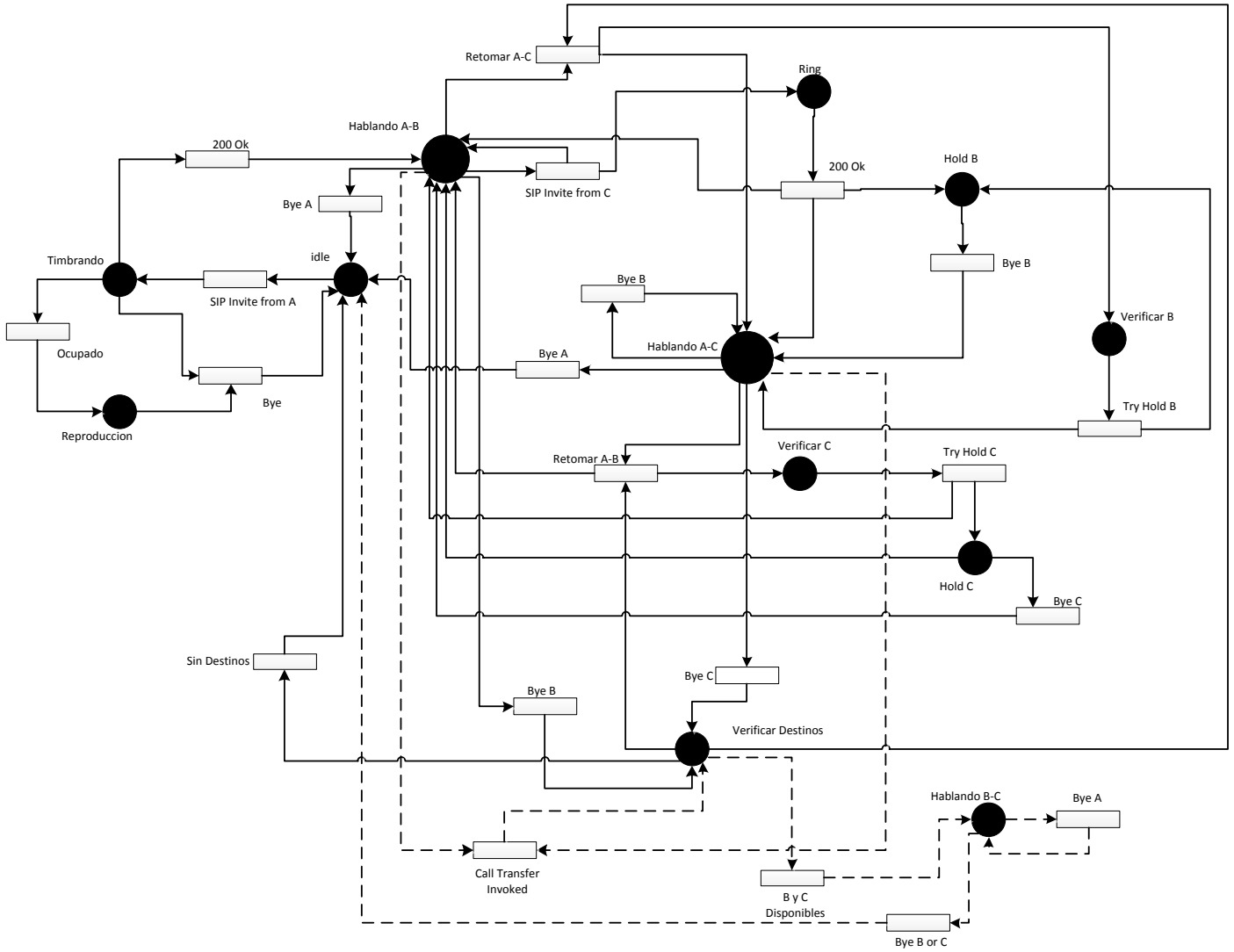


Figura C-9: Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) representado mediante redes de Petri

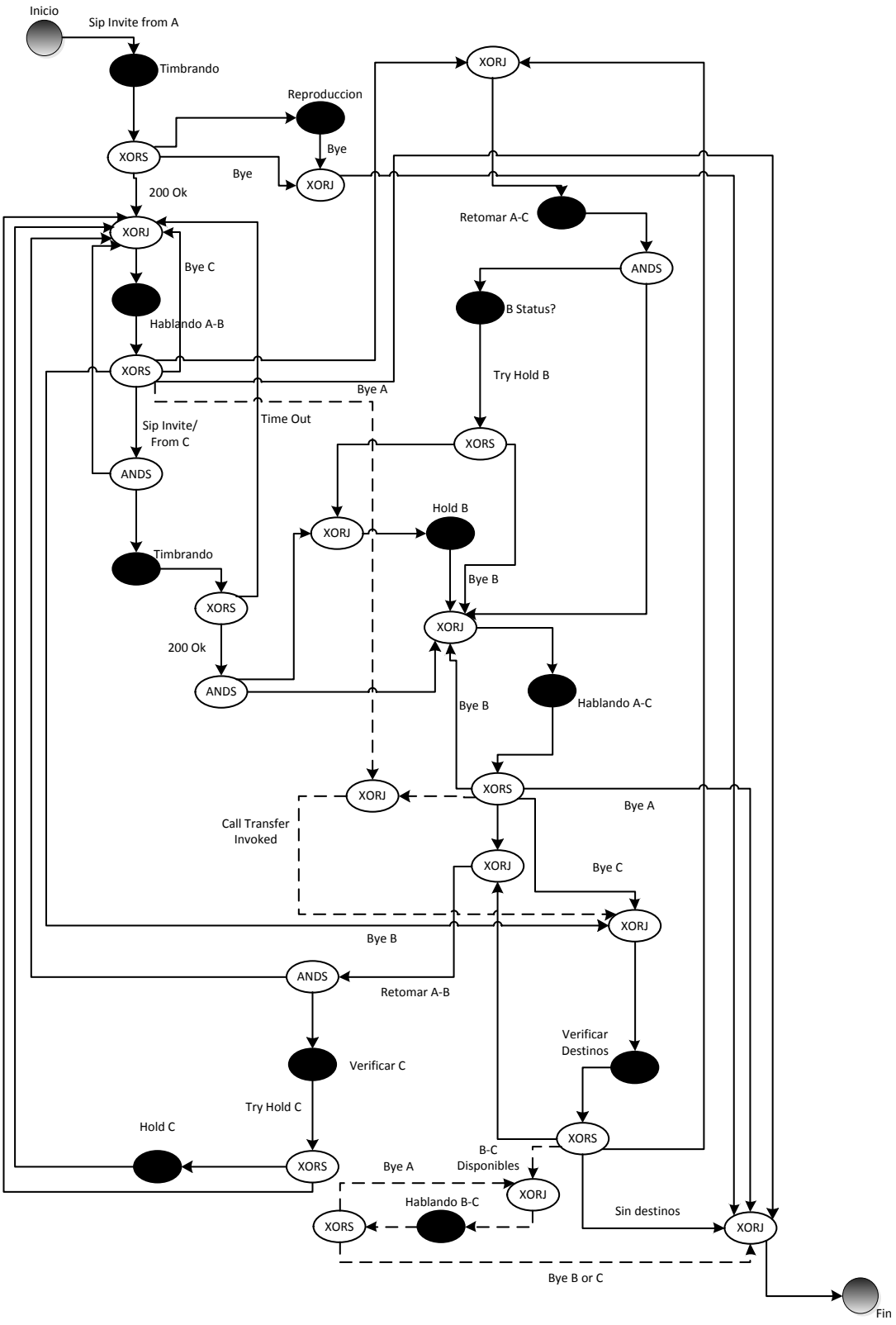


Figura C-10: Servicio de Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) representado mediante Grafos

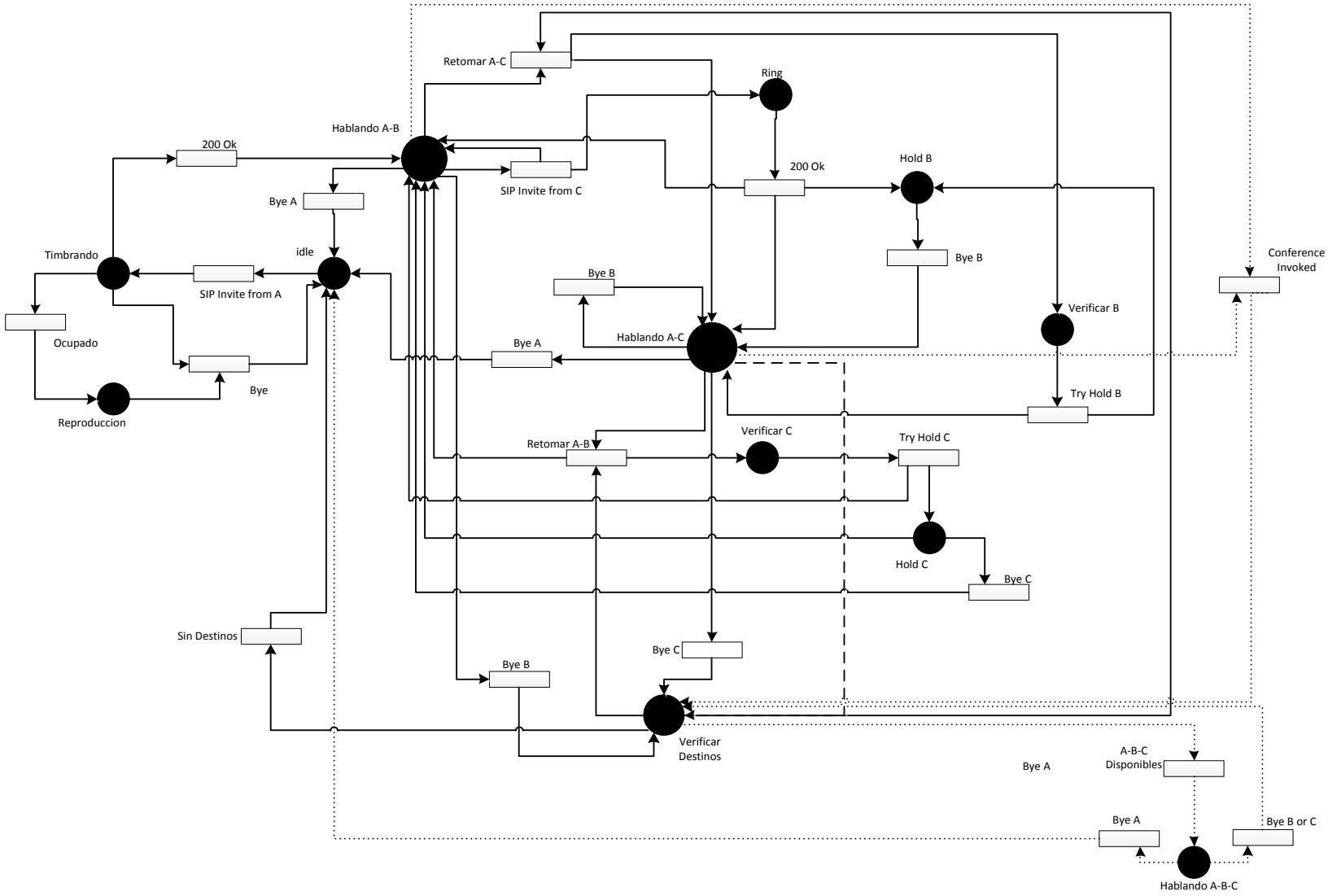


Figura C-11 Llamada + Llamada en Espera (Call Hold) + Conferencia, representado mediante redes de Petri

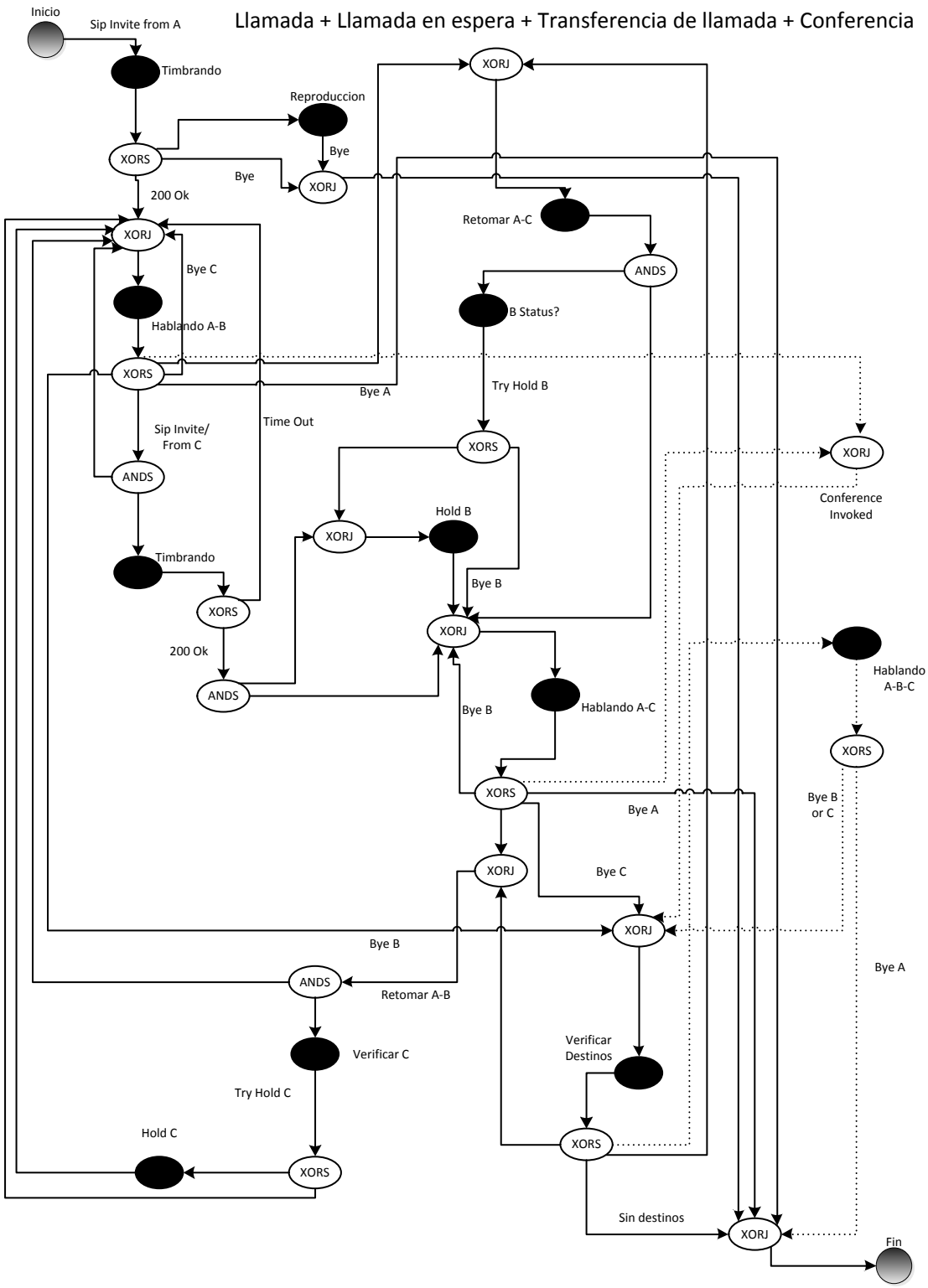


Figura C-12: Servicio de Llamada + Llamada en Espera (Call Hold) + Conferencia representado mediante Grafos

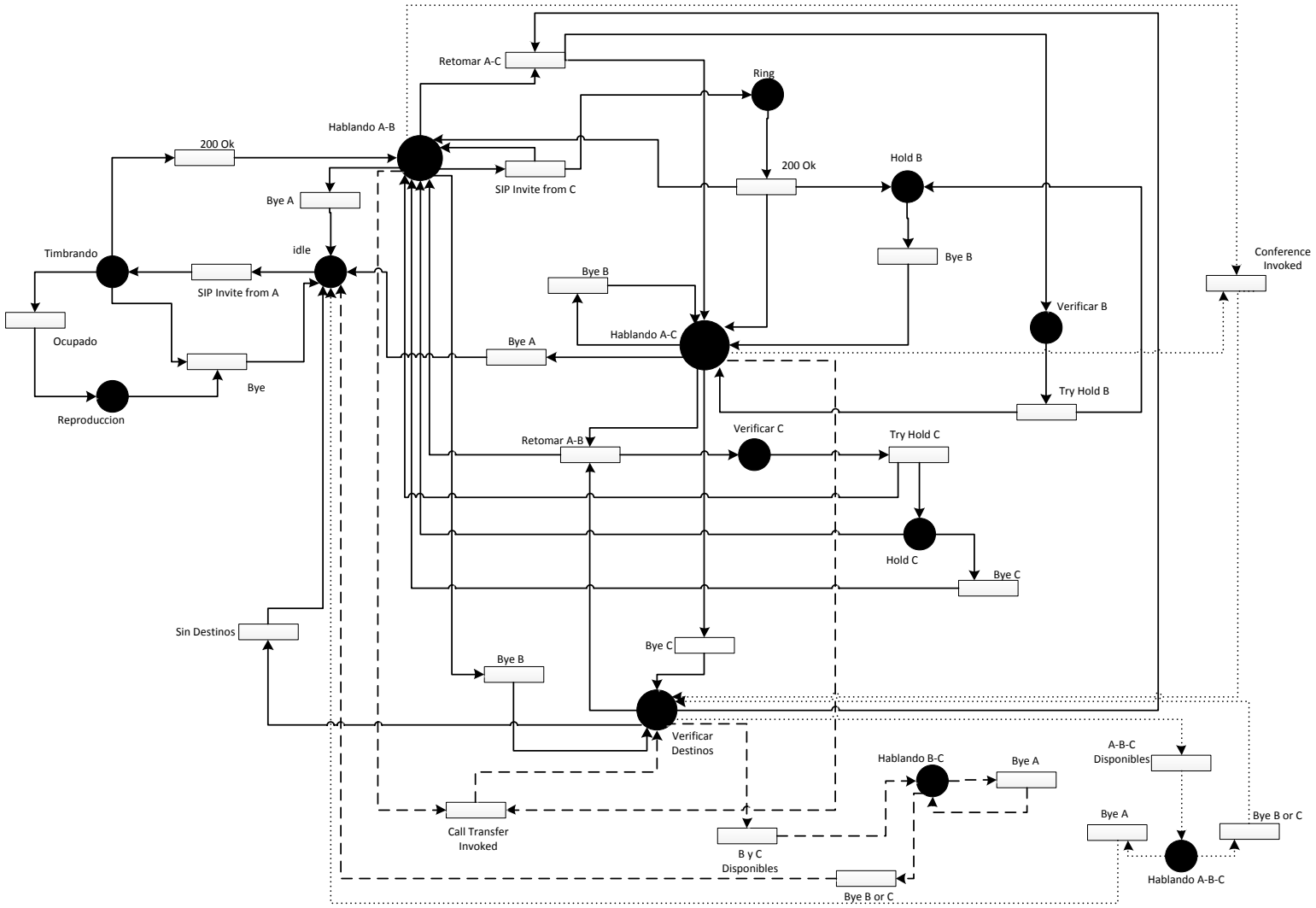


Figura C-13: Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) + conferencia, representado mediante redes de petri

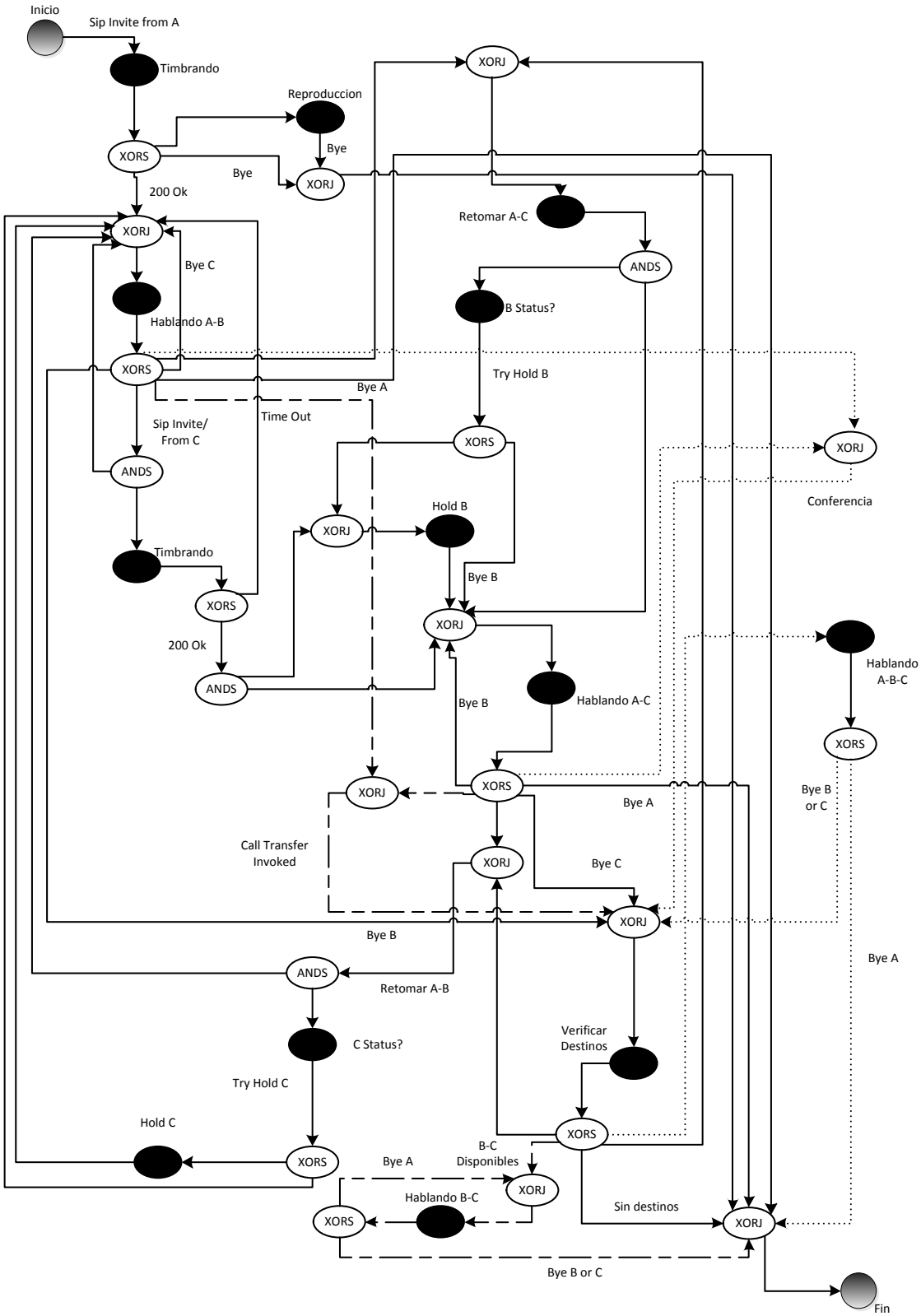


Figura C-14: Llamada + Llamada en Espera (Call Hold) + Transferencia de llamada (Call Transfer) + conferencia representado mediante Grafos

C.1.1.3.7 Llamada + SMS en caso de No respuesta

Este servicio reúne las capacidades de los servicios Telco de llamada y mensajería. Éste comienza cuando un Usuario A, desea llamar a B. En el caso de que B se encuentre ocupado, no disponible o no responda la llamada, el servicio envía un SMS a B, en el cual se le informa que A desea comunicarse con este. La Figura C-15, representa el servicio en redes de Petri y grafos.

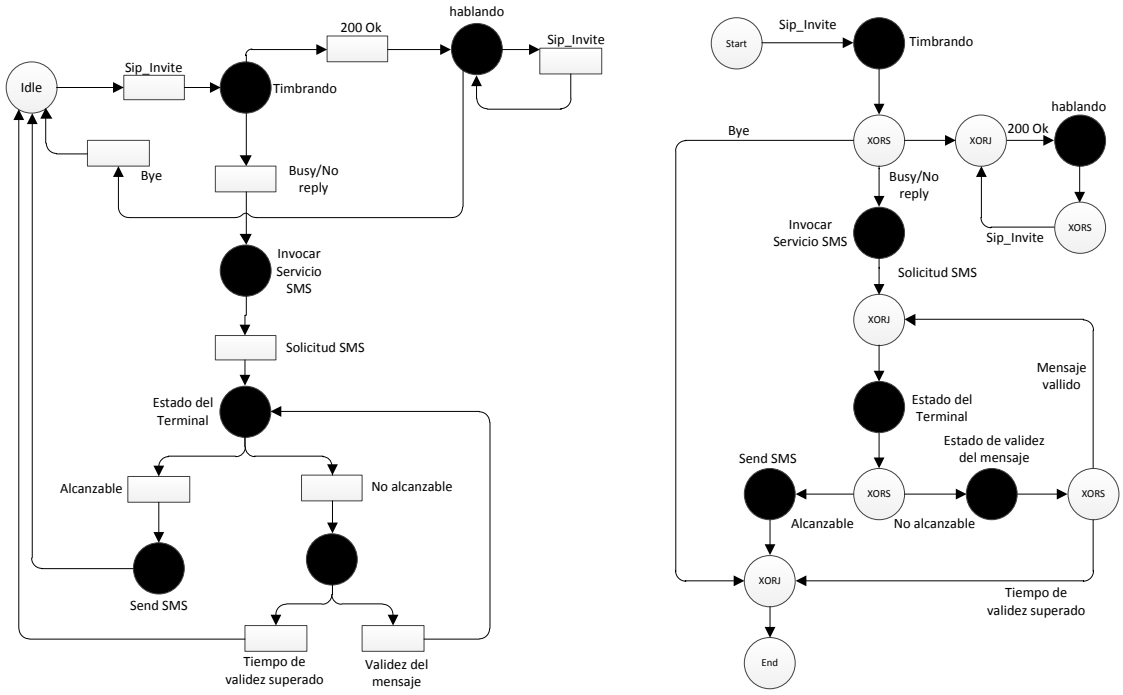


Figura C-15: Servicio de llamada + SMS representado en redes de Petri a) y Grafos b)

C.1.1.3.8 Call + Presence Update

Está compuesto por dos servicios; el servicio de llamada, y el servicio de presencia.

El servicio comienza cuando el usuario recibe una invitación (Sip-Invite), en ese momento se activa simultáneamente el servicio de llamada y de presencia; si el usuario contesta, el servicio de llamada le transmite información al servicio de presencia sobre el estado de la llamada. En el caso de que el usuario conteste, el servicio de presencia se actualizara con el estado “ocupado” o “al telefono”; cuando el usuario haya acabado de hablar, el servicio de presencia se actualizara con el estado “disponible”. La Figura C-16 muestra la representación del servicio *Call + Presence Update.*

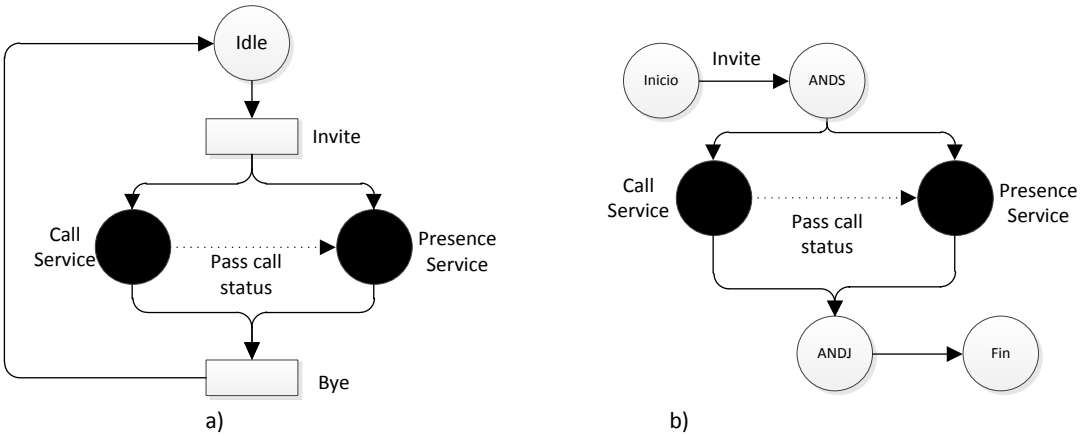


Figura C-16: Representacion del servicio Call + Presence Update representado mediante redes de Petri a) y Grafos b)

C.1.2 Servicios Web

A continuación se presentan un conjunto de servicios Web discriminados en: tradicionales debido a que solo permiten la interacción: usuario final – servicio Web; servicios Web 2.0, los cuales permiten la interacción entre el usuario final – servicio Web y usuario final – usuario final, a través del servicio Web 2.0; y servicios Web compuestos, los cuales incluyen servicios Web tradicionales y servicios Web 2.0.

C.1.2.1 Servicios Web Tradicionales

Los servicios web tradicionales pueden ser modelados genéricamente como se muestra en la Figura C-17, ya que la diferencia radica en el tipo de datos de entrada y de salida.

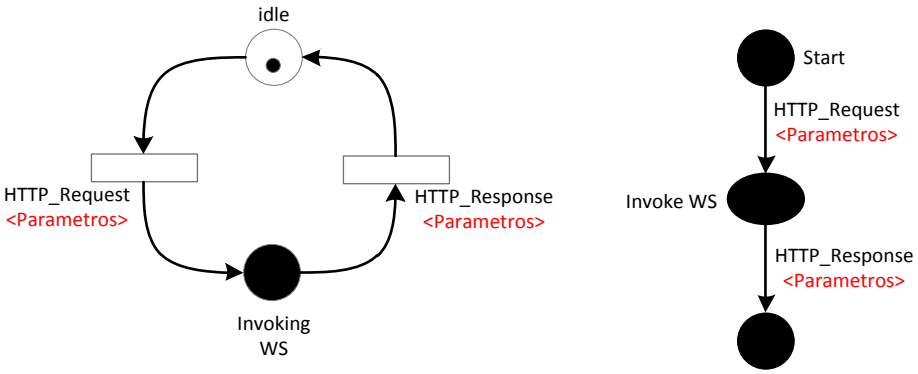


Figura C-17: Representación formal de Servicio web genérico

Si se tienen en cuenta las posibles respuestas del servicio como errores (4XX, 5XX) o respuesta satisfactoria (200 OK), la representación formal se observa en la Figura C-18

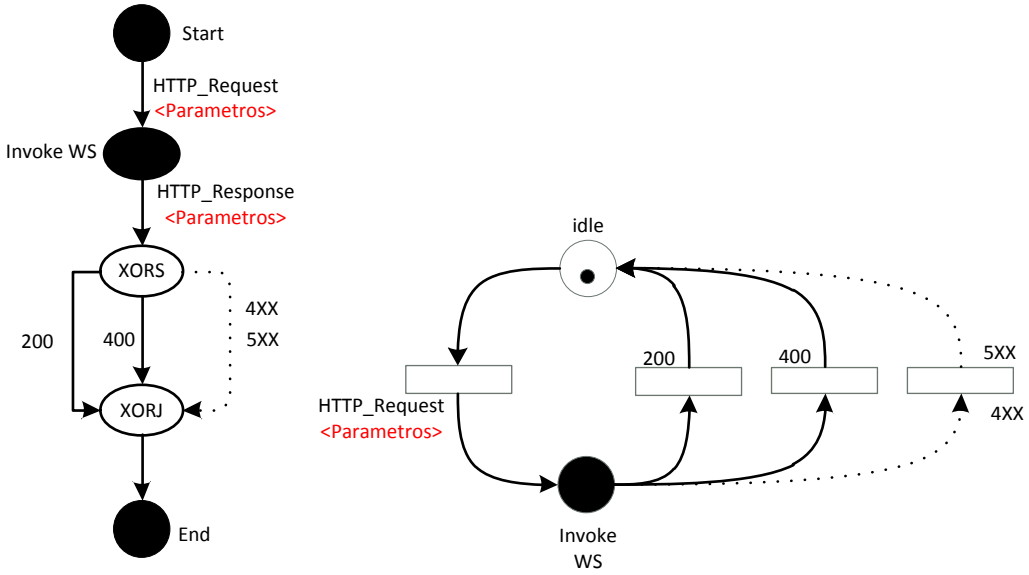


Figura C-18: Representación formal de Servicio web genérico con posibles respuestas

Algunos servicios web como los ofrecidos por One API [4] , la cual es una empresa dedicada al desarrollo de tecnologías móviles y web, ofrece a operadores de telecomunicaciones, facilidades para la integración entre sus plataformas y servicios web a través de Apis que siguen las especificaciones de Parlay X.
Los servicios deben ser accedidos mediante APIS RESTfull las cuales utilizan comandos http: POST, GET, PUT, y DELETE. Estos servicios siguen el modelo de la Figura C-18, y algunos de estos se nombran en la Tabla C-1.
Finalmente, el tipo de respuesta se da en formato JSON para cada uno de los servicios de la Tabla C-1; por ejemplo, para el servicio de pagos, la respuesta se muestra en la Figura C-19.

Tabla C-1: Servicios Web, One Api

Servicio	Método de Invocación	Operación	Parámetros Obligatorios de petición
Sms	Post	Enviar Sms a uno o más terminales	• address: número MSISDN (ej +057305123576) • senderadress (string): dirección de la fuente que envía el mensaje • message (string): cuerpo del mensaje, máximo 160 caracteres
MMS	Post	Enviar MMS a uno o más terminales	• address: número MSISDN senderadress (string): direccion de la fuente que envía el mensaje • message (string): mensaje tipo MIME
Ubicación	GET	Obtiene la ubicación de un dispositivo	• address: número MSISDN • requestedaccuracy: presicion en metros del resultado
Pagos	POST	Hacer el cargo de un monto de dinero	• amount: cantidad que va a ser cargada al usuario

		a un usuario	• currency: moneda en la que se hace el cargo (ej usd, colp) • enduserid: número MSISDN • referencecode: codigo de la transacción. es utilizado en caso de reclamos por el valor cargado • transactionoperationstatus: “chargered”
Capacidades del dispositivo	GET	Obtener las capacidades del dispositivo	address: número MSISDN
Perfil de conexión de datos	GET	Obtiene el tipo de conexión que emplea el dispositivo para conectarse a la red movil	address: número MSISDN

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 12345
Date: Thu, 21 May 2012 02:51:59 GMT
Location: http://example.com/payment/v1/
tel%3A%2B16309700001/transactions/amount/abc123

{"amountTransaction": {
   "clientCorrelator": "54321",
   "endUserId": "tel:+16309700001",
   "paymentAmount": {
      "chargingInformation": {
         "amount": "10",
         "code" : "xyz",
         "currency": "USD",
         "description": "Alien Invaders Game"
      },
      "totalAmountCharged": "12.99",
      "chargingMetaData" : {
         "onBehalfOf" : "Example Games Inc",
         "purchaseCategoryCode" : "Game",
         "channel" : "WAP",
         "taxAmount" : "0"
      }
   },
   "referenceCode": "REF-12345",
   "serverReferenceCode": "ABC-123",
   "resourceURL": " http://example.com/payment/v1/
tel%3A%2B16309700001/transactions/amount/abc123
   "transactionOperationStatus": "Charged"
}}
```

Figura C-19: Respuesta del servicio de pagos en formato JSON

C.1.2.2 Servicios Web 2.0

Gracias a los nuevos servicios que están disponibles en la internet, ha surgido el concepto no estandarizado de Web 2.0, el cual busca diferenciar los servicios tradiciones donde el usuario se limita a hacer una consulta-peticion, de los nuevos servicios que sirven como canal entre usuarios para que mantengan contacto, compartan y modifiquen información que se encuentra disponible en internet, etc.

Sin embargo, dada la gran cantidad de nuevos servicios disponibles, la tarea de clasificar estos servicios ya sea por su tecnología, o por la funcionalidad que cumplen para el usuario, se convierte en un proceso algo tedioso; por lo anterior se han realizados estudios como el de Shang *et al* [5]*,* el cual busca clasificar servicios Web 2.0 no por su tecnología, ni por su funcionalidad, sino por algo que denominan "el grado de creación de nuevo conocimiento" la cual se basa en un ciclo, que permite la divulgación de información para que los usuarios se apropien de esta, permitiendo así, la creación de nuevo conocimiento.

La primera fase de este ciclo es llamada: socialización, donde los usuarios: comparten información en la web 2.0, participan en comunidades, se comunican mediante Voip, etc.; la segunda fase es llamada: externalización, donde los usuarios escriben mails, comparten información mediante chat, etc.; la tercera fase es llamada: combinación, donde se emplean tecnologías de la web 2.0 para compartir la información, como: RSS (Really Simple Syndication), Etiquetado (Tagging), Folksonomias, etc; Por último, la internalización es un proceso sistemático que refleja el aprendizaje colectivo a través de acciones y prácticas. Éste ciclo se muestra en la Figura C-20.

Figura C-20: Ciclo del modelo de creación de conocimiento en sitios Web 2.0 (Shang *et al.* 2011 – Elseiver)

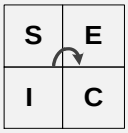
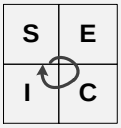
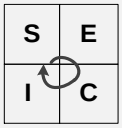
Las 4 fases en el modelo de creación de conocimiento SECI (Socialización S, Externalización E, Combinación C, Internalización I) son tenidas en cuenta para la clasificación de los servicios y adicionalmente los siguientes aspectos:

- El grado de control que tiene el servicio sobre la información que se comparte sobre este, el cual es definido como bajo y alto; bajo: cuando existe un registro de los usuarios, la información compartida es estandarizada; y alto cuando la información compartida tiene que ser revisada y autorizada para que sea compartida con otros usuarios.
- El servicio al cliente: el proceso de aprendizaje del usuario se puede beneficiar a través de las características de la WEB 2.0, como: la asimilación, regeneración, reinterpretación y la retención del nuevo conocimiento

Teniendo en cuenta las fases del modelo (SECI), el grado de control que tiene el servicio sobre la información y el valor que le da el usuario al servicio, se completa el modelo para la clasificación de los servicios, como se muestra en la, definiendo así cuatro roles principalmente:

- Intercambiador: Comprende S – E, con mecanismo de control bajo (Tipo I)
- Agregador: Comprende S – E – C, con mecanismo de control bajo (Tipo II)
- Colaborador: Comprende S – E – C – I, con mecanismo de control alto (Tipo III)
- Liberador: Comprende S – E – C – I con mecanismo de control Bajo (Tipo IV)

Tabla C-2: Clasificación de servicios de la WEB 2.0, siguiendo el modelo de creación de conocimiento

	Intercambiador	**Agregador**	**Colaborador**	**Liberador**
Aplicaciones Web 2.0	Llamadas y VoiP, Chat, Email	Blogs, Música, foto, video RSS	Wikis, oficina, negocios, programación, juegos	Negocios, oficina, programación
Tipo	I	II	III	IV
Circulo de Creación de Conocimiento.				
Mecanismo de Control	Registro	Registro, estandarización	Registro, estandarización, sistematización, autorización, revisión	Acceso, estandarización
Servicio para el cliente	Compartir información instantáneamente	• Compartir y retener información • Información suficiente	• Compartir, retener, asimilar conocimiento bajo estándares específicos • Conocimiento de calidad	• Compartir, retener, asimilar conocimiento bajo estándares libres • Conocimiento de libre acceso
Ejemplos	MSN, Skype	Twitter, Youtube, Facebook	Wikipedia, Answers.com	Linux

Gracias a al trabajo de Shang *et al.* [5] es posible clasificar servicios como tipo I, II, III y IV. Algunos servicios de ejemplo siguiendo esta clasificación se muestran en la Tabla C-3.

Tabla C-3: Servicios Web 2.0 según el ciclo del modelo de creación de conocimiento en sitios Web 2.0

Tipo	Nombre	Descripción
Tipo I	MSN	Abreviación de MicroSoft Network, conjunto de servicios que incluyen mensajería instantánea, correo electrónico, VoIp, etc.
	Skype	Software que permite la comunicación mediante texto, voz y video sobre internet (VoIp), empleando el modelo punto a punto (P2P).
	Google Talk	Es un cliente de mensajería instantánea y VoIP que hace uso del protocolo extensible de mensajería y comunicación de presencia (XMPP).
Tipo II	MySpace	Sitio web donde los usuarios comparten sus pensamientos, fotos, actividades, entre otros. Cuenta con servicios de noticias, clasificados, videos, aplicaciones para móviles, etc, además de apis para desarrolladores.
	Youtube	Es un sitio web donde los usuarios pueden subir y compartir videos. Utiliza la tecnología Flash para reproducir los videos en línea.
	Facebook	Sitio web de red social, inicialmente creado para que estudiantes de la universidad de Hardvard se mantuvieran conectados, pero en la actualidad cualquier usuario con una cuenta de correo electrónico puede acceder y compartir sus pensamientos, fotos, videos y desde el año 2011 la posibilidad de realizar videollamadas.
	Twitter	Es una red social que permite compartir a sus usuarios información mediante: texto de máximo 140 caracteres y fotos; cuenta con una api abierta para desarrolladores.
	43things	Es una red social basada en los principios del etiquetado (tagging). Los usuarios ingresan una serie de gustos y expectativas, y dependiendo de estas, los usuarios son conectados a otros, que compartan características similares.
	Linkedin	Es una red social orientada a negocios y a profesionales. Este sitio permite a los usuarios, encontrar ofertas laborales, realizar negocios, conocer gente que comparte sus mismos gustos profesionales, entre otros.
Tipo III	Answers.com	Es un sitio basado en el intercambio de conocimiento sobre internet. Permite a los usuarios escribir preguntas o consultar preguntas previamente respondidas sobre diferentes temas.
	Saleforces.com	Es un sitio enfocado a desarrolladores donde estos pueden desarrollar sus aplicaciones, programas, etc. en línea, gracias a una plataforma que está en un servidor.
	Yahoo Widget	Es un sitio web que permite crear widgets en línea basados en la tecnología de JavaScript, AdobeFlash entre otros.
	Wikipedia	Es un proyecto para escribir comunitariamente una enciclopedia libre. Los usuarios pueden editar el contenido de los artículos de manera muy simple.
Tipo IV	Openoffice	Es una plataforma ofimática[2] de libre distribución que incluye: Procesador de textos, hojas de cálculo, presentaciones, etc. Ésta disponible en varias plataformas como: Microsoft Windows, GNU/Linux, BSD, Solaris y Mac OS.
	Linux	Linux es el Núcleo del sistema operativo GNU/Linux. Es el ejemplo más representativo de software libre; todo el código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General).

Algunos de los servicios descritos anteriormente son modelados utilizando las representaciones formales de redes de Petri y Grafos, lo cual se muestra a continuación.

[2] Ofimática: es una recopilación de aplicaciones, las cuales son utilizados en oficinas y sirve para diferentes funciones como: crear, modificar, organizar, escanear, imprimir.

C.1.2.2.1 Servicio de Skype

La Figura C-21 y Figura C-22 ilustran la representación del servicio Sype mediante redes de Petri y Grafos. Éste servicio ofrece una gran cantidad de funcionalidades como: Chat, llamada, video conferencia, pagos entre muchos otros.

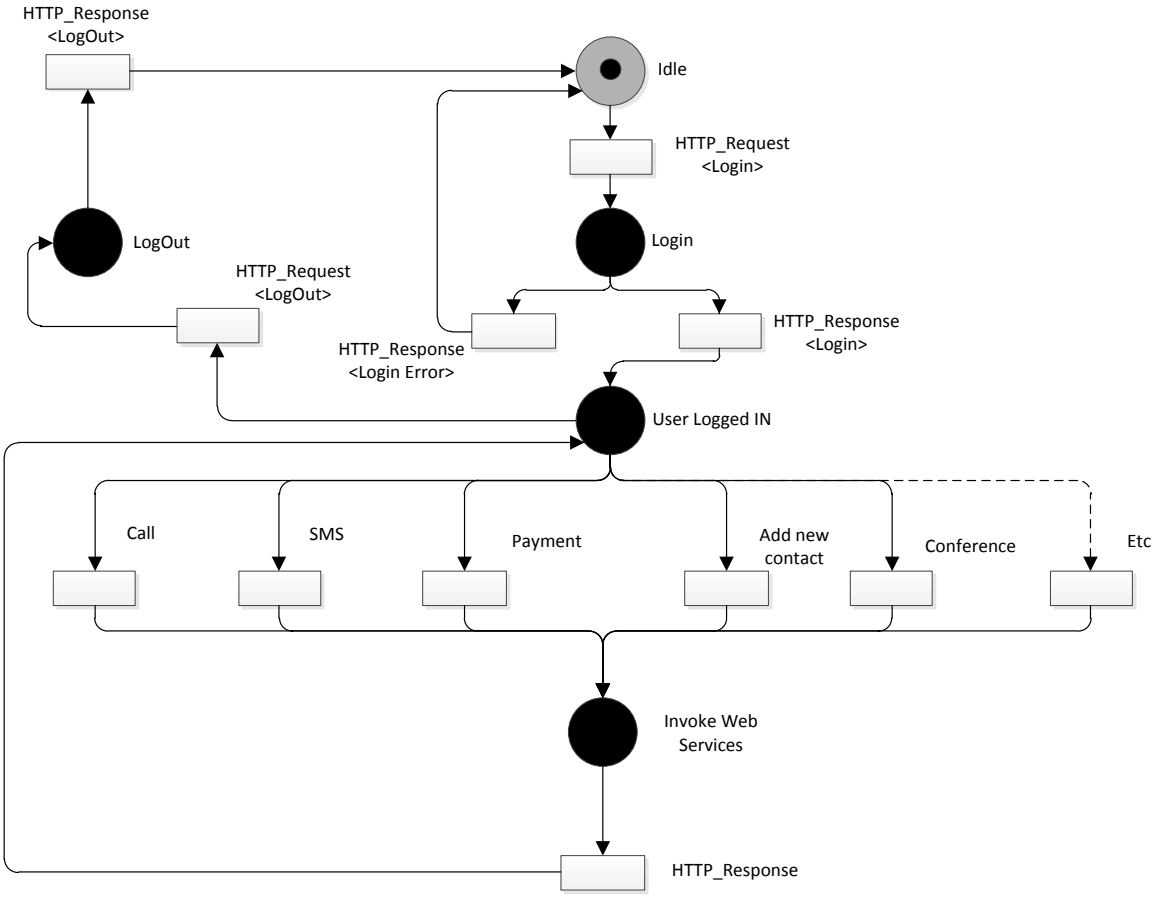


Figura C-21: Servicio Skype representado mediante redes de Petri

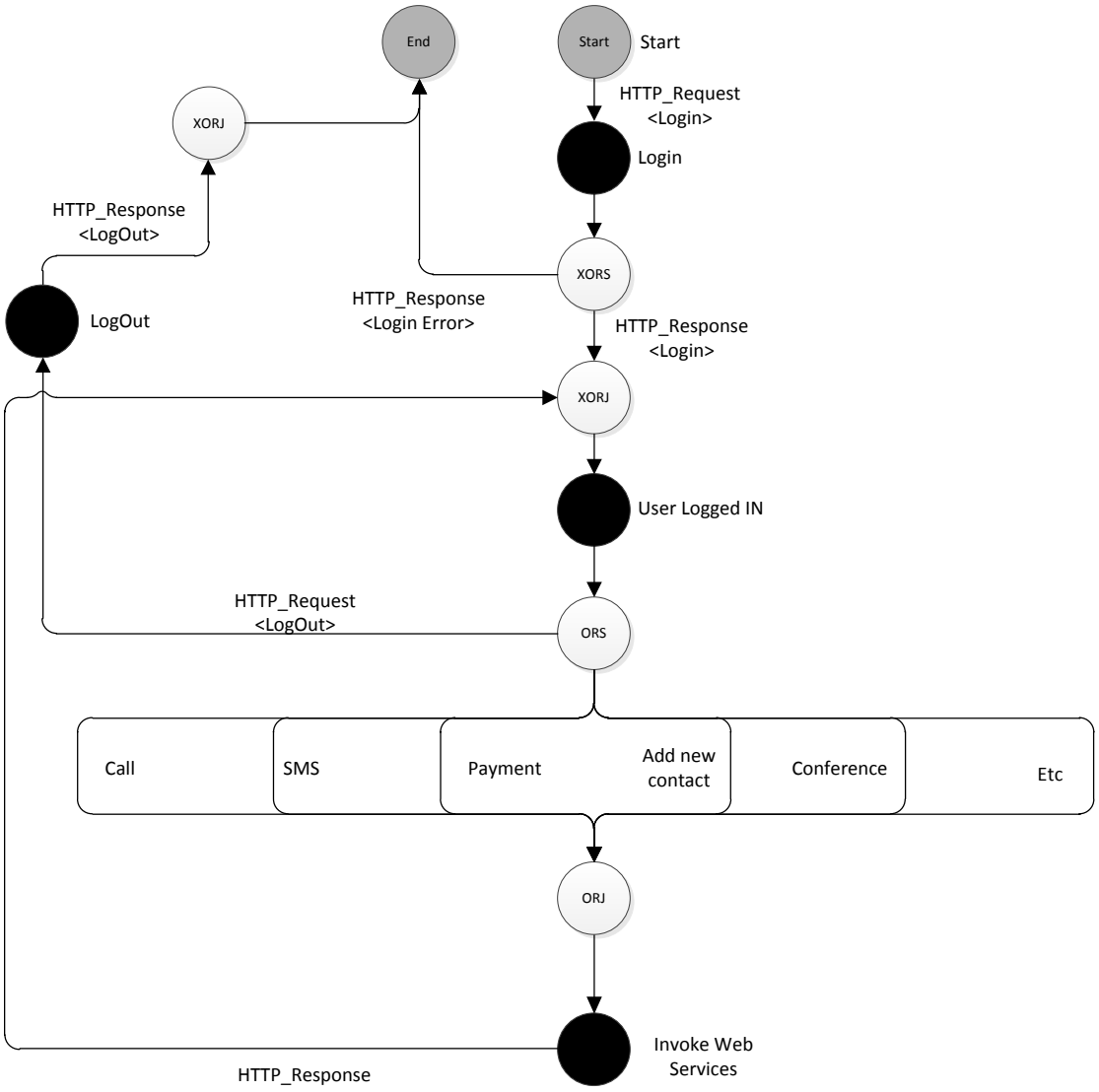


Figura C-22: Servicio Skype representado mediante Grafos

C.1.2.2.2 Servicio de Twitter

Éste servicio fue originalmente concevido siguiendo el modelo de los mensajes de texto SMS, con la variación de que estos son enviados en forma masiva, adicionalmente, los usuarios pueden elegir las personas que envian dichos mensajes y elegior quien lee los que ellos publican. Twitter en los últimos años ha crecido en términos de funcionalidades, algunas de ellas se ilustran en la Figura C-23 y Figura C-24.

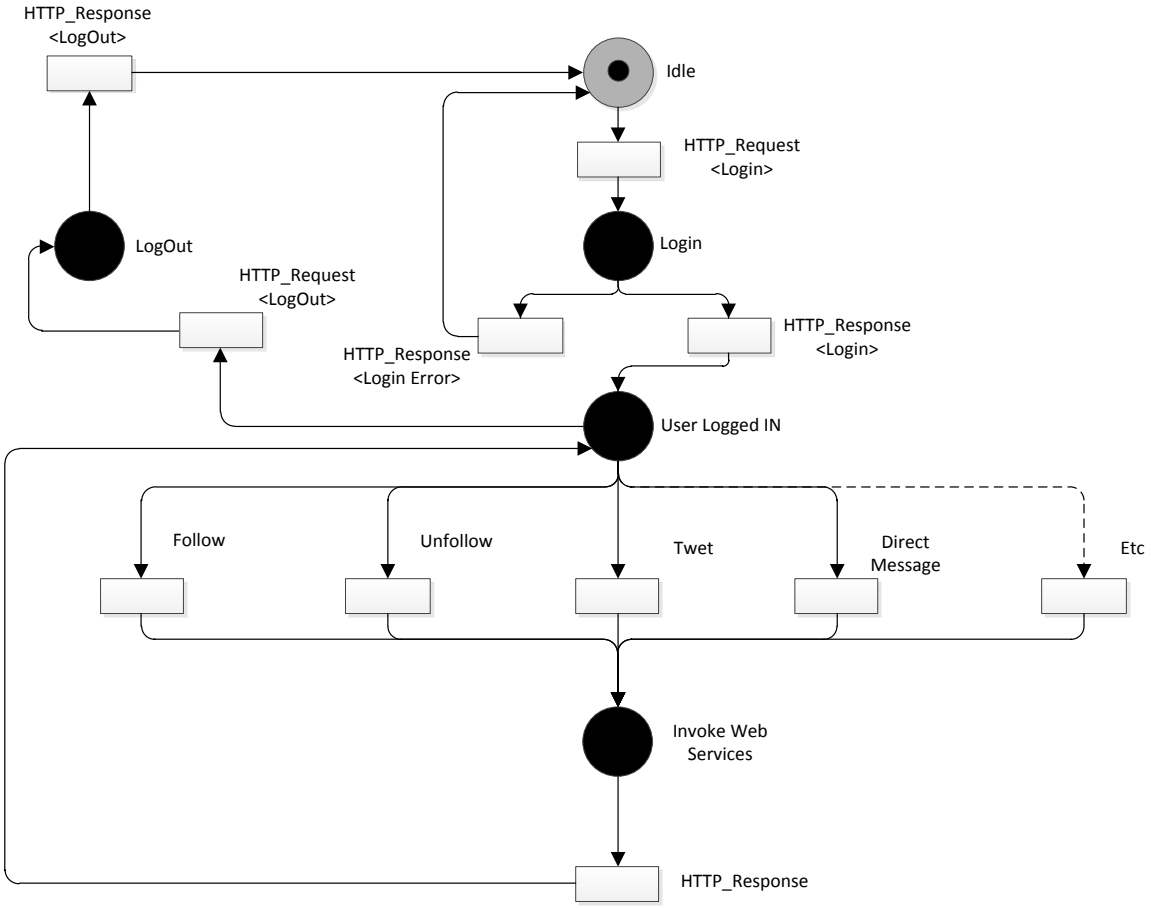


Figura C-23: Servicio de Twitter representado mediante redes de Petri

C.1.2.2.3 Servicio Facebook

Esta red social integra funcionalidades para mantener comunicados a los usuarios que hacen parte de ella. Algunas de estas son: publicación del "estado", etiquetado de fotos, chat y hace algún tiempo, video conferencia con la integración del servicio Web Skype. Algunas de estas funcionalidades se ilustran en la representación de redes de Petri y grafos (Figura C-25 y Figura C-26).

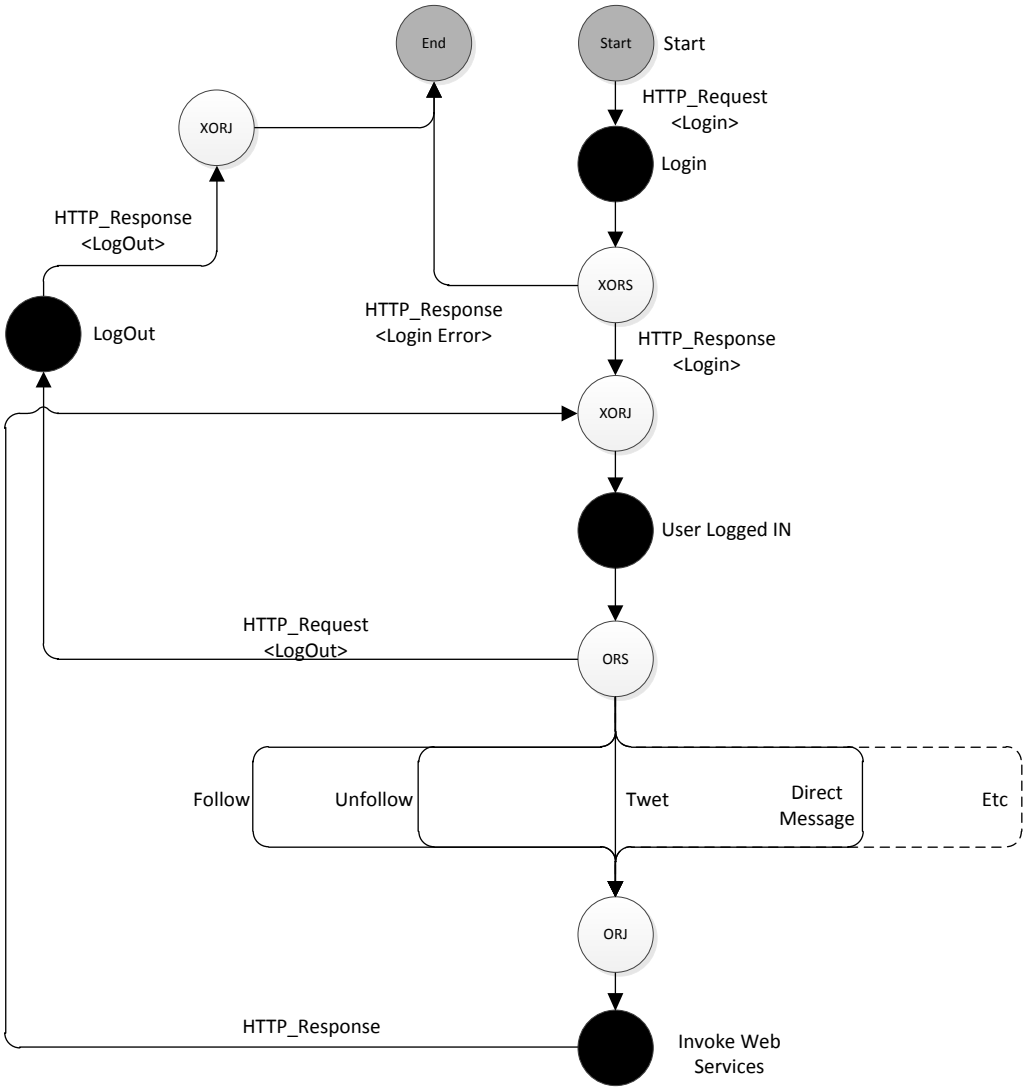


Figura C-24: Servicio de Twitter representado mediante grafos

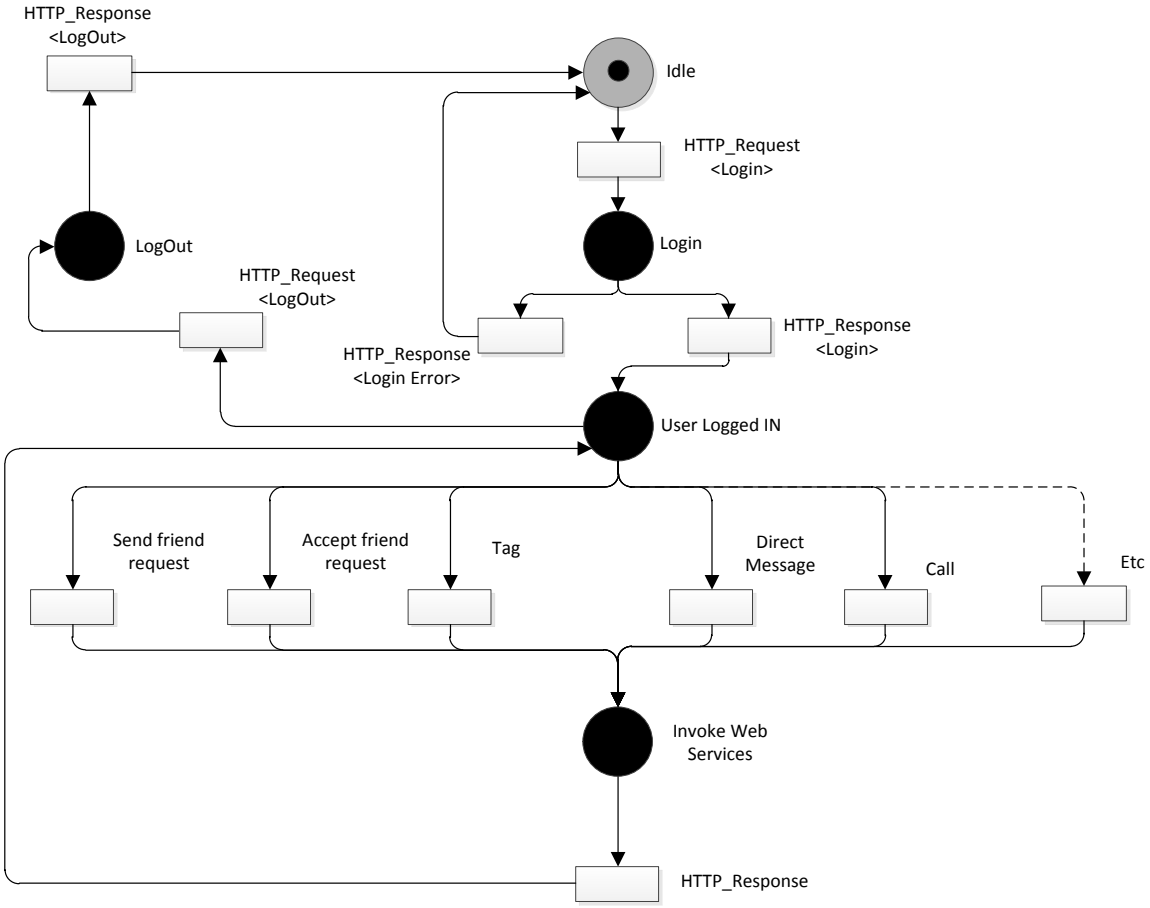


Figura C-25: Serrvicio de Facebook Representado mediante redes de Petri

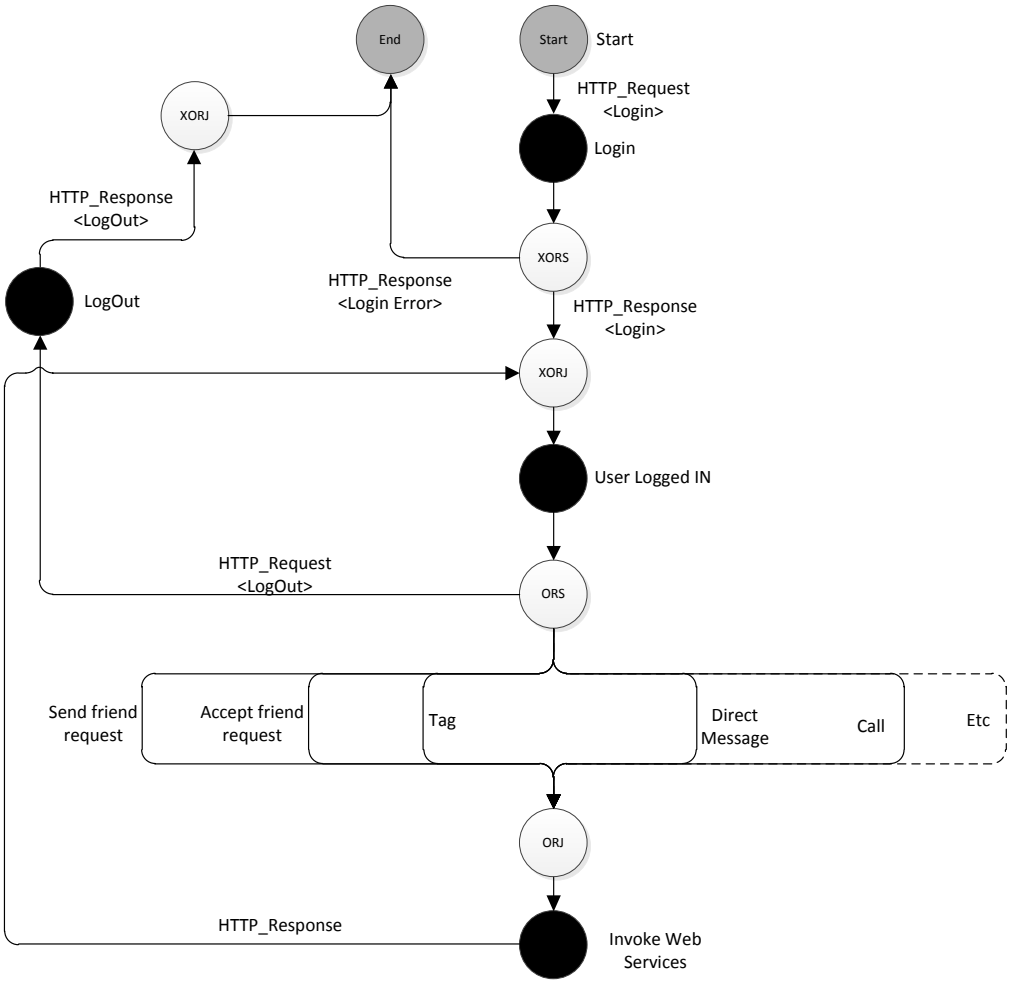


Figura C-26: Servicio de Facebook Representado mediante Grafos

C.1.2.3 Servicios Compuestos

C.1.2.3.1 Amazon Seller Control

Este servicio representado mediante redes de Petri y Grafos en la Figura C-28 y Figura C-29 respectivamente, permite a los vendedores mayoritarios, tener control de sus ventas y consultar si sus precios son competitivos respecto a otros usuarios que venden productos idénticos.
El servicio comienza con una petición de login para ingresar al servicio, una vez el usuario se ha registrado satisfactoriamente, éste puede buscar los productos que ofrece mediante palabras clave, o con el nombre completo del producto del cual desea obtener información. Una vez el usuario ha elegido un producto de su stock, el servicio le muestra información de: los precios competitivos de otros usuarios, el precio más económico de venta, el precio al cual el vendedor está ofreciendo el producto, y la información de perfil de los vendedores que son competencia.
Esta información es importante para el vendedor, porque le permite conocer si tiene que disminuir el precio de sus productos, para ser competitivo en el mercado.

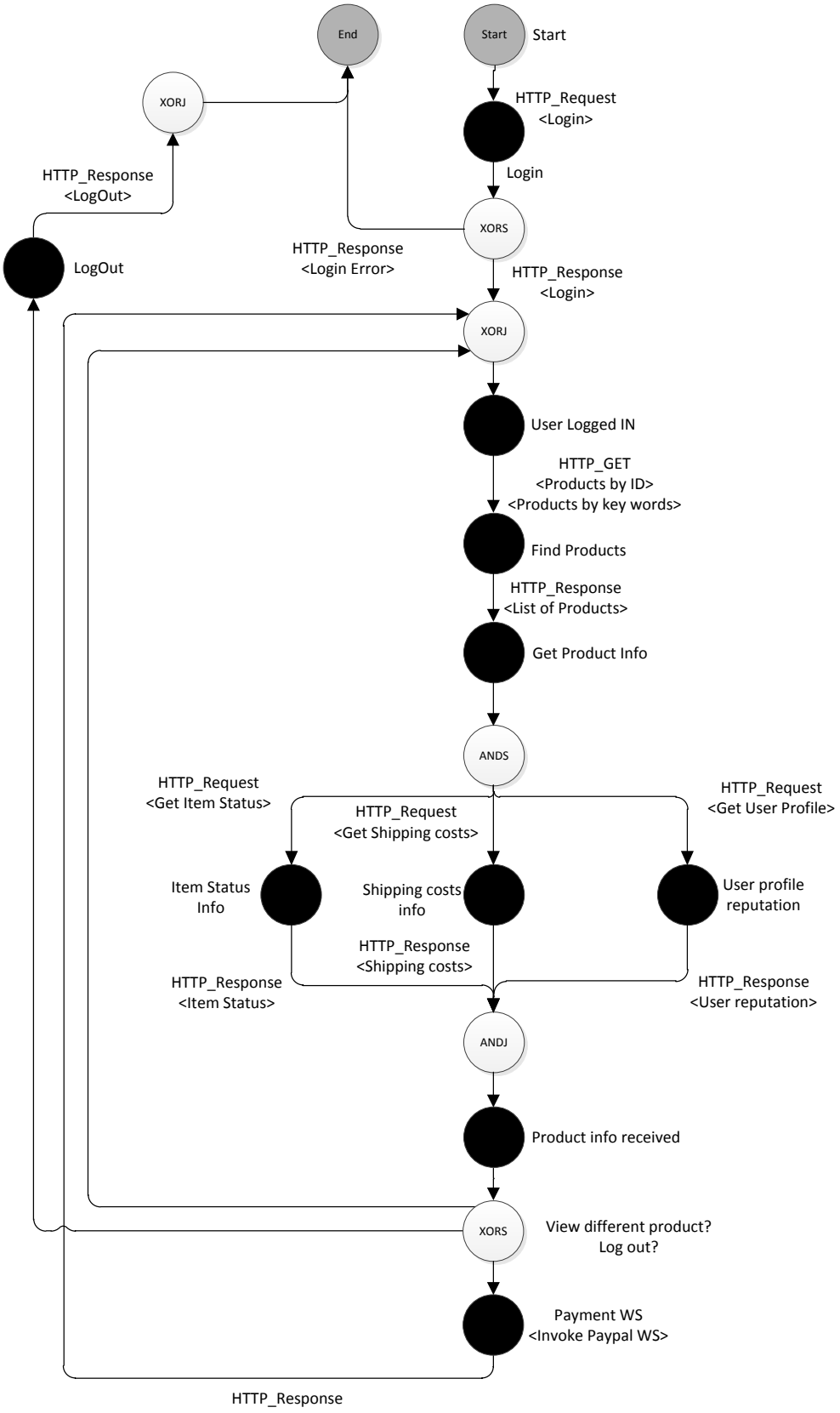


Figura C-27: Servicio Ebay product finder representado mediante redes de Grafos

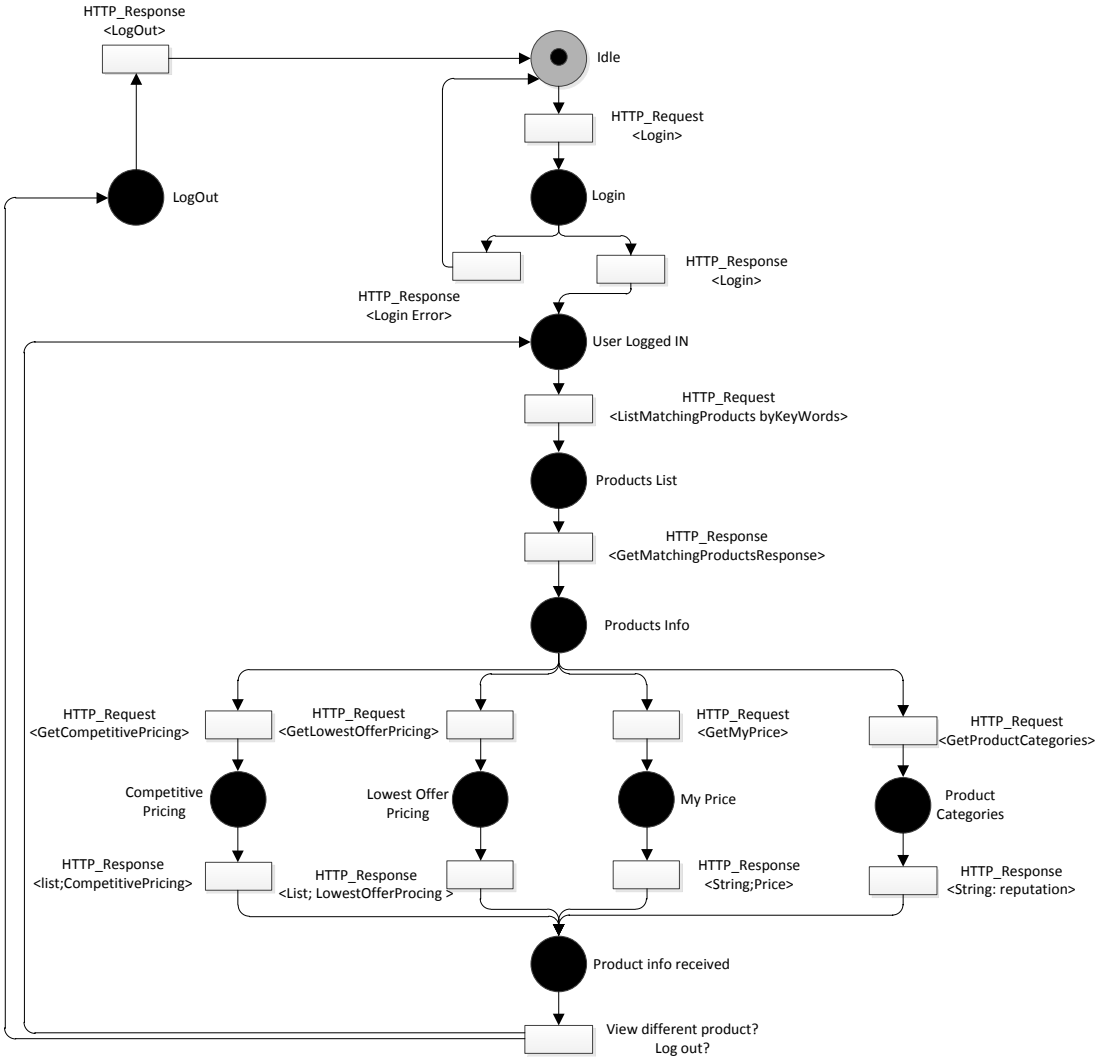


Figura C-28: Servicio amazon seller control representado mediante redes de Petri

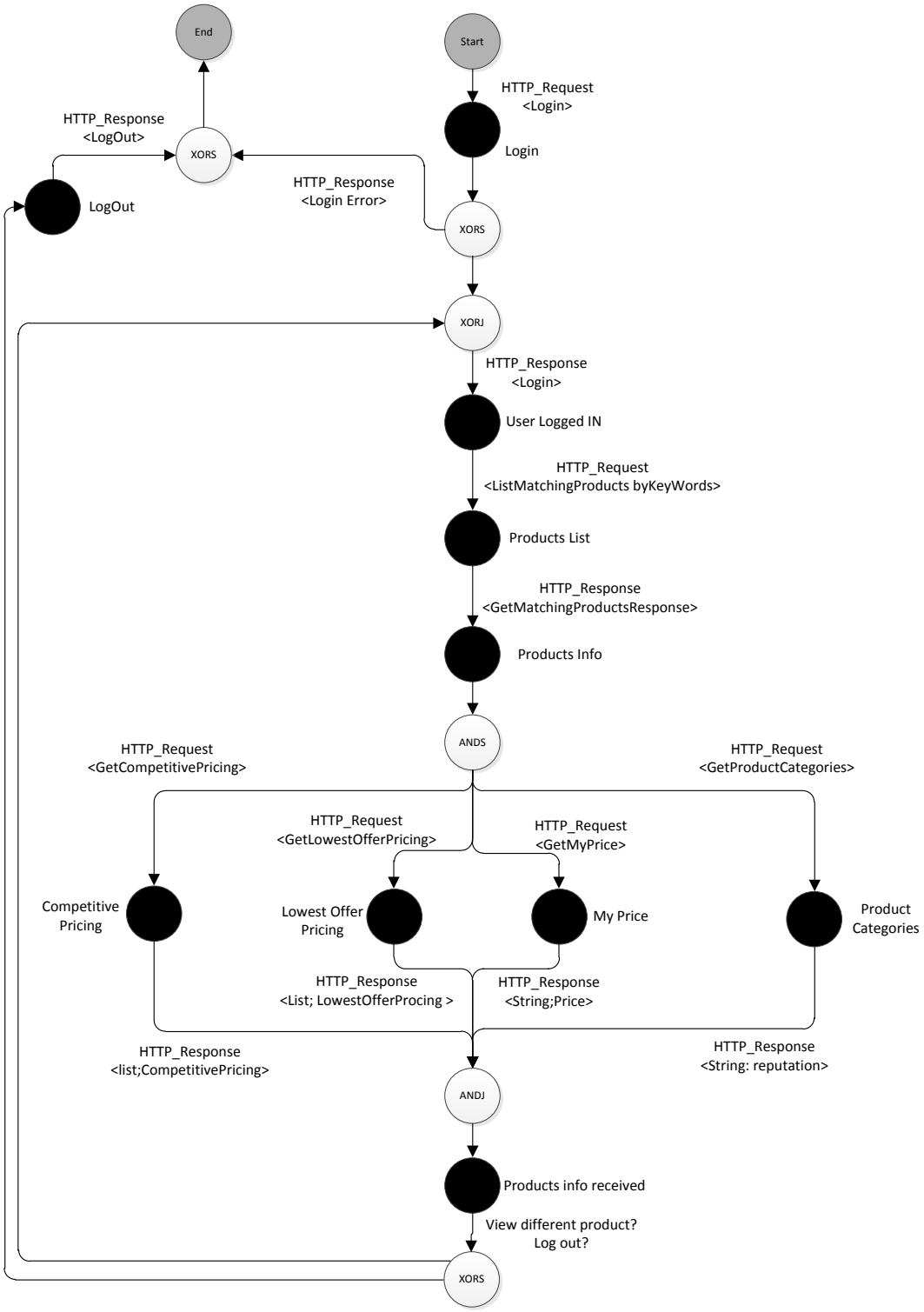


Figura C-29: Servicio amazon seller control representado mediante de Grafos

C.1.3 Servicios Convergentes

Los servicios convergentes que se muestran a continuación son aquellos que están compuestos por servicios web como por servicios de telecomunicaciones.
A continuación se presenta una lista de servicios convergentes.

C.1.3.1 Shopping

En el servicio de compras [6] de Mobicients el usuario tiene la posibilidad comprar diferentes productos en un portal Web; una vez hecha la compra, este recibe una llamada para confirmar la compra del producto y la fecha estimada de envío.
La figura Figura C-30 y Figura C-31 muestra la representación del servicio en redes de Petri y Grafos respectivamente. El servicio comienza con la invocación del servicio Web que permite escoger los productos para la compra. Una vez el usuario realiza el pago, el servicio realiza una llamada automáticamente al usuario para confirmar su compra y mediante un lector de pulsos de teléfono (DTMF Dual Tone Multi-Frecuency) puede aprobar o rechazar el pedido. Además, si monto es superior a 100USD se hace una llamada a un administrador, el cual puede rechazar o aprobar la orden, y si es menor a ese monto, la orden se aprueba y se hace una llamada nuevamente al usuario para que establezca la hora y fecha de envió del producto.

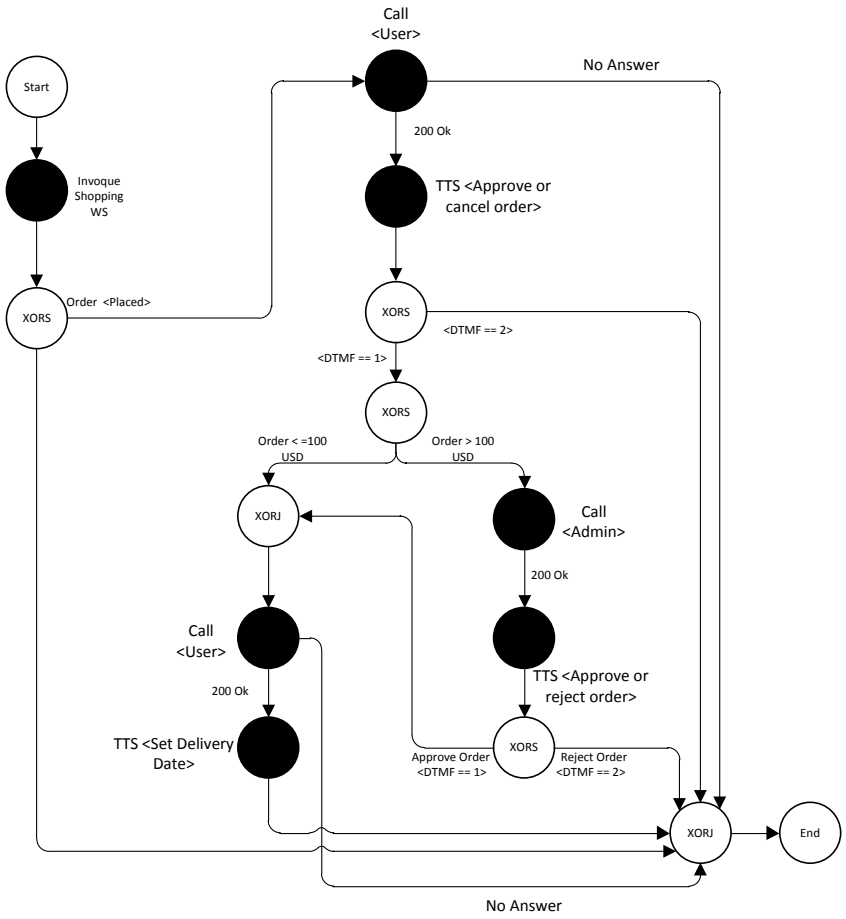


Figura C-30: Servicio Shopping de Mobicents representado mediante redes de Petri

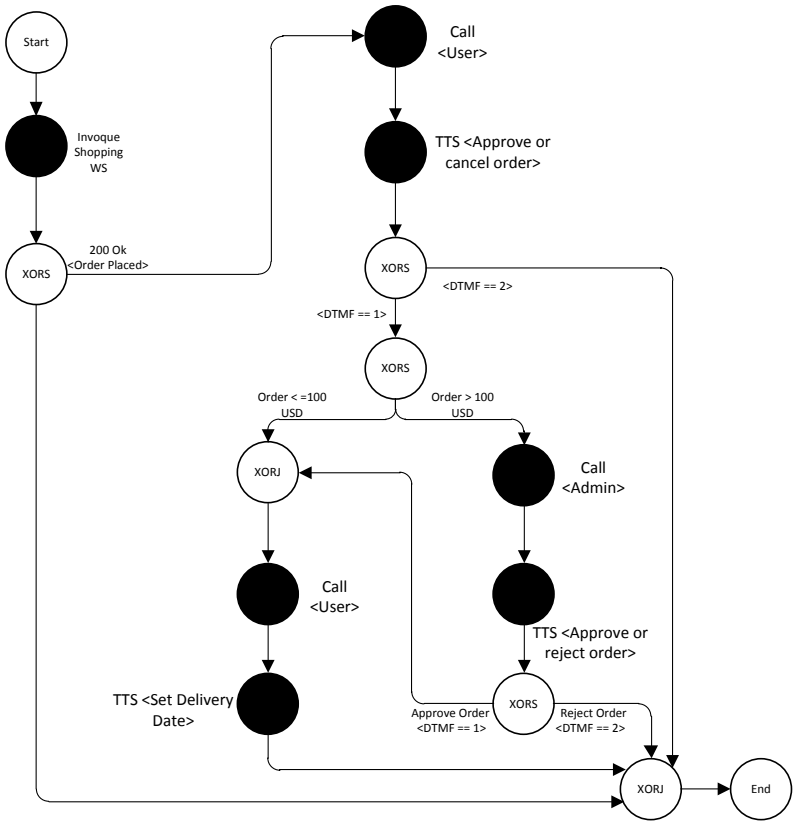


Figura C-31: Servicio Shopping de mobicents representado mediante Grafos

C.1.3.2 Financial Message

Este servicio permite hacer una consulta mediante un mensaje de texto, sobre el estado de una acción en una bolsa de valores.

El servicio comienza cuando el usuario envía un mensaje de texto con el nombre de la acción que desea consultar, el mensaje de texto es analizado en el nodo o lugar “manejo de mensajes Sip”, donde se extrae el nombre de la acción. Este mismo nodo se encarga de formar la petición HTTP-GET, que será enviada al servicio WEB, este último, retorna el valor de la acción.

Con el valor de la acción ya obtenido, se consulta la información del usuario que hizo la petición. Esta información se pasa de regreso al nodo “manejo de mensajes sip” el cual, forma un mensaje de texto con el valor de la acción al usuario que hizo la consulta, para finalmente enviarlo.

Es de resaltar que las transiciones o arcos “Http_Request” , “Http_responce” y el nodo o lugar Invoque “WS” de la Figura C-32 representan el servicio web de la bolsa de valores; por otra parte, el resto de transiciones/lugares y nodos/arcos representan el intercambio de información del contenedor JSLEE, el cual se encarga del manejo de los mensajes Sip y de formar las peticiones Http al servicio Web.

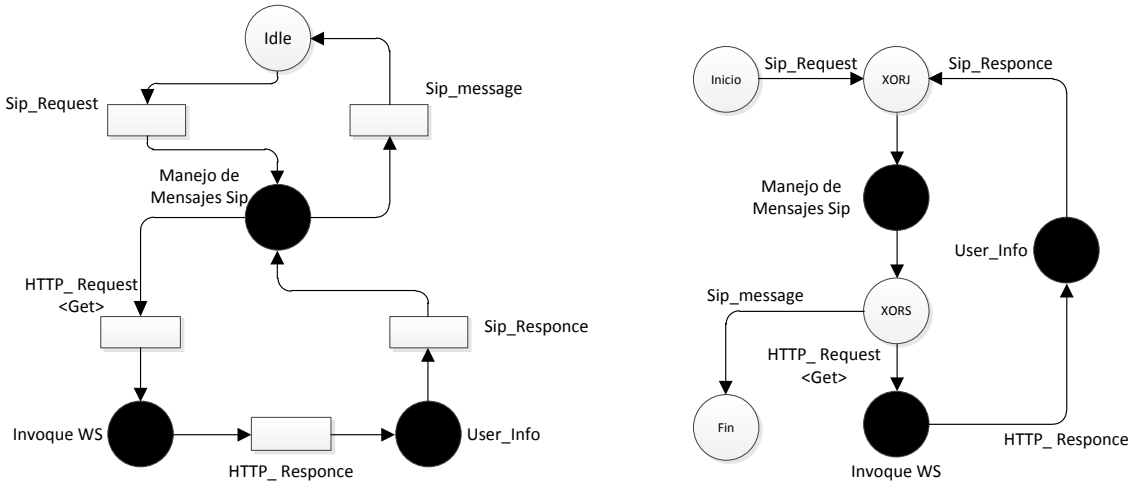


Figura C-32: Servicio FInancial Message representado mediante redes de Petri a) y Grafos b)

C.1.3.3 GeoClick2Dial

GeoClick2Dial (geo-localización + click to dial) está compuesto por un servicio web donde el usuario puede visualizar en un mapa la ubicación de sus contactos de Facebook, para realizar una llamada al que este desee.

El servicio Web de localización lo componen principalmente dos servicios Web, el primero de estos es implementado empleando el Api de Facebook, de donde se obtiene la ubicación de los amigos del usuario; y el segundo es implementado con el Api de Google Maps, el cual muestra la ubicación de los amigos de Facebook mediante marcadores.

Por otra parte el servicio click to dial se encuentra alojado en un servidor JainSlee, este recibe la información del servicio Web de la persona a la cual se desea llamar para conectar a las dos usuarios mediante el servicio de llamada.

La figura Figura C-33, muestra la representación del servicio en redes de Petri y Grafos respectivamente. Éste comienza con una petición HTTP para invocar el servicio Web donde puede visualizar en el mapa sus amigos de Facebook. Una vez el usuario haya dado click sobre el amigo al cual desea llamar, el servicio Web envía una respuesta Http al contenedor JSLEE, el cual procesa esta petición.

JSLEE se encarga de establecer la comunicación entre las dos partes, en caso de que una no conteste la invitación a hablar, el servicio Web se termina.

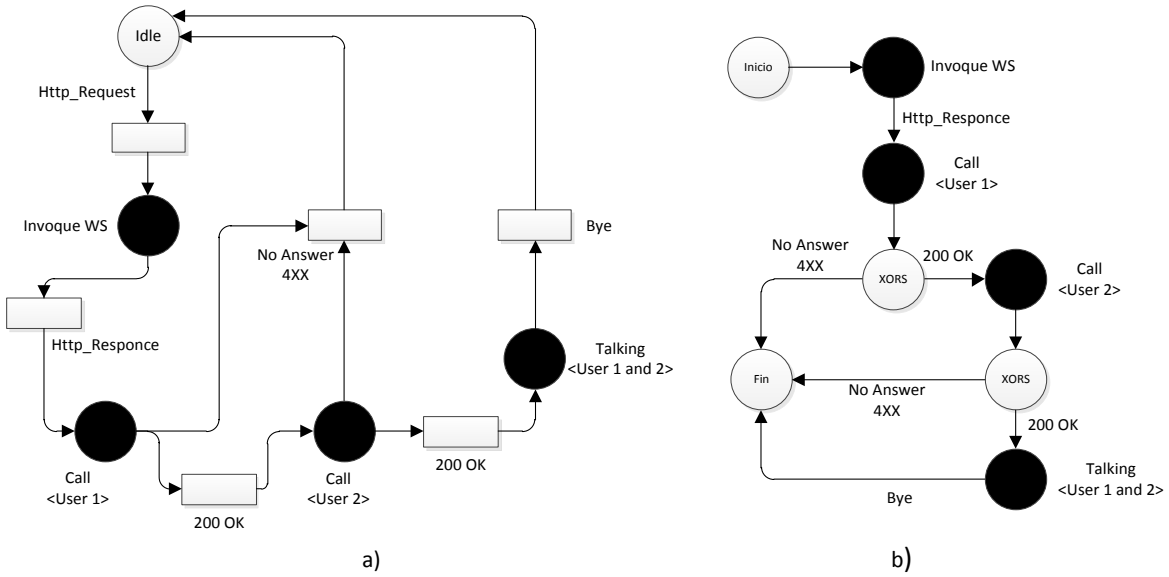


Figura C-33: Servicio click2dial representado mediante redes de Petri a) y Grafos b)

C.1.3.4 Travel Confirmation

El servicio travel confirmation, permite al usuario escoger su destino de viaje y reservar su vuelo. Una vez realizada la reserva el servicio, permite escoger al usuario dos formas de confirmación de su reserva, mediante una llamada o mediante SMS. Si el usuario escoge la confirmación mediante una llamada y al momento de realizarse la misma, este no contesta, el servicio envía la confirmación de la reserva mediante SMS.
La Figura C-34, muestra el servicio de confirmación de viaje representado mediante redes de petri y grafos.

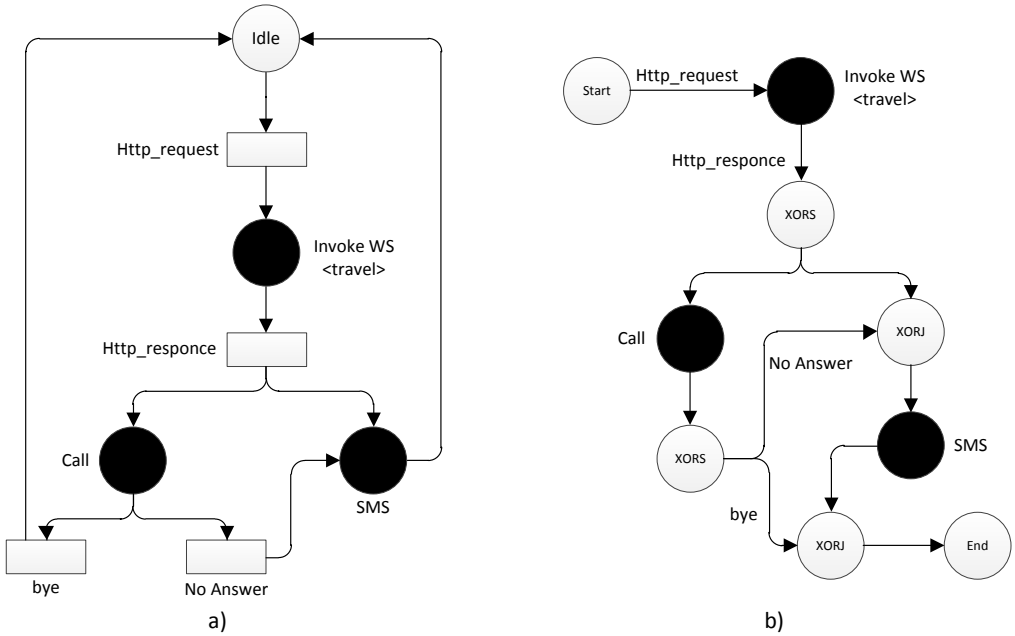


Figura C-34: Servicio travel confirmation representado mediante redes de Petri a) y Grafos)

C.1.3.5 Facebook Message Reminder + Call and SMS

El servicio representado mediante redes de Petri y Grafos en la Figura C-35, está compuesto por 3 servicios atomicos: Facebook, el cual hace uso de algunas funcionalidades del Api de acceso libre de Facebook[3]; y los servicios de llamada y SMS, soportados mediante la especificación JSLEE.
El servicio comienza cuando se detectan eventos cercanos, a los cuales un usuario registrado al servicio va a asistir, seguidamente, se invoca al servicio de presencia para conocer el estado (disponible, no disponible).Si el usuario se encuentra como disponible, el servicio realiza una llamada, para recordarle que en evento cercano al cual va a asistir, ya está cercano a comenzar; de lo contrario el servicio le envía un SMS.
EL servicio finaliza una vez que se hayan realizado una de las anteriores acciones.

C.1.3.6 Ebay Product Finder + Call

Este servicio presenta funcionalidades similares al servicio *Ebay Product Finder,* la diferencia está en la agregación del servicio de llamada y SMS, los cuales son utilizados para notificar que el pago se ha realizado satisfactoriamente. El servicio de llamada es disparado y si el usuario contesta, escuchara una grabación con la informacion del articulo que compro, de lo contrario se envía un SMS a su dispositivo móvil con dicha informacion.
La representación del servicio en redes de Petri y grafos se muestran en la Figura C-36 y Figura C-37 respectivamente.

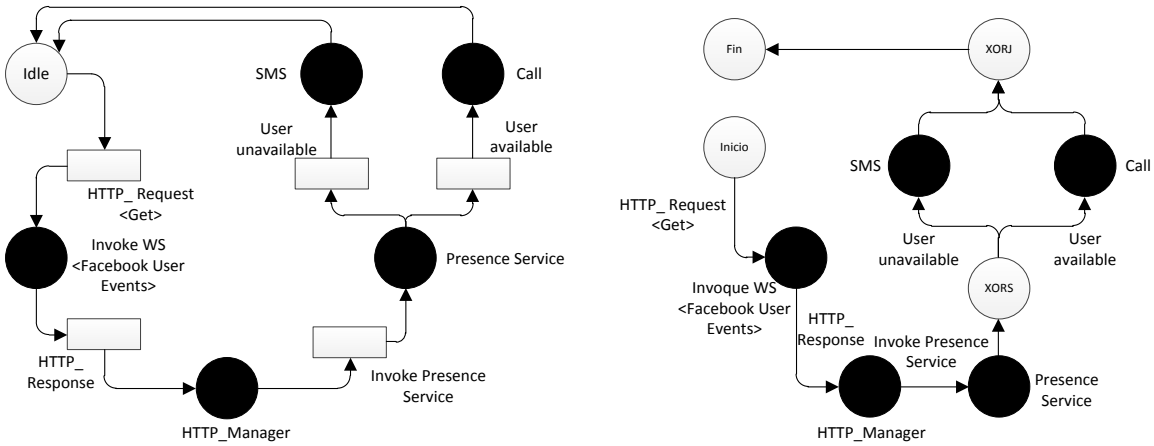


Figura C-35: Servicio Facebook Message Reminder + Call and SMS representado mediante redes de Petri a) Grafos b)

[3] Api Facebook: http://developers.facebook.com/docs/reference/php/facebook-api/

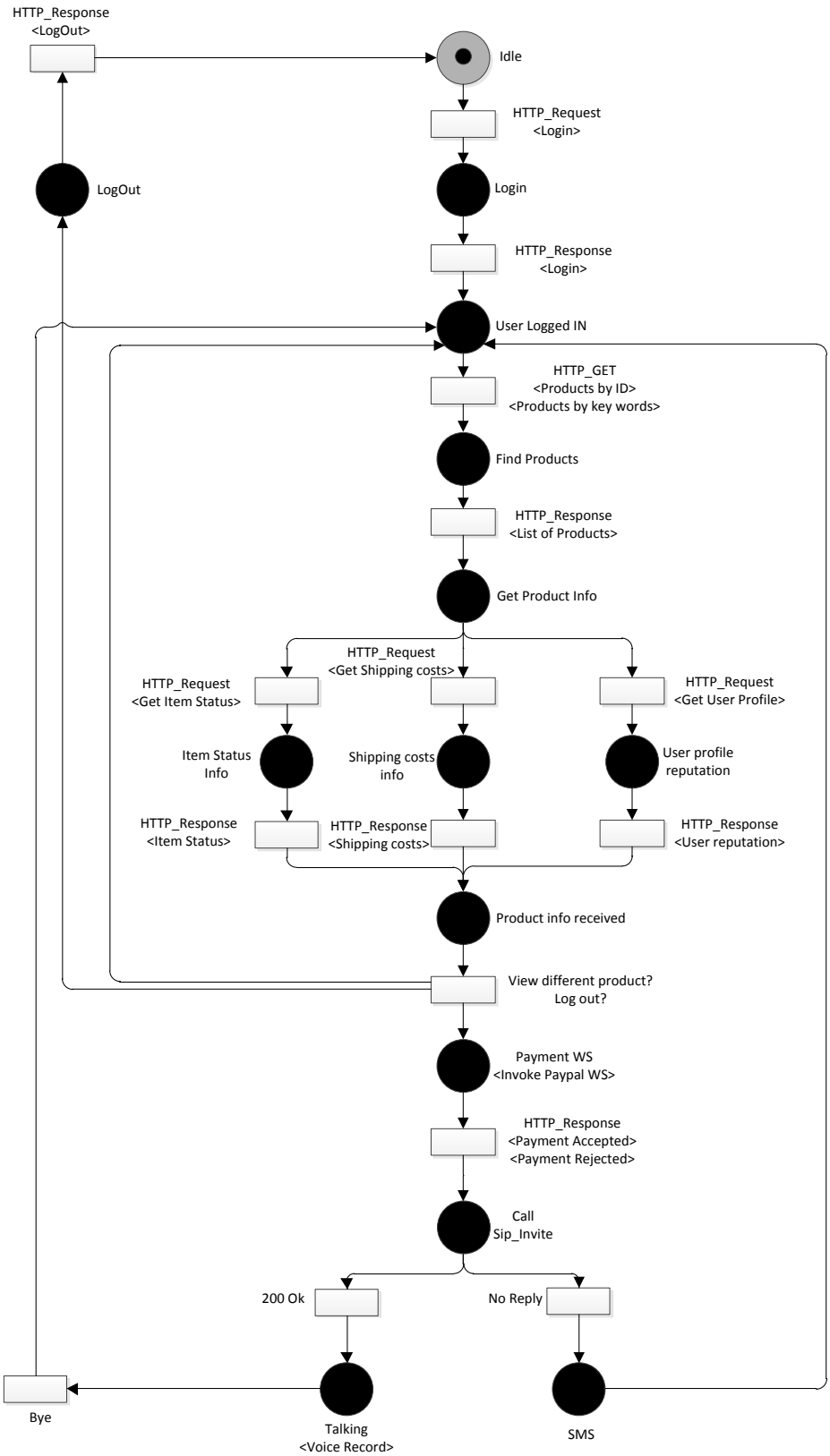


Figura C-36: Servicio Ebay product finder + Call representado mediante redes de Petri

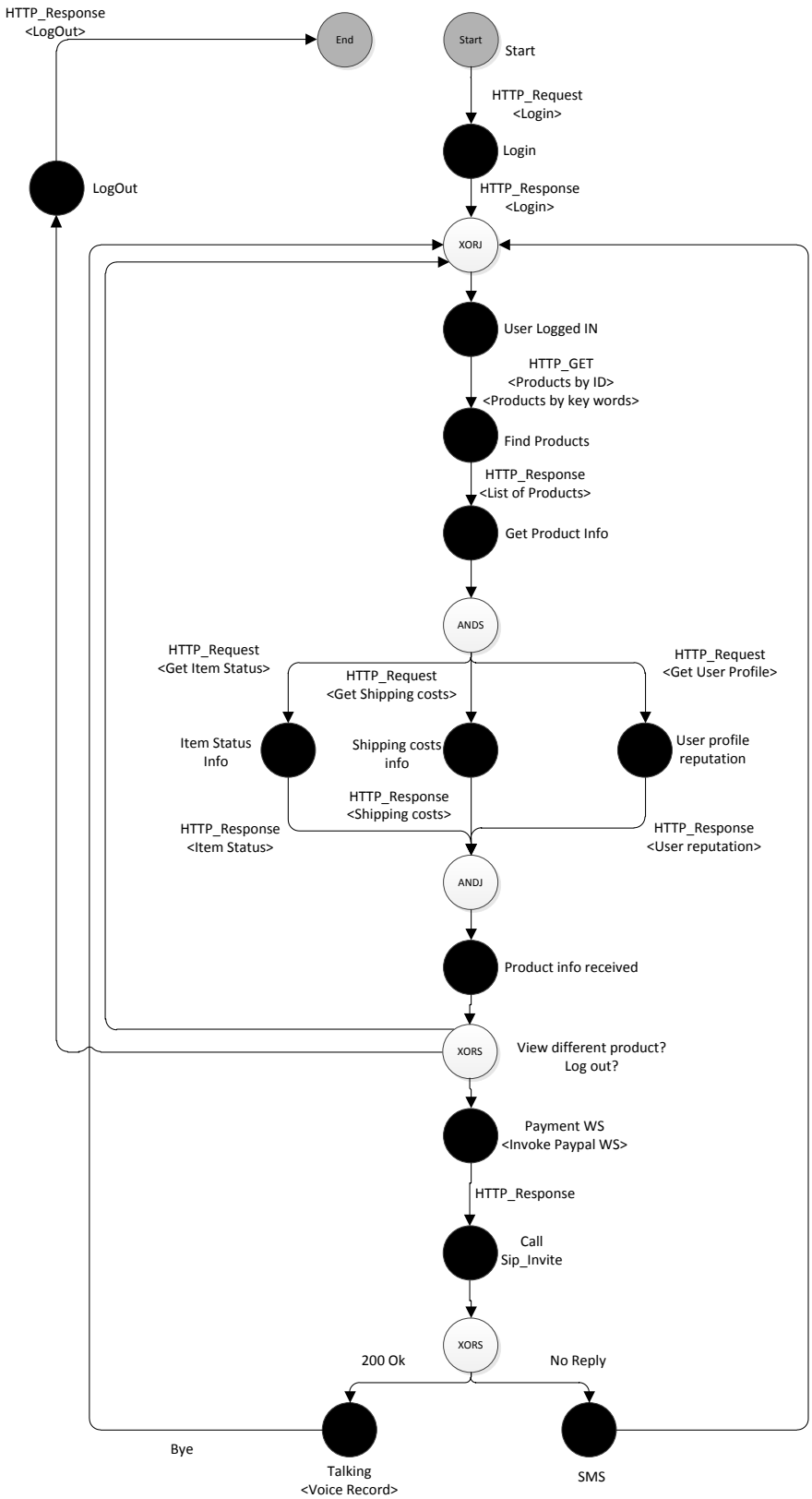
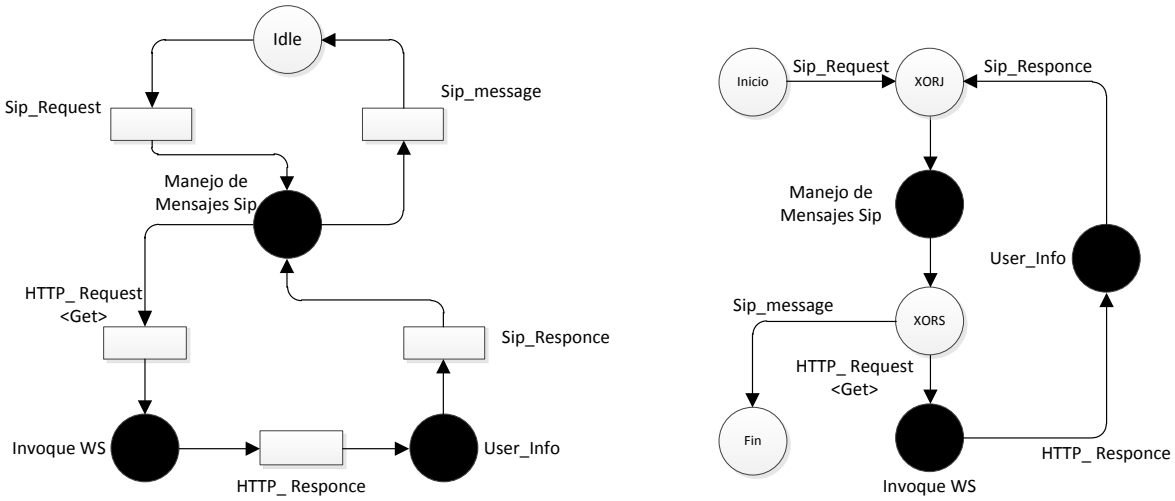


Figura C-37: Servicio Ebay product finder + Call representado mediante Grafos

C.1.3.7 Facebook Message Reminder



C.1.3.8 CallMessage

Éste servicio permite a un usuario realizar una llamada SIP y comunicarse con un usuario de destino. Si la llamada no es exitosa, el servicio envía automáticamente un mensaje directo de Twitter con la información de llamada, el cual será recibido por el usuario de destino. La representación en redes de Petri y Grafos se muestra en la Figura C-38.

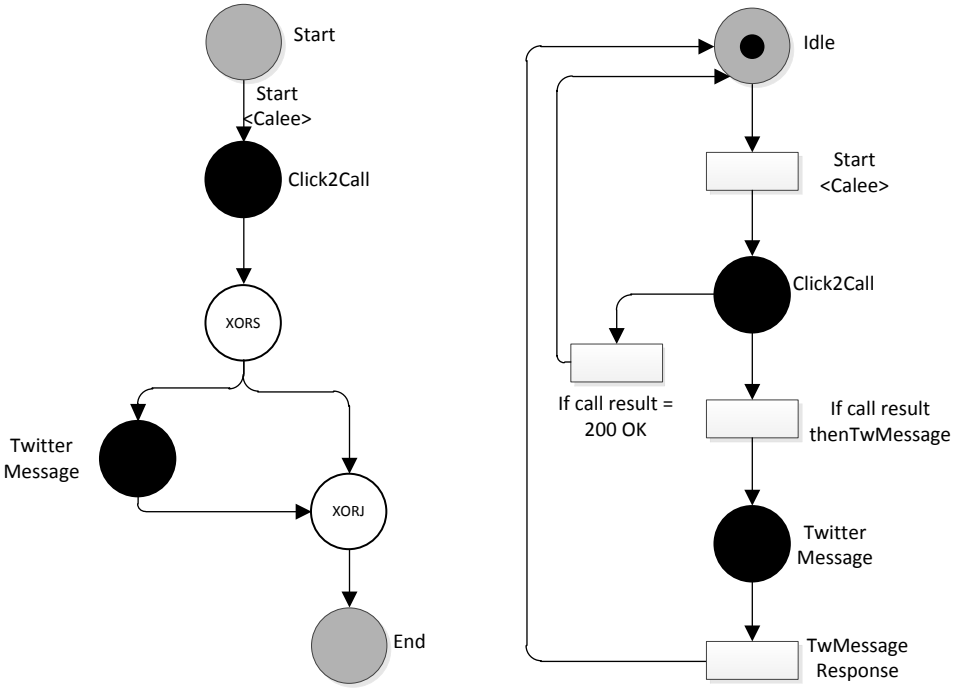


Figura C-38: Representacion en redes de Petri y Grafos del servicio CallMessage

C.1.3.9 TwitterSMS

Permite, enviar un SMS o un mensaje directo de Twitter a un usuario de destino. La representación en redes de Petri y Grafos del servicio TwitterSMS se muestra en la Figura C-39.

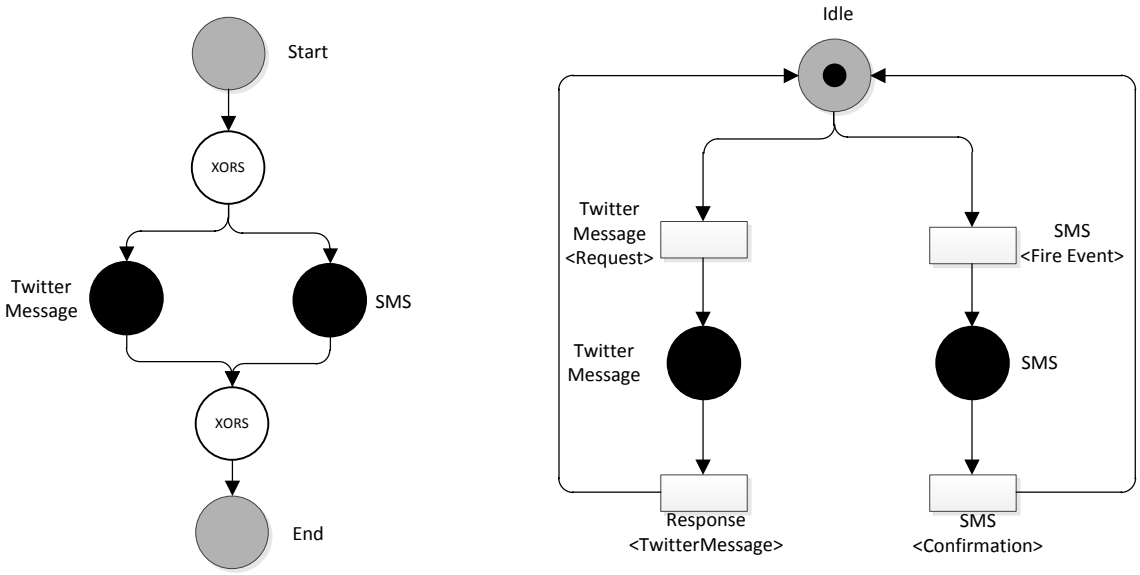


Figura C-39: Servicio TwitterSMS representado mediante redes de Petri y grafos

C.1.3.10 TwitterCalls

El servicio representado mediante redes de Petri y Grafos en la Figura C-40 y Figura C-41, está compuesto principalmente por cuatro servicios: Twitter[4], el cual puede ser implementado empleando el API de Twitter de libre acceso; Llamada, servicio implementado mediante la especificación JSLEE en el entorno de creación de servicios de Mobicents; el servicio de mensajería corta SMS; y finalmente el servicio de presencia; estos últimos, implementados también con JSLEE y Mobicents.
El servicio inicia cuando el usuario ingresa a un portal Web con su nombre de usuario y contraseña de twitter (adicionalmente debe aceptar unos permisos, que exige Twitter para que el portal use su información), inmediatamente el servicio lo registra en JSLEE y actualiza el estado de presencia.
Seguidamente se le muestra al usuario una interfaz donde puede observar a las personas que lo siguen y que el sigue, si solo si, se han registrado en el portal. En dicha interfaz el usuario encontrara tres opciones:

- Llamar a un seguidor: El usuario podrá llamar a un seguidor que se encuentre registrado en el portal y se encuentre disponible teniendo en cuenta el estado de presencia, de lo contrario esta acción no se podra realizar.

[4] Api Rest Twitter: https://dev.twitter.com/docs/api

Si se cumplen las anteriores condiciones, mediante el servicio Click2Dial, el usuario podrá comunicarse, sin embargo, si dado el caso que la llamada no sea respondida, el servicio automáticamente le enviara un TwiterMessage (Mensaje de twitter) y un SMS a su dispositivo, que dirá: “El usuario ‘Pedrito’ intento comunicarse contigo a las ‘00:00’ horas”.

- Enviar un Tweet: El usuario podrá enviar un Tweet.
- Enviar un TwitterMessage: El usuario podrá enviar un mensaje interno de Twitter.

Finalmente, cuando el usuario desea salir de la aplicación el servicio de presencia se actualiza, dando por terminado el servicio.

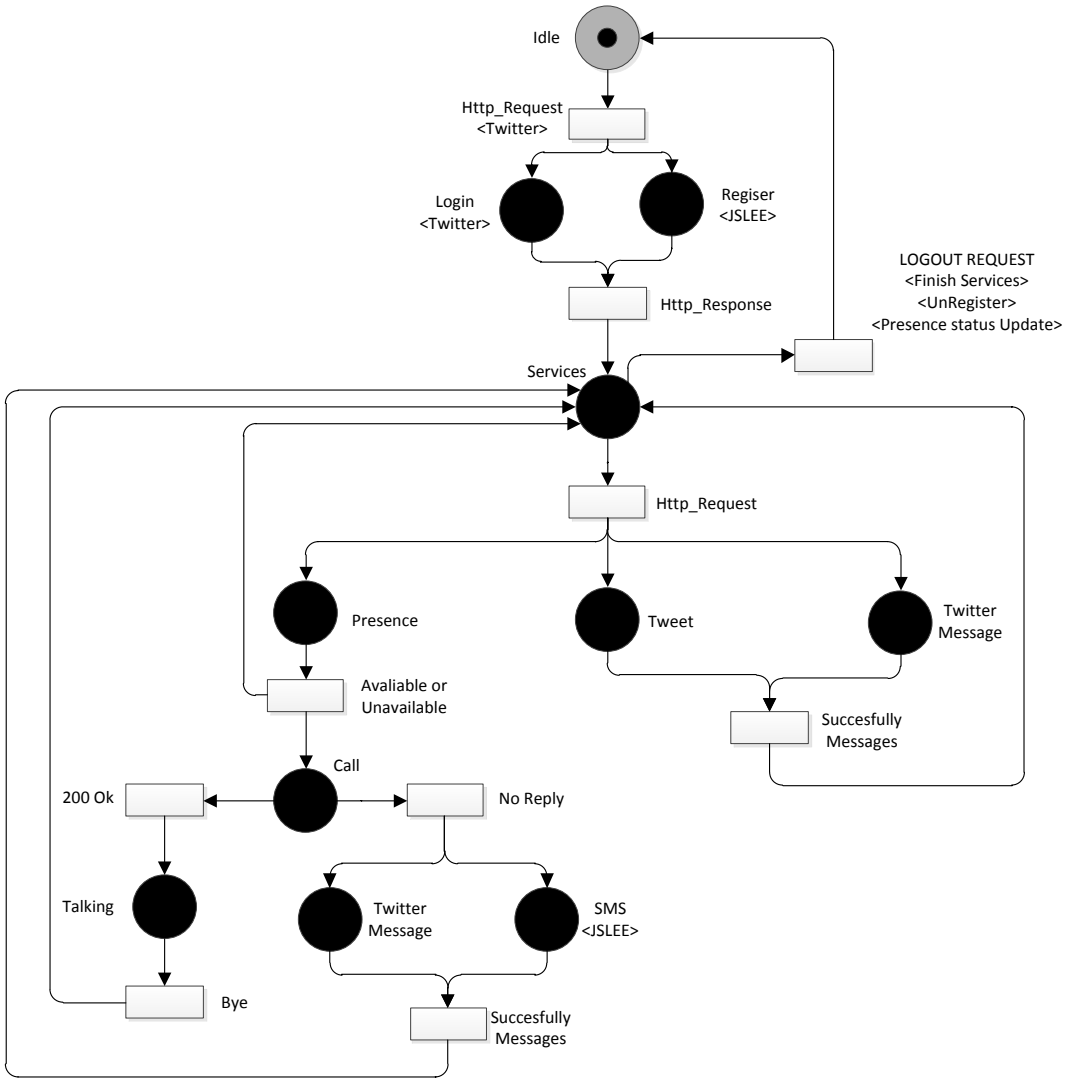


Figura C-40: Servicio Twitter Converged representado mediante redes de Petri

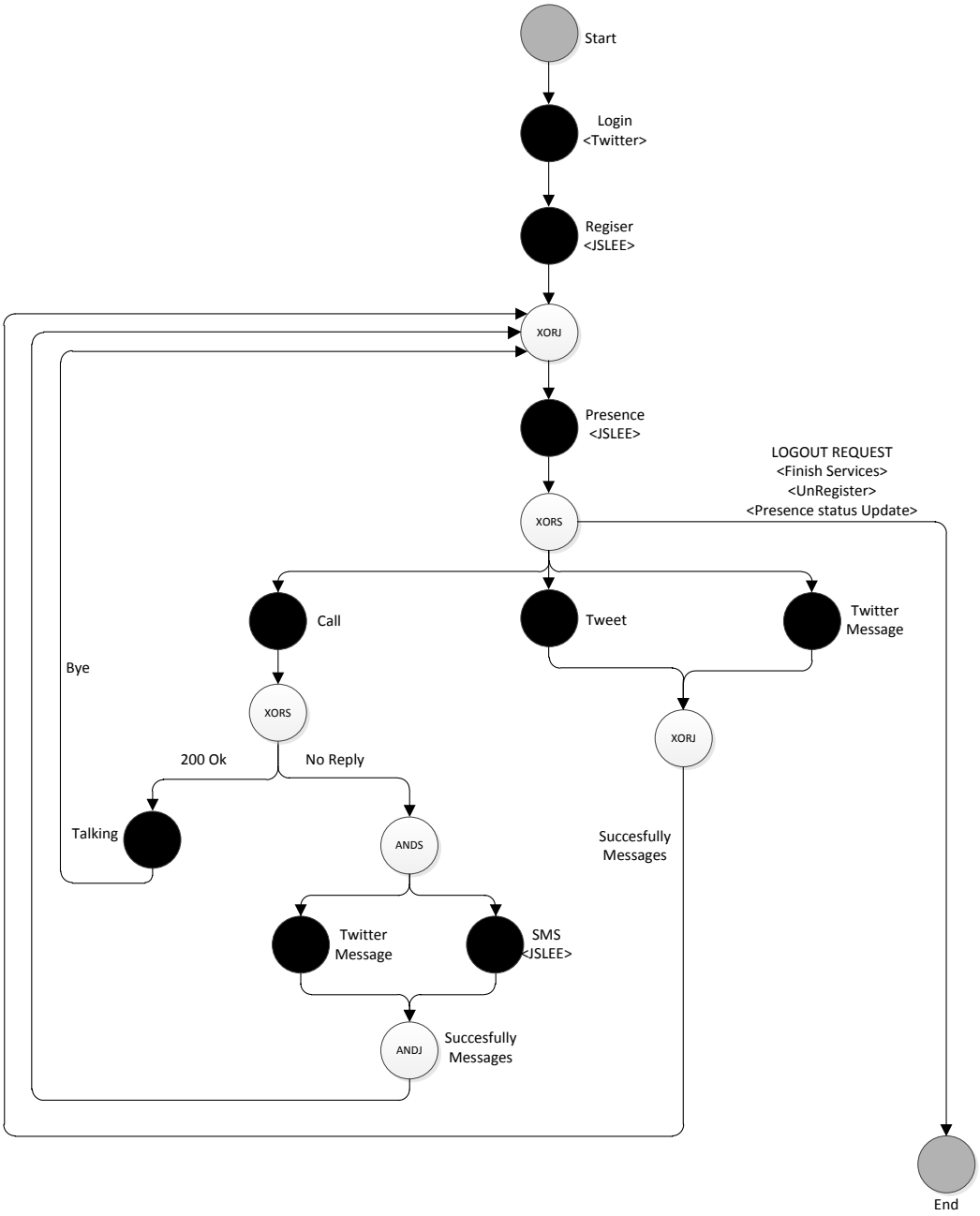


Figura C-41: Servicio Twitter Converged representado mediante Grafos

C.1.4 Total de servicios y patrones de flujo de control Detectados

C.1.4.1 Servicios

Tabla C-4: Total de servicios modelados

Servicios Telco	1	Llamada básica
	2	SMS
	3	Llamada + Desvío Incondicional
	4	Llamada + Desvío en Caso de no Respuesta
	5	Llamada en Espera
	6	Llamada en Espera + Transferencia de Llamada
	7	Llamada en Espera + Conferencia
	8	Llamada en Espera + Conferencia + Transferencia de Llamada
	9	Llamada + SMS en caso de ausencia de Respuesta
	10	Llamada + actualización de presencia.
Servicios Web	11	SMS Web
	12	MMS
	13	Ubicación
	14	Capacidades del dispositivo
	15	Pagos
	16	Perfil de conexión de datos
	17	Servicio Skype
	18	Servicio de Twitter
	19	Servicio de Facebook
	20	Amazon – Control de Vendedor
Servicios Convergentes	21	Compras de Mobicents
	22	Mensaje Financiero
	23	Localización + Click2Call
	24	Viajes + Confirmación Sms
	25	Recordatorio de eventos de Facebook con SMS
	26	Recordatorio de eventos de Facebook con SMS y Llamada
	27	EbayProductFinder + Llamada
	28	TwitterSMS (Envio de SMS y Tuits)
	29	CallMessage (Envio de mensajes directos de Twitter + Llamada)
	30	TwitterCalls (Llamada + Envio de SMS + Tuits + Estado de Presencia)

C.1.4.2 Patrones de consulta

Los patrones de consulta corresponden a los 21 patrones mostrados en la Tabla A-2.

C.1.4.3 Patrones Detectados en los servicios modelados

La Tabla C-5 muestra los patrones detectados en los servicios modelados.

Tabla C-5: Total de Patrones Detectados en los servicios modelados

ID Servicio	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	total
1				1	2						1					1		1				**6**
2				2	1											1		1				**5**
3				1	1						1					1		1				**5**
4				1	2						1					1	1	1				**7**
5		4		2	7						5				2	1	1	2				**24**
6		4		2	9						6				3	1	1	3				**29**
7		4		2	8						6				3	1	1	2				**27**
8		4		2	10						7				3	1	1	3				**31**
9				2	2						1					1		2				**8**
10		1	1						1							1						**4**
11					1						1					1						**3**
12					1						1					1						**3**
13					1						1					1						**3**
14					1						1					1						**3**
15					1						1					1						**3**
16					1						1					1						**3**
17				1	1	1		1								1	1					**6**
18				1	1	1		1								1	1					**6**
19				1	1	1		1								1	1					**6**
20	1	1	1	1	1				1		1					1	1					**9**
21					1						4					1						**6**
22				1	1											1	1					**4**
23				2	1											1						**4**
24				2	1						1					1						**5**
25				1							1					1						**3**
26	1			1							1					1						**4**
27	1	1	1		1						2					1	1					**8**
28				1							1					1						**3**
29				1							1					1						**3**
30		1	1	1	1	1			1		2					1	1					**10**
TOTAL	**3**	**20**	**4**	**29**	**58**	**4**	**0**	**3**	**3**	**0**	**48**	**0**	**0**	**0**	**11**	**30**	**12**	**16**	**0**	**0**	**0**	**241**

Anexo D

D Descripción e interfaces graficas de los servicios de prueba

En éste anexo se exponen las interfaces graficas de tres servicios creados con la herramienta GT-4SC, de igual forma se describe su funcionamiento y algunas consideraciones para cada servicio.

D.1 Servicio TwittSMS

D.1.1 Descripción del servicio

Éste servicio le permite a un usuario, enviar un SMS o un mensaje directo de Twitter a un usuario de destino.

D.1.2 Consideraciones

Tanto el usuario emisor como el receptor, deben estar registrados en la base de datos de la aplicación, adicionalmente el usuario receptor debe seguir en Twitter al usuario emisor, debido a que solo de esta forma los mensajes directos de Twitter son autorizados por la red social

D.1.3 Implementacion en la herramienta GT-4SC

La implementación del servicio TwittSMS se observa en la **Figura D-1**. La cual comprende el uso de los servicios básicos: *TwMessage* y *SMS*; y los patrones de flujo de control: *Deferred Choice* y *Simple Merge*.

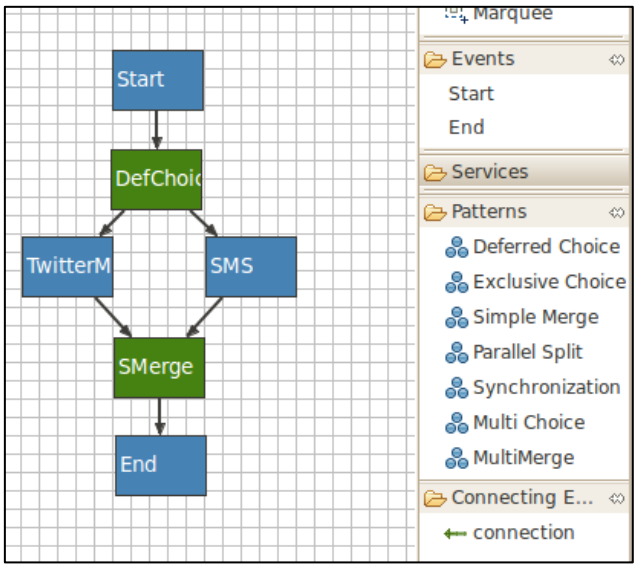


Figura D-1: Composicion del servicioTwittSMS en GT-4SC

D.1.4 Interfaces de usuario

Interfaz principal: en esta interfaz, (Figura D-2) el usuario emisor ingresa su *Twitter screen ID* (ej: @UsuarioEmisor) y el del usuario de destino (ej: @UsuarioDestino) en los campos de texto *From* y *To* respectivamente. Posteriormente el usuario decide entre enviar un mensaje directo de Twitter y un SMS.

Interfaz de respuesta: interfaz presentada al usuario después de enviar un mensaje directo de Twitter o un SMS (Figura D-3).

Figura D-2: Interfaz usuario - TwittSMS

Figura D-3: Interfaz de respuesta - TwittSMS

D.2 Servicio CallMessage

D.2.1 Descripción del servicio

Éste servicio permite a un usuario realizar una llamada SIP y comunicarse con un usuario de destino. Si la llamada no es exitosa, el servicio envía automáticamente un mensaje directo de Twitter con la información de llamada, el cual será recibido por el usuario de destino.

D.2.2 Consideraciones

Tanto el usuario que origina la llamada como el de destino, deben estar registrados en la base de datos de la aplicación, adicionalmente los dos usuarios deben estar registrados en un terminal SIP (softphone), con el fin de recibir los eventos SIP originados desde la interfaz web; finalmente, el usuario de destino debe seguir en Twitter al usuario que origina la llamada, debido a que solo de esta forma los mensajes directos de Twitter son autorizados por la red social.

D.2.3 Implementacion en la herramienta GT-4SC

La implementación del servicio CallMessage se observa en la Figura D-4 La cual comprende el uso de los servicios básicos: *C2C* y *TwMessage*; y los patrones de flujo de control: *Exclusive Choice* y *Simple Merge*.

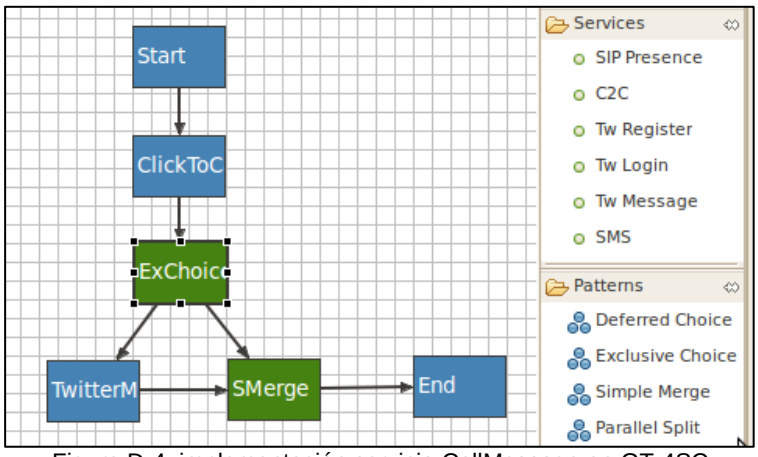


Figura D-4: implementación servicio CallMessage en GT-4SC

D.2.4 Interfaces de usuario

Interfaz principal: en esta interfaz, (Figura D-5) el usuario que origina la llamada ingresa su Twitte screen ID (ej: @UsuarioEmisor) y el del usuario de destino (ej: @UsuarioDestino) en los campos de texto From y To respectivamente. Posteriormente el usuario presiona el botón Call para originar una llamada SIP y establecer la comunicación con el usuario de destino.

Interfaz de respuesta: interfaz presentada al usuario después de originar una llamada SIP (Figura D-6)

Figura D-5: Interfaz de usuario CallMessage

Figura D-6: Interfaz de usuario CallMessage

D.3 Servicio Twitter Calls

D.3.1 Descripción del servicio

Éste servicio permite a un usuario registrarse al servicio TwitterCalls y en caso de haberse registrado anteriormente ingresar a esta utilizando su Twitter Screen ID.
Una vez haya ingresado a la aplicación, el usuario podrá seleccionar a un usuario de destino y enviarle un SMS, un mensaje directo de Twitter o llamarlo. Si el servicio de llamada SIP fue el escogido y la comunicación no es exitosa, el servicio TwitterCalls enviará un SMS y un mensaje directo de Twitter al usuario de destino con la información de la llamada.

D.3.2 Consideraciones

Ninguna.

D.3.3 Implementación en la Herramienta

La implementación del servicio CallMessage se observa en la Figura D-7, la cual comprende el uso de los servicios básicos: *TwRegister*, *Twlogin*; *TwMessage, SMS, SipPresence, C2C* y los patrones de flujo de control: *Exclusive Choice, Deferred Choice, Parallel Split, Simple Merge*, Synchronizastion, MultiChoice.

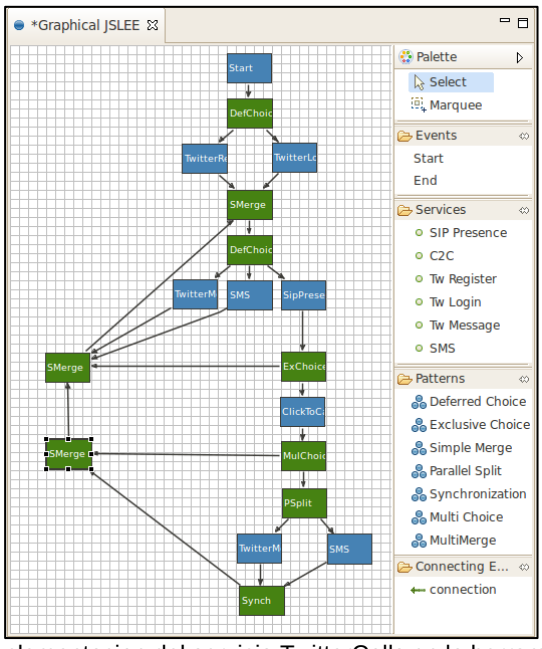


Figura D-7: Implementacion del servicio TwitterCalls en la herramienta GT-4SC

D.3.4 Interfaces de usuario

Interfaz de inicio: la interfaz de inicio (Figura D-8), permite al usuario registrarse a la aplicación, si el usuario ya se ha registrado previamente, podrá ingresar haciendo uso de su Twitter Screen IDn (ej: @usuario).

Interfaz de registro: el registro a la aplicación se realiza directamente con la red social Twitter como se muestra en la Figura D-9, por lo cual el servicio TwitterCalls no tendrá acceso a las contraseñas de los usuarios. Una vez registrado, el usuario ingresará a la aplicación donde se encontrará con la interfaz de usuarios.

Interfaz de usuarios: en la interfaz que se muestra en la Figura D-10 el usuario podrá encontrar una lista de amigos que están registrados en la aplicación y al mismo tiempo lo siguen en Twitter y haciendo click sobre la foto de cada uno de ellos podrá accederé a varias opciones o servicios que ofrece TwitterCalls.

Interfaz usuario de destino: muestra los datos de un usuario seleccionado (foto, twitter screen ID, twitter ID, estado SIP) y ofrece tres opciones o servicios: enviar un mensaje directo de Twitter, enviar un SMS o realizar una llamada SIP. Lo anterior se ilustra en la Figura D-11.

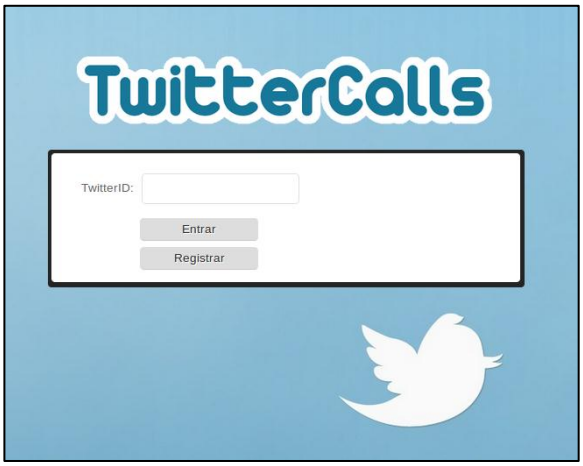


Figura D-8: Interfaz usuario - TwitterCalls

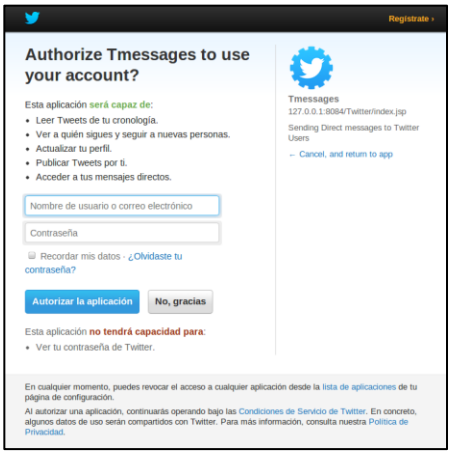


Figura D-9: Interfaz de registro - TwitterCalls

Figura D-10: Interfaz de usuarios - TwitterCalls

Figura D-11: Interfaz de usuarios de destino TitterCalls

D.3.5 Ejemplo de Funcionamiento

Una vez el usuario ha ingresado a la aplicación (usuario logueado o registrado), y ha escogido un usuario de destino, podrá hacer uso de tres funcionalidades o servicios: enviar un SMS, un mensaje directo de Twitter o realizar una llamada SIP. A continuación se dará un ejemplo funcional de estos servicios tomando como usuario de prueba a @TgaTest.

Para enviar un mensaje directo de twitter (Figura D-12) el usuario ingresa su mensaje en el campo de texto y hace click sobre el botón TwMessage. En este caso, el mensaje será enviado al buzón de mensajes de Twitter del usuario @GustavoEnriquez.

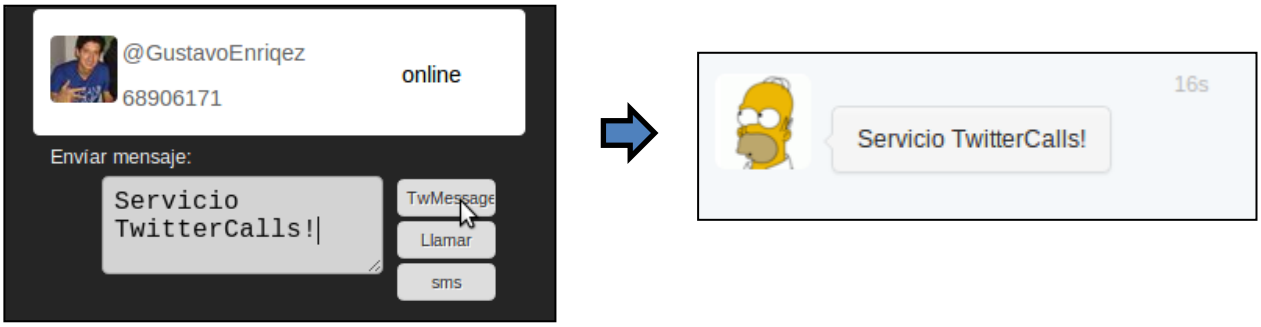


Figura D-12: TwitterCalls – Mensaje Directo

Para enviar un SMS Figura el usuario ingresa su mensaje en el campo de texto y hace click sobre el botón SMS. Como puede observase en la **Figura D-13** el mensaje y el numero del usuario de destino llegan a la servicio SMS y este lo envía por medio de un routerSMS (Teléfono móvil con aplicación de envío de mensajes).

```
::::Datos Recibidos en Servicio SMS!:
3006191739 El servicio Twitter Calls le informa que el usuario @TgaTest le hizo una llamada el viernes, 05 de octubre de 2012 a las 6:25PM
```

Figura D-13: TwitterCalls - SMS

Para realizar una llamada, los dos usuario deben estar registrados en un terminal SIP como se observa en la Figura D-14, en este caso se utilizan dos clientes sip de prueba (Twinkle y xlite) para registrar a los usuarios @TgaTest y @GustavoEnriqez

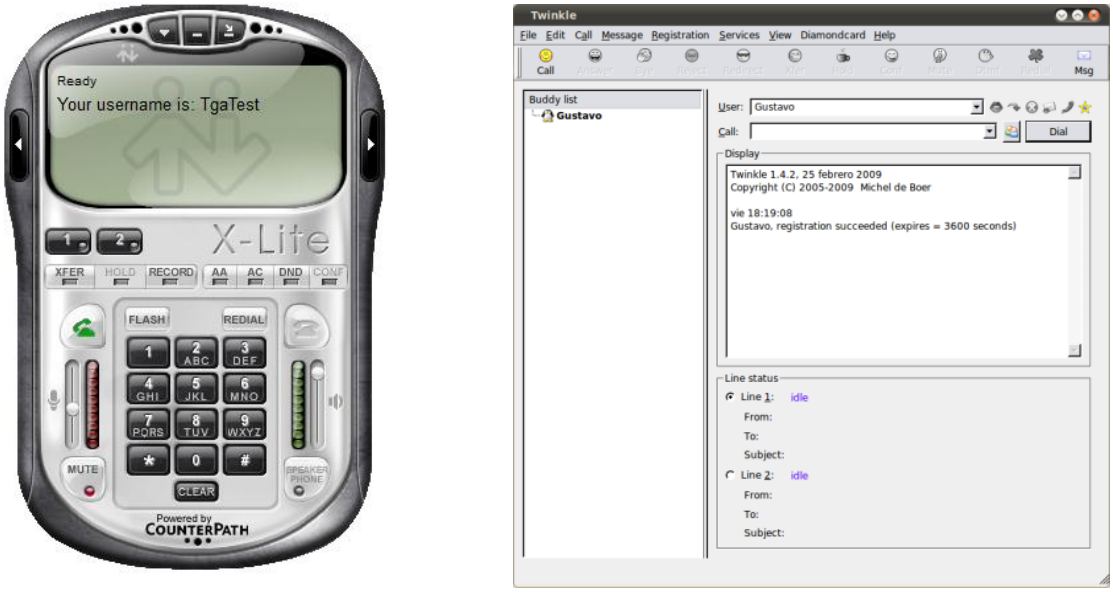


Figura D-14: TwitterCalls – Llamada (a)

Una ves el usuario @TgaTest haga click sobre el botón llamar, el usuario @GustavoEnriqez recibirá la notificación y si la llamada es respondida, el servicio redirige la llamada hacia el usuario @TgaTest informándole que se puede establecer la comunicación, lo anterior se puede observar en las Figura D-15 y Figura D-16.

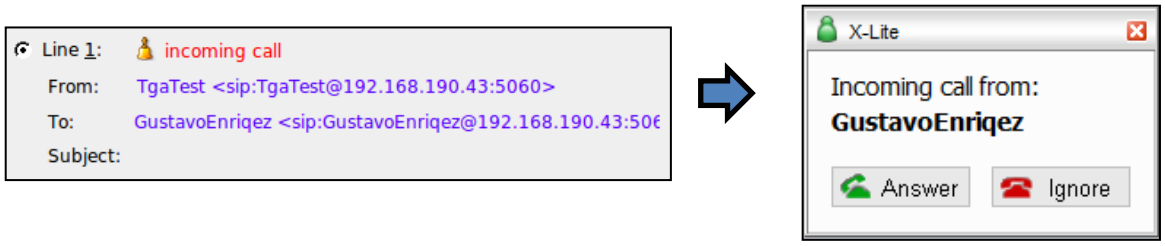


Figura D-15: TwitterCalls – Llamada (b)

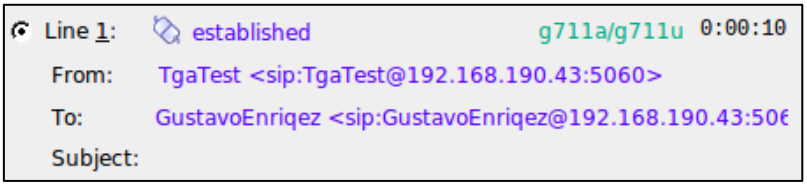


Figura D-16: TwitterCalls – Llamada (c)

Si después de cierto tiempo, el usuario de destino no responde a la solicitud de llamada, un SMS y un mensaje directo de twitter le es enviado, con la información de llamada (hora, fecha, usuario de destino) como se observa en la Figura D-17

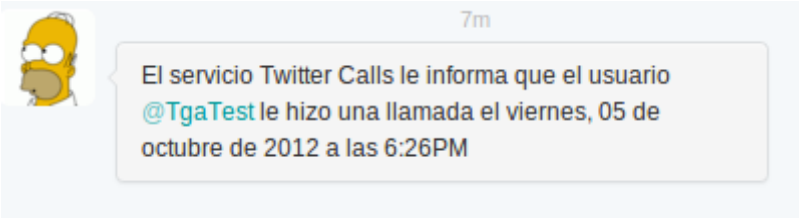


Figura D-17: TwitterCalls – Llamada no contestada

Anexo E

E Comparación del soporte de Patrones de Flujo de Ejecución.

En este Anexo se presenta el soporte de patrones de flujo de ejecución del prototipo funcional desarrollado en el presente trabajo de grado, y se realiza una comparación con el soporte de dos lenguajes: JBPM y BPEL, los cuales han sido utilizados para la composición de servicios Telco 2.0 en [7] [8] [9].

En la Tabla E-1, se muestran los patrones de flujo de control soportados por BPEL, JBPM y GT-4SC. Como se puede observar, GT-4SC soporta directamente un mayor número de patrones (21) en comparación a BPEL (16) y JBPM (13).

En la Tabla E-2, se muestran los patrones de datos BPEL, JBPM y GT-4SC. Como se puede observar, GT-4SC soporta directamente un mayor número de patrones de datos (16), en comparación a BPEL (13) y JBPM (5).

Tabla E-1: Patrones de Flujo de Control Soportados por BPEL, JBPM y GT-4SC

Patrones de Flujo de Control	**Viabilidad de Implementación***		
Patrones Básicos	BPEL	JBPM	GT-4SC
Sequence	+	+	+
Parallel split	+	+	+
Synchronization	+	+	+
Exclusive choice	+	+	+
Simple merge	+	+	+
Total	5	5	5
Patrones de Ramificación y Sincronización Avanzada			
Multi-choice	+	-	+
Structured synchronizing merge	+	-	+[5]
Multi-merge	-	+	+
Structured discriminator	-	-	-
Blocking discriminator	-	-	-
Cancelling discriminator	-	-	-
Structured partial join	-	-	-
Blocking partial join	-	-	-
Cancelling Partial Join	-	-	-
Generalized AND-join	-	+	+[6]
Local synchronizing merge	+	-	+[7]
General synchronizing merge	-	-	-
Thread merge	+/-	+/-	-
Thread split	+/-	+/-	-
Total	3	2	5
Patrones Basados en Estados			
Deferred Choice	-	+	+
Interleaved Parallel Routing	+/-	-	-
Milestone	-	-	-
Critical section	+	-	+[8]
Interleaved Routing	+	-	+
Total	2	1	3
Patrones de Múltiples Instancias			
Multiple instances without synchronization	+	+	+
Multiple instances with a prior design-time knowledge	-	-	-
Multiple instances with a prior run-time knowledge	-	-	-
Multiple instances without a prior run-time knowledge	-	-	-
Static partial join for multiple instances	-	-	-
Cancelling partial Join for multiple instances	-	-	-
Dynamic partial Join for multiple instances	-	-	-
Total	1	1	1
Patrones de Cancelación y Terminación Forzada			
Cancel task	+	+	+
Cancel case	+	-	+
Cancel region	+/-	-	-
Cancel multiple instance activity	-	-	-
Complete multiple instance activity	-	-	-
Total	2	1	2
Patrones de Iteración			
Arbitrary cycles	-	+	+
Structured loop	+	-	+
Recursion	-	-	-
Total	1	1	2

[5]Structured synchronizing merge: GT-4SC lo soporta mediante la union de los patrones Multi Choice y Synchronization.
[6]Generalized AND-join: GT-4SC lo soporta mediante la unión de los patrones Parallel split y Synchronization.
[7]Local synchronizing merge: GT-4SC lo soporta mediante la union de los patrones Multi Choice y Synchronization.
[8]Critical Section: GT-4SC lo soporta directamente mediante los patrones Parallel Split y Synchronization, sin embargo, es necesario definir en el código JAIN SLEE el orden de ejecución de las tareas.

Patrones de Terminación			
Implicit termination	+	+	+
Explicit termination	-	-	-
Total	1	1	+
Patrones de Activación			
Transient trigger	-	+	+[9]
Persistent Trigger	+	-	+[10]
Total	1	1	2
TOTAL DE PATRONES	**16**	**13**	**21**
* **+** Si lo soporta; **+/-** Lo soporta con limitantes; **–** No lo soporta			

Tabla E-2: Parones de de Datos Soportados por BPEL, JBPM y GT-4SC

Patrones de Datos	**Viabilidad de Implementación***		
Data Visibility	BPEL	JBPM	GT-4SC
Task data	+/-	+/-	-
Block data	-	-	-
Scope Data	+	-	+
Multiple Instance Data	-	-	-
Case Data	+	+	+
WorkFlow Data	-	-	-
Environment Data	+	+/-	+
Total	3	1	3
Internal Data Interaction			
Data Interaction between Tasks	+	+	+
Data Interaction - Block Task to Sub-workflow Decomposition	-	-	-
Data Interaction - Sub-workflow Decomposition to Block Task	-	-	-
Data Interaction - to Multiple InstanceTask	-	-	-
Data Interaction - from Multiple Instance Task	-	-	-
Data Interaction - Case to Case	+/-	+/-	-
Total	1	1	1
External Data Interaction			
Data Interaction - Task to Environment - Push-Oriented	+	+/-	+
Data Interaction - Environment to Task - Pull-Oriented	+	+/-	+
Data Interaction - Environment to Task - Push-Oriented	+/-	+/-	-
Data Interaction - Task to Environment - Pull-Oriented	+	-	+
Data Interaction - Case to Environment - Push-Oriented	-	-	-
Data Interaction - Environment to Case - Pull-Oriented	-	-	-
Data Interaction - Environment to Case - Push-Oriented	-	-	-
Data Interaction - Case to Environment - Pull-Oriented	-	-	-
Data Interaction - Workflow to Environment - Push-Oriented	-	-	-
Data Interaction - Environment to Workflow - Pull-Oriented	-	-	-
Data Interaction - Environment to Workflow - Push-Oriented	-	-	-
Data Interaction - Workflow to Environment - Pull-Oriented	-	-	-
Total	3	0	3
Data Transfer			
Data Passing by Value – Incoming	+	-	+
Data Passing by Value – Outgoing	+	-	+
Data Passing - Copy In/Copy Out	-	+	+
Data Passing by Reference - Unlocked	+	-	+
Data Passing by Reference - Locked	+/-	-	-
Data Transformation - Input	-	+	+
Data Transformation – Output	-	+	+
Total	3	3	6

[9]Transient Trigger: GT-4SC lo soporta mediante el patrón Syncronization y Exclusive Choice
[10]Persistent Trigger: GT-4SC lo soporta mediante el patrón Syncronization y Exclusive Choice

Data Routing Patterns			
Task Precondition - Data Existence	+/-	-	-
Task Precondition - Data Value	+	-	+
Task Postcondition - Data Existence	-	-	-
Task Postcondition - Data Value	-	-	-
Event-based Task Trigger	+	-	+
Data-based Task Trigger	+/-	-	-
Data-based Routing	+	+/-	+
Total	3	0	3
TOTAL	**13**	**5**	**16**
* **+** Si lo soporta; **+/-** Lo soporta con limitantes; **–** No lo soporta			

Anexo E

F Valores para la distribución t-Student

Tabla t-Student

Grados de libertad	0.25	0.1	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3137	12.7062	31.8210	63.6559
2	0.8165	1.8856	2.9200	4.3027	6.9645	9.9250
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8408
4	0.7407	1.5332	2.1318	2.7765	3.7469	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0321
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3646	2.9979	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.1448	2.6245	2.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467
16	0.6901	1.3368	1.7459	2.1199	2.5835	2.9208
17	0.6892	1.3334	1.7396	2.1098	2.5669	2.8982
18	0.6884	1.3304	1.7341	2.1009	2.5524	2.8784
19	0.6876	1.3277	1.7291	2.0930	2.5395	2.8609
20	0.6870	1.3253	1.7247	2.0860	2.5280	2.8453
21	0.6864	1.3232	1.7207	2.0796	2.5176	2.8314
22	0.6858	1.3212	1.7171	2.0739	2.5083	2.8188
23	0.6853	1.3195	1.7139	2.0687	2.4999	2.8073
24	0.6848	1.3178	1.7109	2.0639	2.4922	2.7970
25	0.6844	1.3163	1.7081	2.0595	2.4851	2.7874
26	0.6840	1.3150	1.7056	2.0555	2.4786	2.7787
27	0.6837	1.3137	1.7033	2.0518	2.4727	2.7707
28	0.6834	1.3125	1.7011	2.0484	2.4671	2.7633
29	0.6830	1.3114	1.6991	2.0452	2.4620	2.7564
30	0.6828	1.3104	1.6973	2.0423	2.4573	2.7500
31	0.6825	1.3095	1.6955	2.0395	2.4528	2.7440
32	0.6822	1.3086	1.6939	2.0369	2.4487	2.7385
33	0.6820	1.3077	1.6924	2.0345	2.4448	2.7333
34	0.6818	1.3070	1.6909	2.0322	2.4411	2.7284
35	0.6816	1.3062	1.6896	2.0301	2.4377	2.7238
36	0.6814	1.3055	1.6883	2.0281	2.4345	2.7195
37	0.6812	1.3049	1.6871	2.0262	2.4314	2.7154
38	0.6810	1.3042	1.6860	2.0244	2.4286	2.7116
39	0.6808	1.3036	1.6849	2.0227	2.4258	2.7079
40	0.6807	1.3031	1.6839	2.0211	2.4233	2.7045

Referencias

[1] W. M. P. van der Aalst, A. Hofstede, N. Russell, and N. Mulyar, “Workflow Control-Flow Patterns, A Revised View,” in *BPM Center Report BPM-06-22*, 2006.

[2] P. Wohed, B. Andersson, A. H. Hofstede, N. Russell, and W. M. P. van der Aalst, “Patterns-based Evaluation of Open Source BPM Systems : The Cases of jBPM , OpenWFE , and Enhydra Shark,” *Information and Software Technology*, vol. 51, no. 8, pp. 1187– 1216, 2009.

[3] WorkFlow patterns, “Workflow data patterns - Evaluation,” 2006. [Online]. Available: http://www.workflowpatterns.com/evaluations/.

[4] GSMA, “OneAPI.” [Online]. Available: https://gsma.securespsite.com/access/Access API Wiki/Home.aspx. [Accessed: 02-Mar-2012].

[5] S. Shang, E. Li, Y. Wu, and O. Hou, “Understanding Web 2.0 service models: A knowledge-creating perspective,” *Information & Management*, vol. 48, no. 4–5, pp. 178–184, May 2011.

[6] Mobicents, “Shopping - Mobicents.” [Online]. Available: http://www.mobicents.org/shopping-demo.html.

[7] T. Eichelmann, W. Fuhrmann, U. Trick, and B. Ghita, “Enhanced concept of the TeamCom SCE for automated generated services based on JSLEE,” in *Eighth International Network Conference (INC)*, 2010.

[8] M. Femminella, E. Maccherani, and G. Reali, “A software architecture for simplifying the JSLEE service design and creation,” in *Software, Telecommunications and Computer Networks (SoftCOM)*, 2010, vol. 22.

[9] D. Zhu, Y. Zhang, B. Cheng, B. Wu, and J. Chen, “HSCEE : A Highly Flexible Environment for Hybrid Service Creation and Execution in Converged Networks,” *Journal of Convergence Information Technology*, vol. 6, no. 3, pp. 264–276, 2011.