

**SUPERVISORES DE SECUENCIAS DE EVENTOS
DISCRETOS DISEÑADOS CON REDES DE PETRI
BAJO EL SOFTWARE ARENA**



**María Teresa Cárdenas Galvis
Víctor Gabriel Sandoval Ramos**

Universidad del Cauca

**Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Línea de Control
Popayán, Junio de 2012**

**SUPERVISORES DE SECUENCIAS DE EVENTOS
DISCRETOS DISEÑADOS CON REDES DE PETRI
BAJO EL SOFTWARE ARENA**

**María Teresa Cárdenas Galvis
Víctor Gabriel Sandoval Ramos**

Proyecto de grado presentado como requisito para optar al título de Ingeniero en
Automática Industrial

Director:
PhD. Carlos Alberto Gaviria López

Universidad del Cauca

**Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control**

Línea de Control

Popayán, Junio de 2012

CONTENIDO

1. MARCO TEÓRICO.....	1
1.1. SISTEMAS DE MANUFACTURA FLEXIBLE	1
1.2. SISTEMAS DE EVENTOS DISCRETOS	2
1.3. REDES DE PETRI	3
1.3.1. Definición Formal	4
1.3.2. Representación	4
1.3.3. Tipos De RdP.....	7
1.3.4. Propiedades de una RdP	10
1.4. SUPERVISORES BASADOS EN RDP	12
1.4.1. Definición	12
1.5. SOFTWARE PARA LA SIMULACIÓN DE SEDS.....	14
1.5.1. Herramientas utilizadas para el desarrollo de la aplicación.	16
2. MODELADO DE PROCESOS CON SOFTWARE ARENA	18
2.1. ARENA	18
2.1.1. Características del software	19
2.1.2. Entorno de ARENA.....	21
2.1.3. Módulos y bloques	22
2.1.4. Ejemplo de Aplicación en ARENA	25
2.1.5. ¿Por qué modelar previamente en Red de Petri?.....	31
3. IMPLEMENTACIÓN DE RED DE PETRI EN ARENA	33
3.1. ASIMILACIÓN DEL FLUJO DE MARCAS EN RDP AL FLUJO DE ENTIDADES EN ARENA.....	33
3.2. MODELO DE UN LUGAR DE UNA RDP EN ARENA.....	34
3.2.1. Lugar con uno o múltiples arcos de entrada y uno de salida.....	34
3.2.2. Lugar con múltiples arcos de entrada y salida	36
3.2.3. Lugar asociado a un recurso.....	37
3.2.4. Lugares con pesos en sus arcos de entradas o salidas.....	38
3.3. MODELO DE UNA TRANSICIÓN EN ARENA.....	39
3.3.1. Transición simple de una entrada y una salida	40
3.3.2. Transición de múltiples entradas y salidas.....	40
3.3.3. Transición de uso de recurso	41

3.3.4.	Transición temporizada	43
3.4.	ESTRUCTURAS BÁSICAS DE LAS RDP Y SUS EQUIVALENTES EN ARENA	43
4.	METODOLOGÍA PARA EL CÁLCULO E IMPLEMENTACIÓN DE SUPERVISORES BASADOS EN RDP	50
4.1.	METODOLOGÍA PARA EL DISEÑO DE UN SUPERVISOR EN RDP	50
4.1.1.	Invariantes de Lugar	51
4.1.2.	Restricciones que incluyen el vector del disparo	56
4.1.3.	Transiciones No controlables y No observables.....	60
4.1.4.	Transiciones que incluyen tiempo.....	63
4.2.	MÉTODO PARA LA IMPLEMENTACIÓN DE SUPERVISORES BASADOS EN RDP EN ARENA	64
4.2.1.	Lugares de supervisión en ARENA.....	64
4.2.2.	Modificadores de transición en ARENA	67
5.	DISEÑO E IMPLEMENTACIÓN DE LOS CASOS DE ESTUDIO	69
5.1.	CASO DE ESTUDIO 1: ROBOTS Y MESA GIRATORIA.....	69
5.1.1.	Modelo conceptual	70
5.1.2.	Modelo en RdP	71
5.1.3.	Implementación del modelo RdP en ARENA.....	73
5.1.4.	Supervisor basado en RdP para el caso de estudio 2	76
5.1.5.	Implementación del Supervisor en RdP en ARENA para el caso de estudio 1	79
5.2.	CASO DE ESTUDIO 2: TRES ROBOTS Y CUATRO MÁQUINAS	80
5.2.1.	Modelo conceptual	81
5.2.2.	Modelo en RdP	82
5.2.3.	Implementación del modelo RdP en ARENA.....	84
5.2.4.	Supervisor basado en RdP para el caso de estudio 2.....	87
5.2.5.	Implementación del Supervisor en RdP en ARENA para el caso de estudio 2	90
5.3.	CASO DE ESTUDIO 3: AGVS	91
5.3.1.	Modelo conceptual	91
5.3.2.	Modelo en RdP	93
5.3.3.	Implementación del modelo RdP en ARENA.....	94
5.3.4.	Supervisor basado en RdP para el caso de estudio 3.....	97
3.4.1.	Implementación del Supervisor en RdP en ARENA para el caso de estudio 3	99

6. VERIFICACIÓN Y VALIDACIÓN.....	101
6.1. VERIFICACIÓN DEL MÉTODO DE IMPLEMENTACIÓN DE RDP EN ARENA.....	101
6.1.1. Comparación entre flujos de marcas y de entidades para el caso de estudio 1.....	103
6.1.2. Comparación entre flujos de marcas y de entidades para el caso de estudio 2.....	104
6.1.3. Comparación entre flujos de marcas y de entidades para el caso de estudio 3.....	106
6.2. VALIDACIÓN DEL DESEMPEÑO DEL SUPERVISOR EN ARENA.....	107
6.2.1. Validación del desempeño del supervisor en RdP para el caso de estudio 1.	109
6.2.2. Validación del desempeño del supervisor en RdP para el caso de estudio 2.	111
6.2.3. Validación del desempeño del supervisor en RdP para el caso de estudio 3	112
7. CONCLUSIONES, TRABAJOS FUTUROS Y LIMITACIONES.....	114
7.1. CONCLUSIONES	114
7.3. TRABAJOS FUTUROS.....	115
7.4. LIMITACIONES	115
REFERENCIAS BIBLIOGRÁFICAS.....	116

LISTA DE FIGURAS

Figura 1.1 Respresentación gráfica de una RdP	5
Figura 1.2 Lugar Activo.....	5
Figura 1.3 Transición sensibilizada.	5
Figura 1.4 Transición validada [11].....	6
Figura 1.5 RdP del ejemplo.	7
Figura 1.6 RdP Ordinaria.....	8
Figura 1.7 RdP Simple (a) correcta (b) Incorrecta.....	8
Figura 1.8 RdP Libre elección (a) correcta (b) incorrecta.....	8
Figura 1.9 Grafo de Estados (a) correcta (b) incorrecta.	9
Figura 1.10 Grafo Marcado (a) correcta (b) incorrecta.	9
Figura 1.11 RdP T-restrictiva (a) correcta (b) incorrecta.	9
Figura 1.12 RdP Pura (a) correcta (b) incorrecta.	9
Figura 1.13 Evolución de una RdP Generalizada [9].....	10
Figura 1.14 Formas de estudiar un sistema.	14
Figura 1.15 Logo PIPE (Izq), Interfaz gráfica de usuario PIPE2.5 (Der).	17
Figura 1.16 Logo Matlab (Izq), Interfaz gráfica de usuario Matlab (Der).	17
Figura 2.1 Productos De Arena [26].	20
Figura 2.2 Estructura Jerárquica de Arena.....	21
Figura 2.3 Entorno del Software ARENA.....	22
Figura 2.4 Módulo de flujo.	23
Figura 2.5 Módulo de datos.	23
Figura 2.6 Secuencia del proceso de aplicación en ARENA.....	26
Figura 2.7 Configuración bloques <i>Create</i>	26
Figura 2.8 Configuración primeros módulos <i>Assign</i>	26
Figura 2.9 Configuración bloque <i>Process</i> y segundo bloque <i>Assign</i>	27
Figura 2.10 Modelado de la evolución de las tapas.	27
Figura 2.11 Configuración módulos <i>Process</i> , <i>Separate</i> y <i>Decide</i>	28
Figura 2.12 Modelo de la evolución de la parte interior.....	28
Figura 2.13 Configuración para bloques <i>Match</i> , <i>Batch</i> y <i>Process</i>	28
Figura 2.14 Configuración para bloques <i>Record</i> y <i>Assign</i>	29
Figura 2.15 Modelo final de etapa de ensamblado para producto terminado.	29
Figura 2.16 Configuración calendario y reloj.	29
Figura 2.17 Configuración de las variables para los <i>displays</i>	30
Figura 2.18 Configuración para el <i>Run Setup</i> de la ejecución.	30
Figura 2.19 Aplicación simulada ejemplo en ARENA.	31
Figura 3.1 Comparación entre el flujo de marcas y el flujo de entidades.	34
Figura 3.2 Modelo del lugar de una RdP en ARENA.....	35
Figura 3.3 Configuración del bloque <i>Create</i> para el marcado inicial de un lugar.	35
Figura 3.4 Modelo de un lugar con múltiples entradas y múltiples salidas.....	36
Figura 3.5 Configuración de un lugar asociado a un recurso.	37

Figura 3.6 Modelo de lugar con peso en el arco de salida.	38
Figura 3.7 Modelo de un lugar con peso en el arco de entrada.	39
Figura 3.8 Transición de una entrada y una salida.	40
Figura 3.9 Modelo de una transición con múltiples entradas y salidas.	40
Figura 3.10 Configuración del bloque <i>Duplicate</i> para múltiples salidas.	41
Figura 3.11 Modelo de una transición que captura un recurso.	42
Figura 3.12 Modelo de una transición que libera un recurso.	42
Figura 3.13 Modelos de transiciones temporizadas: (Izq) uso de recurso (Der) múltiples entradas y salidas.	43
Figura 4.1 Unidad de dosificación.	53
Figura 4.2 Modelo RdP de la unidad de dosificación.	54
Figura 4.3 RdP con un lugar de supervisión basado en invariante de lugar.	56
Figura 4.4 Transformación de una transición.	57
Figura 4.5 RdP del sistema transformado (vector de disparo).	58
Figura 4.6 RdP del sistema supervisado y contraído.	59
Figura 4.7 Unidad de dosificación supervisada.	63
Figura 4.8 Modelo de lugar de supervisión con una entrada, una salida y marcado inicial.	65
Figura 4.9 Modelo de lugar de supervisión con múltiples entradas y salidas.	66
Figura 4.10 Lugar de múltiples entradas y única salida con peso en el arco de salida.	66
Figura 4.11 Modificador de transición PRE.	67
Figura 4.12 Modificador de transición PRE ampliado.	68
Figura 4.13 Modificador de transición POST.	68
Figura 4.14 Modificador de Transición POST ampliado.	69
Figura 5.1 Esquema del proceso de fabricación.	70
Figura 5.2 RdP del modelo de manufactura caso 1 sin supervisor en Pipe.	71
Figura 5.3 Definición de los lugares P0 y P4.	73
Figura 5.4 Modelo RdP en Arena del Caso de Estudio 1.	76
Figura 5.5 Matriz de Incidencia del caso de estudio 1.	76
Figura 5.6 Red de Petri del modelo caso 1 ampliado.	78
Figura 5.7 Supervisor basado en RdP del modelo de manufactura caso 1.	79
Figura 5.8 Modelo de RdP supervisada en ARENA para caso de estudio 1.	80
Figura 5.9 (a) La distribución de la FMS, y (b) Ruta de producción.	81
Figura 5.10 Red de Petri del modelo del caso 2 sin supervisor en Pipe.	82
Figura 5.11 RdP del modelo del caso 2 disminuido en Pipe.	84
Figura 5.12 Definición de los lugares asociados a recursos en ARENA.	84
Figura 5.13 Modelo de RdP en ARENA del caso de estudio 2.	87
Figura 5.14 Matriz de Incidencia del caso de estudio 2.	88
Figura 5.15 Supervisor del modelo RdP para el caso de estudio 2.	89
Figura 5.16 Modelo de supervisor de RdP del caso de estudio 2.	91
Figura 5.17 SMF con AGV.	92
Figura 5.18 RdP del modelo del caso 2 sin supervisor en Pipe.	93

Figura 5.19 Lugares asociados a recursos para el caso 3.	94
Figura 5.20 Modelo RdP en ARENA del caso de estudio 3.	96
Figura 5.21 Matriz de Incidencia del caso de estudio 3.	97
Figura 5.22 Supervisor del modelo RdP para el caso de estudio 3.	98
Figura 5.23 Modelo de RdP Supervisada para el caso de estudio 3.	100
Figura 6.1 Transición modificada para almacenar orden disparo.	102
Figura 6.2 Configuración del bloque <i>Record</i> para almacenar el orden de disparos.	102
Figura 6.3 Definición del módulo <i>Statistic</i>	103
Figura 6.4 RdP luego del disparo de las transiciones en el orden correspondiente.	103
Figura 6.5 Marcado final obtenido en PIPE para el caso 1.	104
Figura 6.6 RdP luego del disparo del orden dado.	105
Figura 6.7 Marcado final obtenido en Pipe para el caso 2.	106
Figura 6.8 RdP luego del disparo del orden dado para caso 3.	106
Figura 6.9 Marcado final obtenido en Pipe para el caso 3.	107
Figura 6.10 Ciclo de bloques para obtener el marcado de la RdP.	108
Figura 6.11 Configuración del bloque <i>Record</i> para obtener el marcado de un lugar.	108
Figura 6.12 Configuración de los módulos <i>Statistic</i>	109
Figura 6.13 Configuración de los bloques <i>Record</i> para el caso 1.	109
Figura 6.14 Configuración de los módulos <i>Statistic</i> para el caso 1.	110
Figura 6.15 Resultado luego de ejecutar (a) <i> analisis_mesa_no_sup.m</i> (b) <i> analisis_mesa_sup.m</i>	110
Figura 6.16 Configuración de los bloques <i>Record</i> para el caso de estudio 2.	111
Figura 6.17 Configuración de los módulos <i>Statistic</i> para el caso 2.	111
Figura 6.18 Resultado luego de ejecutar (a) <i> analisis_FMS_no_sup.m</i> (b) <i> analisis_FMS_sup.m</i>	112
Figura 6.19 Configuración de los bloques <i>Record</i> para el caso de estudio 3.	113
Figura 6.20 Configuración de los módulos <i>Statistic</i> para el caso 3.	113
Figura 6.21 Resultado luego de ejecutar los archivos (a) <i> analisis_AGV_time_no_sup.m</i> (b) <i> analisis_AGV_time_sup.m</i>	113

LISTA DE TABLAS

Tabla 3.1 Modelo en ARENA de selección.....	43
Tabla 3.2 Modelo en ARENA de atribución.....	44
Tabla 3.3 Modelo en ARENA de distribución.	45
Tabla 3.4 Modelo en ARENA de conjunción o sincronización.....	45
Tabla 3.5 Modelo en ARENA de ejecución secuencial.	46
Tabla 3.6 Modelo en ARENA de concurrencia.	47
Tabla 3.7 Modelo en ARENA de conflictos.....	47
Tabla 3.8 Modelo en ARENA de recurso compartido.....	48
Tabla 4.1 Descripción de lugares y transiciones para el ejemplo tanque dosificación.....	54
Tabla 5.1 Descripción de lugares y transiciones para el Robot1.....	72
Tabla 5.2 Descripción de lugares y transiciones para el Robot2.....	72
Tabla 5.3 Descripción de lugares y transiciones relacionadas con la mesa giratoria.	72
Tabla 5.4 Clasificación de lugares y transiciones para caso 1.	73
Tabla 5.5 Modelo y configuración de las transiciones T0, T2 y T4 caso 1.	74
Tabla 5.6 Modelo y configuración de las transiciones T1, T3, T5, T6 y T7 caso 1.....	74
Tabla 5.7 Configuración de las restricciones de avance para el caso1.....	75
Tabla 5.8 Descripción de lugares y transiciones relacionadas con el supervisor caso 1...	78
Tabla 5.9 Configuración de las salidas del lugar de supervisión caso 1.	79
Tabla 5.10 Configuración de las condiciones de avance caso 1.....	80
Tabla 5.11 Descripción de lugares para la ruta P1.....	82
Tabla 5.12 Descripción de lugares para la ruta P2.....	82
Tabla 5.13 Descripción de lugares para la ruta P3.....	83
Tabla 5.14 Descripción de lugares Complementarios.	83
Tabla 5.15 Lugares con múltiples entradas y múltiples salidas caso 2.	85
Tabla 5.16 Lugares de una entrada y una salida caso 2.....	85
Tabla 5.17 Modelo y configuración de transiciones en RdP en ARENA caso 2.....	86
Tabla 5.18 Descripción de lugares y transiciones relacionadas con el supervisor caso 2.	89
Tabla 5.19 Configuración de los lugares de supervisión caso 2.	90
Tabla 5.20 Configuración de las restricciones de avance para incluir el supervisor caso 2.	90
Tabla 5.21 Descripción de lugares caso 3.....	93
Tabla 5.22 Configuración de lugares de múltiples entradas y múltiples salidas caso 3.	94
Tabla 5.23 Configuración de lugares de una entrada y una salida caso 3.....	95
Tabla 5.24 Modelo y configuración de las transiciones temporizadas del caso 3.	95
Tabla 5.25 Modelo y configuración de las transiciones del caso 3.....	96
Tabla 5.26 Descripción de lugares y transiciones relacionadas con el supervisor caso 3.	99
Tabla 5.27 Configuración de las restricciones de avance para incluir el supervisor caso 3.	99

RESUMEN

Durante las últimas décadas la rápida evolución de la tecnología ha producido una proliferación de nuevos sistemas dinámicos, generalmente hechos por el hombre y de gran complejidad. Ejemplos de ellos son las redes de computadoras, sistemas de producción automatizados, control de tráfico aéreo y sistemas en general de comando, de control, de comunicaciones y de información. Todas las actividades en estos sistemas se deben a la ocurrencia asincrónica de eventos discretos, algunos controlados y otros no. Esta característica es la que lleva a definir el término de *Sistemas de Eventos Discretos SED's*.

La simulación de estos sistemas se ha convertido en una técnica muy importante para las empresas, los investigadores y estudiantes; permitiéndoles conocer con anticipación el comportamiento de un sistema real o hipotético, esto se debe a la popularidad de la simulación y a que los computadores y el software son cada vez mejores.

El software de simulación ARENA de Rockwell Automation es una herramienta que permite construir el modelo de un sistema o proceso a estudiar de manera gráfica, mediante la utilización de una serie de módulos que representan el proceso modelado tipo "diagrama de flujo", describiendo el flujo de las entidades en su paso por el sistema. Una vez realizado este, se introducen los datos de dichos módulos y se ejecuta la simulación. No obstante, el modelado en ARENA no incluye mecanismos formales de modelado que permitan conocer de antemano las buenas propiedades del modelo obtenido.

Para lograr óptimos resultados en una simulación, se debe partir de modelos que expresen a cabalidad el comportamiento del sistema a estudiar, para ello en este proyecto se utiliza como formalismo de representación las redes de petri RdP, ya que son ampliamente conocidas y tienen un poder descriptivo capaz de capturar las características de los SED's. Adicionalmente, para mejorar el desempeño y satisfacer los criterios de seguridad del modelo de un sistema, se plantea en este trabajo el uso de una metodología de diseño de supervisores basados en redes de petri para resolver problemas que no son contemplados como tal por dicho formalismo, usando las herramientas PIPE y Matlab para el cálculo del supervisor.

Finalmente, se implementan los modelos y los supervisores basados en el formalismo de RdP para tres casos de estudio que describen el comportamiento de sistemas de manufactura flexible, en el software de simulación industrial ARENA, lo cual se logra, a través de un método propio creado para la traducción de una RdP y su supervisor; de manera que se aproveche las características de esta herramienta y sea un estándar para modelar RdP en este software, además de permitir verificar el desempeño de los modelos obtenidos y validar dicho método.

1. MARCO TEÓRICO

En el presente capítulo se realiza una contextualización con las definiciones formales, características y conceptos claves para el entendimiento del proyecto realizado, lo que constituye una base inicial sobre los temas directamente relacionados con el presente trabajado de grado.

1.1. SISTEMAS DE MANUFACTURA FLEXIBLE

Un Sistema de Manufactura Flexible (FMS) es difícil de definir, ya que un sin número de personajes ha tratado de describirlo bajo su propia perspectiva. El concepto se introduce como tal a finales de la década de los sesenta. Para 1981 Klahorst, T.H. definió a los Sistemas de Manufactura Flexible como “Grupo de máquinas y equipos relacionados reunidos para procesar una familia de partes, que incluye unos componentes primarios (máquinas herramientas, sistema de manejo de materiales y una red de control computacional) y unos componentes secundarios (tecnología de proceso control numérico por computador – CNC, herramental, dispositivos de sujeción y sistema de administración de operaciones)”, mientras que Ranky, P. (1983), lo define como “Sistemas que incorporan el proceso distribuido de información y el flujo automatizado de materiales a través de maquinaria controlada por computadora, celdas de ensamble, robots industriales, máquinas de inspección, sistemas de manejo y almacenamiento de materiales e integrados por computadora” [1].

En general los FMS son sistemas conformados por máquinas que contienen varias estaciones de trabajo conectadas por un sistema de alimentación de materiales que es capaz de permitir el flujo de trabajos pasando por diversas rutas a través del sistema (transporte), permitiendo que varios tipos de productos puedan ser simultáneamente procesados debido a que se tienen las herramientas y la información de procesamiento necesarias para trabajar en múltiples tipos de productos, brindando a estos sistemas, una gran flexibilidad y adaptabilidad, que es aprovechada para satisfacer las necesidades de los clientes en cuanto a la variedad de productos que pueden ser elaborados, en un tiempo y costo razonables [2]. Típicamente, un FMS es programado para operar de acuerdo a objetivos predefinidos; por ejemplo, la optimización del flujo de materiales o la maximización de uso de las estaciones. El computador central selecciona una pieza específica para ser mecanizada de acuerdo a los programas de producción almacenados en su memoria. La lleva, la fija en la máquina y luego ejecuta el primer programa de mecanizado, y así sucesivamente. Grabados en el mismo computador se encuentran los pasos para todos los procesos de las distintas piezas, de tal manera que sea él quien discierna cuál será la máquina que empezará a mecanizar tal o cual pieza. Así mismo

será él quien tome las decisiones de cuando una máquina deja de operar, ya sea por ubicación o por tiempo. De esta manera, las piezas viajan simultáneamente por el sistema en orden aleatorio, parando sólo en estaciones seleccionadas. Cuando el procesamiento está completo, las piezas terminadas son enviadas a la estación de carga/descarga, donde son removidas por un Robot o AGV o si es el caso, un operario. Esta es la única operación manual requerida en la operación de un FMS (excluyendo obviamente preparación de las herramientas, mantención y monitoreo); todas las otras funciones son automáticas en la mayoría de los casos [3].

Las herramientas de simulación han ido creciendo en importancia al ayudar a planear, diseñar y administrar FMSs. Solo utilizando las capacidades de un computador para modelar distintos escenarios y configuraciones, puede un planificador estar seguro de seleccionar la más efectiva solución para las necesidades particulares del sistema. Existen dos tipos de programas que realizan simulaciones: Los de simulación discreta y los de simulación continua. Los primeros permiten analizar la factibilidad de lograr cierta producción en determinado período de tiempo maximizando el uso de las máquinas, y los simuladores continuos permiten detectar inconvenientes particulares, así como los tiempos exactos de mecanizado para así determinar también los costos exactos de cada operación [3].

1.2. SISTEMAS DE EVENTOS DISCRETOS

El desarrollo de nuevas tecnologías ha generado nuevos sistemas dinámicos de alta complejidad con una dinámica caracterizada por la ocurrencia de eventos discretos por lo general irregulares y desconocidos que alteran su estado. Esta característica ha llevado a la definición del tipo de sistemas dinámicos conocidos como *Sistemas De Eventos Discretos (SED)* [4].

Un SED es un sistema de estados discretos dirigido por eventos, es decir, su evolución de estados depende únicamente de la ocurrencia de eventos discretos asíncronos en el tiempo. Las características típicas de los SED incluyen concurrencia, comportamiento asíncrono, selección no determinista, exclusión mutua, y restricciones de tiempo real [5].

La evolución en el tiempo del sistema se traduce en cambios de estado de algún atributo de una entidad, a través de un evento que ocurre en cierto instante.

La aplicación de los SED en automatización es típica de procesos en los que se presentan tareas secuenciales y concurrentes. Entre ellos se pueden mencionar: procesos de mecanizado, prensado, galvanizado, líneas de transferencia, procesos de inyección, soplado, termo formado, fundición, soldadura, redes de computadoras, redes de comunicaciones, tráfico vehicular, logística, FMS, entre otros etc. Los SED son una clase importante de sistemas dinámicos que evolucionan en el tiempo por la abrupta

ocurrencia de ciertos fenómenos físicos llamados eventos en intervalos de tiempo posiblemente irregulares [6]. Los SED reemplazan la discretización por la cuantificación de las variables de estado, mejorando la respuesta dinámica de los sistemas de control digital y reduciendo el costo computacional y el tráfico de información entre la planta y el controlador [7].

Un formalismo proporciona un conjunto de convenciones para especificar una clase de objetos con precisión y sin ambigüedades. En la literatura podemos encontrar numerosos formalismos (Autómatas, Grafos de Eventos, Cadenas de Markov, DEVS¹, Statecharts, ACD², Redes de Petri, Grafcet, RTTL³/TTM⁴, CCS⁵,...), así como, muchas generalizaciones y particularizaciones de los mismos [5].

Para el desarrollo de esta tesis se ha elegido como formalismo de trabajo las Redes de Petri, por ser una de las herramientas de mayor desarrollo y aplicación y por la amplia disponibilidad de recursos y producción académica con que se cuenta en la actualidad.

1.3. REDES DE PETRI

Las Redes de Petri (RdP) tienen más de 50 años de existencia desde su formulación en 1962 por *Carl Adam Petri*, realizándose durante este periodo miles de publicaciones sobre o relacionadas con: teorías de RdP, desarrollo de nuevos tipos de RdP, aplicaciones exitosas de RdP, nuevos ámbitos de aplicación, nuevas formas de implementación, etc [8].

Las RdP son una herramienta gráfica y matemática para el estudio de un gran número de sistemas, siendo un formalismo ampliamente aceptado de modelización de SED's con el cual es posible controlar, evaluar y optimizar diferentes procesos. Poseen un gran poder descriptivo y han sido diseñadas específicamente para modelar sistemas con componentes interactuantes y por esto son capaces de capturar muchas de las características de los sistemas manejados por eventos (operaciones asíncronas, bloqueos, conflictos, etc.) [1]. Las RdP, fueron desarrolladas para simular propiedades dinámicas de los sistemas complejos, permitiendo modelar fácilmente sistemas con eventos concurrentes, asíncronos, distribuidos, paralelos y/o estocásticos como se explica en [4]. Con su ayuda podemos modelar el comportamiento y la estructura de un sistema, y

¹ DEVS: Es un formalismo de especificación general para SED (Discrete Event System specification)

² ACD: (Activity Cycle Diagram) La representación alterna estados pasivos y actividades

³ RTTL: (*Real-Time Temporal Logic*) es una lógica de primer orden que incluye cuantificación explícita sobre la variable tiempo.

⁴ TTM: (Timed Transition Models) Modelos utilizados junto con la lógica anterior.

⁵ CCS: (*Calculus for Communicating Systems*) lenguajes algebraicos se basan en la descripción de las operaciones que se efectúan sobre los procesos.

llevar el modelo a condiciones límite, que en un sistema real son difíciles de lograr o muy costosas.

La teoría de RdP ha llegado a ser reconocida como una metodología establecida en la literatura de la robótica para modelar los sistemas de manufactura flexibles. Una de las grandes ventajas en la aplicación de las RdP a los sistemas reales, está representada en la capacidad de monitorear su evolución [1].

1.3.1. Definición Formal

Una RdP se define como una 6-tupla $RdP = (P, T, Pre, Post, M, W)$, donde:

- $P = \{p_1, p_2, \dots, p_n\}$ es el conjunto finito de lugares, para $n > 0$.
- $T = \{t_1, t_2, \dots, t_m\}$ es el conjunto finito de transiciones, con $P \cup T \neq \emptyset$ y $P \cap T = \emptyset$ para $m > 0$.
- Pre: es la función de pre-incidencia que especifica los arcos dirigidos de los lugares a las transiciones.
- Post: es la función de post-incidencia que especifica los arcos dirigidos de las transiciones a los lugares.
- $M(p)$: es el marcaje de la red. Un marcaje inicial de la RdP es denotado por M_0 .
- W : es el conjunto de pesos asociados a los arcos [1].

Una Red de Petri con la estructura $(P, T, Pre, Post, W)$ sin especificar su marcado inicial es denotada por N . Por otro lado, una RdP con un marcado inicial dado es denotado por $RdP=(N, M_0)$.

Un marcado M de una RdP es una función desde el conjunto de los lugares P al conjunto de los enteros no negativos \mathbb{N} :

$$M: P \rightarrow \mathbb{N}$$

Si n es el número de lugares de la RdP, un marcado puede interpretarse como un vector de dimensión n , $M=(m_1, m_2, \dots, m_n)$, en el que m_i es el número de marcas que M asigna a p_i y se verifica $M(p_i)=m_i$ [9].

1.3.2. Representación

Un modelo es una representación (en términos matemáticos) de las características más importantes del objeto o sistema de estudio. Manipulando esta representación, se pueden obtener nuevos conocimientos del sistema modelado sin ningún coste o peligro para el sistema real. Sin embargo, el modelado por sí solo sirve de poco, es necesario, analizar el sistema modelado. El sistema se modela primero como una red de Petri y después, este modelo se analiza. Este análisis nos lleva a una mejor comprensión del comportamiento

del sistema modelado [10]. Existen dos tipos de representaciones para la RdP descritas a continuación.

Representación Gráfica

Las RdP son un tipo particular de grafo dirigido, pesado y bipartito, compuesto por lugares y transiciones que se encuentran unidos por arcos, donde los lugares representan los estados del modelo, los arcos indican la dirección de evolución de la RdP y las transiciones representan las entradas del sistema; de su activación o desactivación dependerá el funcionamiento de la red [1]. En la representación gráfica los lugares son representados con círculos, las transiciones con barras y los arcos por flechas. Algunos arcos van desde un lugar a una transición y otros desde una transición a un lugar. Las marcas se representan como puntos negros en los lugares [1]. Se Destaca de esta representación gráfica que es fácil, clara y sin ambigüedades.

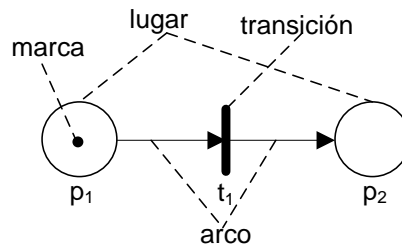


Figura 1.1 Representación gráfica de una RdP.

Los lugares que contienen marcas se consideran lugares activos.



Figura 1.2 Lugar Activo.

A las transiciones se les asocia eventos (funciones lógicas de las variables de entrada). Una transición se dice que está sensibilizada cuando todos sus lugares origen están marcados.

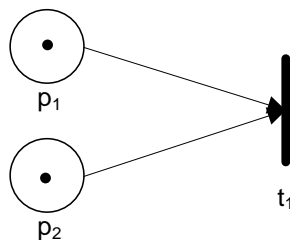


Figura 1.3 Transición sensibilizada.

Cuando ocurre un evento asociado a una transición, se dice que la transición está validada.

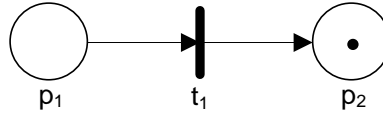


Figura 1.4 Transición validada [11].

Existen dos términos muy importantes que se manejan comúnmente: la habilitación de una transición y el disparo de una transición. La habilitación permite modelar si un evento puede o no ocurrir y con el disparo se origina al menos un cambio de estado en el sistema [12].

Representación Matemática

Una transición tiene un determinado número de lugares de entrada (o precondiciones) y de lugares de salida (o postcondiciones). Cada uno de estos se puede representar por una matriz binaria de dos dimensiones, donde las columnas representan las transiciones, las filas los lugares y las celdas la conexión entre ambas. Las matrices reciben los nombres de “Matriz de incidencia previa” (pre-incidencia) y “Matriz de incidencia posterior” (post-incidencia) respectivamente.

Podemos decir que una RdP, N se encuentra definida matricialmente por medio de dos matrices. Sea $n = |P|$ (número de lugares de P) y $m = |T|$ (número de transiciones de T). Se denominan:

Matriz de incidencia previa:

$$C^- = [c^-_{ij}]^{n \times m} \quad \text{donde } c^-_{ij} = Pre(p_i, t_j) \quad (1.1)$$

Matriz de incidencia posterior:

$$C^+ = [c^+_{ij}]^{n \times m} \quad \text{donde } c^+_{ij} = Post(p_i, t_j) \quad (1.2)$$

Donde se entiende $Pre(p_i, t_j)$ como el número de marcas que se retiran del lugar p_i al dispararse la transición t_j , y $Post(p_i, t_j)$ como el número de marcas que se depositan en el lugar p_i al dispararse la transición t_j .

Matriz de incidencia de N :

$$C = C^+ - C^- \quad (1.3)$$

La matriz de incidencia define para cada lugar, el balance de marcas resultante del disparo de una transición.

Ejemplo del cálculo de la Matriz de Incidencia

Dada la RdP $R_1 = (N, M_0)$, donde:

$$P = \{p_1, p_2, p_3, p_4\},$$

$$T = \{t_1, t_2, t_3\},$$

$$M_0 = (5, 0, 0, 0)$$

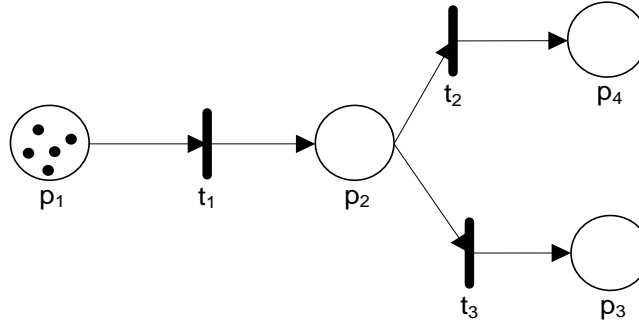


Figura 1.5 RdP del ejemplo.

Las matrices de Incidencia previa y de incidencia posterior son:

$$C^- = Pre(p_i, t_i) = \begin{matrix} & \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{matrix} \quad (1.4)$$

$$C^+ = Post(p_i, t_i) = \begin{matrix} & \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{matrix} \quad (1.5)$$

Finalmente la combinación de las anteriores matrices proporciona la Matriz de Incidencia del modelo [11].

$$C = C^+ - C^- = \begin{matrix} & \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{matrix} \quad (1.6)$$

1.3.3. Tipos De RdP

- **Red de Petri ordinaria** si sus funciones de incidencia sólo pueden tomar valores 0 y 1 (todos sus arcos son de peso unitario).

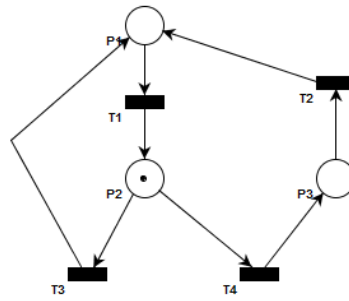


Figura 1.6 RdP Ordinaria.

- **Red de Petri Simple:** Para cada transición, a lo sumo uno de sus lugares de entrada puede ser compartido con otras transiciones.

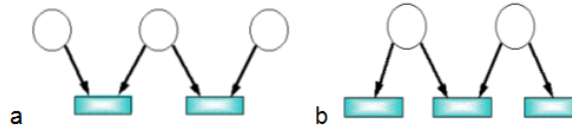


Figura 1.7 RdP Simple (a) correcta (b) Incorrecta.

- **Red de Petri Libre Elección:** Una red de Petri es de libre elección si todos los lugares tienen como conjunto de salida más de una transición, pero este conjunto de transiciones solo debe tener un conjunto de entrada unitario.

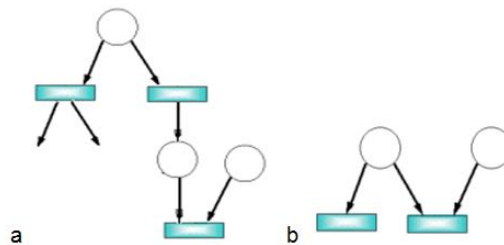


Figura 1.8 RdP Libre elección (a) correcta (b) incorrecta.

- **Grafo de Estados:** Es una red de Petri ordinaria tal que cada transición t tiene exactamente un lugar de entrada y un lugar de salida, es decir:

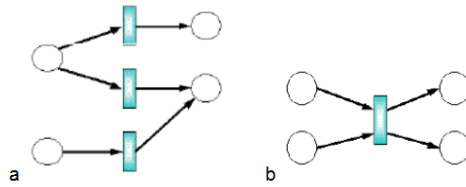


Figura 1.9 Grafo de Estados (a) correcta (b) incorrecta.

- **Grafo Marcado:** Es una red de Petri ordinaria tal que cada lugar p tiene exactamente una transición de entrada y una transición de salida.

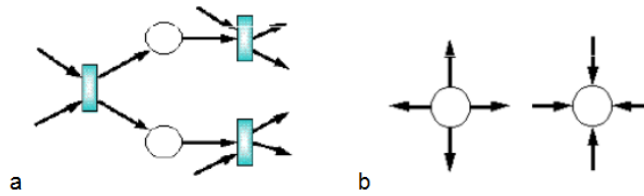


Figura 1.10 Grafo Marcado (a) correcta (b) incorrecta.

- **Red de Petri T-Restructivas:** Son aquellas para las cuales todas las transiciones deben tener al menos un lugar de entrada y un lugar de salida.

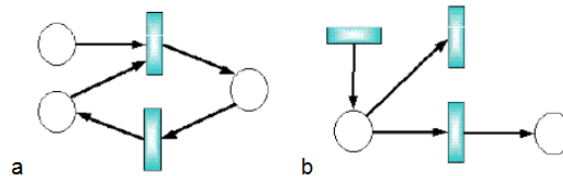


Figura 1.11 RdP T-restrictiva (a) correcta (b) incorrecta.

- **Red de Petri Pura o No Reflexiva:** si ningún lugar es a la vez entrada y salida de una misma transición.

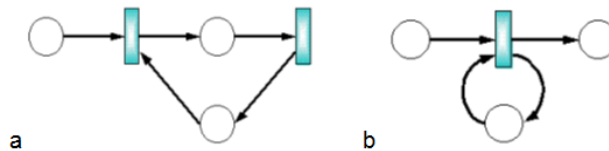


Figura 1.12 RdP Pura (a) correcta (b) incorrecta.

- **Red de Petri Generalizada:** sus funciones de incidencia pueden tomar cualquier valor entero mayor o igual que cero.

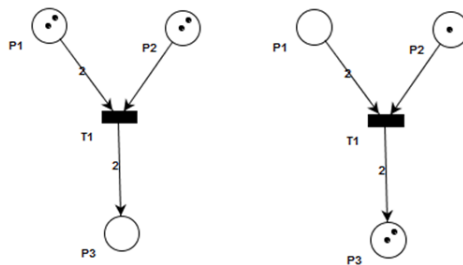


Figura 1.13 Evolución de una RdP Generalizada [9].

- Red de Petri con Tiempo:** Las RdP no incluyen concepto alguno de tiempo, por ello, solamente es posible describir la estructura lógica de los sistemas y no su evolución temporal. La introducción del tiempo en el modelo permite la descripción de comportamientos dinámicos de los sistemas, considerando la evolución de estados y la duración de cada acción tomada por el sistema. Hay múltiples formas diferentes de introducir el concepto de tiempo.

Una primera posibilidad consiste en asociar a cada transición un número que indica, en alguna unidad temporal adecuada, el retardo entre la habilitación y el disparo de la transición. Una RdP Temporizada puede ser definida como:

$TPN = \{P, T, Pre, Post, M_0, Q\}$, donde $P, T, Pre, Post$ y M_0 se definen como antes y $Q = (q_1, q_2, \dots, q_m)$ es el conjunto de retardos asociados a las transiciones.

Una segunda posibilidad para la introducción del concepto de tiempo, consiste en asignar un retardo q al proceso de convertir una ficha en disponible luego que la misma llega a un nuevo lugar. Por ello, cada ficha puede estar en uno de dos estados: disponible y no disponible; solamente fichas disponibles habilitan transiciones. La falta de disponibilidad de una ficha modela el tiempo utilizado desarrollando una actividad. En esta abstracción, el tiempo es asociado a los lugares [11].

1.3.4. Propiedades de una RdP

Diversos investigadores han desarrollado metodologías para construir redes de Petri a partir de las especificaciones de un sistema de producción, destacándose el trabajo de Zhou y Dicesare [13], donde a partir de las especificaciones de un proceso de manufactura diseñan una RdP. La fortaleza del modelado de las RdP radica en sus propiedades, que se dividen en dos grandes áreas, las dependientes del marcado inicial llamadas propiedades *dinámicas* o *del comportamiento* y las propiedades independientes del marcado, llamadas *estructurales* o *estáticas* [8].

Propiedades Dinámicas

- Alcanzabilidad: es la principal propiedad dinámica y consiste en que cada disparo de una transición habilitada modifica la distribución de los marcados dentro de la red, de acuerdo con las reglas de disparo. Una secuencia de disparos generara una secuencia de marcados. Se dice que un marcado M_n es alcanzable desde el macado M_0 si y solo si existe una secuencia de disparos que transforme M_0 en M_n . La secuencia de disparos se denota por s :
$$s = M_0 t_1 M_1 t_2 \dots t_n M_n$$
- Limitable o Acotada: si el numero de marcas de la red en cada lugar no excede un numero finito k para cualquier marcado alcanzable desde M_0 y existirá dentro de todos los posibles marcados de la red, $M(p) \leq k$.
- Vivacidad: si no importa cual marcado haya sido alcanzado, siempre será posible una nueva secuencia s de disparos y alcanzar un nuevo marcado. Libre de bloqueos.
- Reversibilidad: si desde cualquier marcado que se pueda alcanzar se puede volver al marcado inicial. Si un sistema tiene bloqueos, no es reversible.
- Cobertura: Un marcado M dentro de una RdP (N, M_0) en un conjunto de marcados cubiertos o contenido, si existe un marcado M' dentro de $R(N, M_0)$ tal que $M'(p) \geq M(p)$ para cada lugar p dentro de la Red.
- Persistencia: Si para cualquiera de dos transiciones habilitadas, el disparo de una transición no deshabilitara a la otra transición. Esta propiedad es importante en situaciones de paralelismo y sincronización.

Los métodos de análisis de las propiedades dinámicas son tres:

1. El árbol de cobertura.
2. Matriz de incidencia y ecuación de estado.
3. Reglas de reducción [8].

Propiedades Estructurales

- Vivacidad Estructural: Una RdP es estructuralmente viva si tiene un marcado inicial para N .
- Controlabilidad: Una RdP se dice que es completamente controlable si cualquier marcado es alcanzable desde cualquier otro marcado. Para una red

completamente controlable se cumple que $\text{Rango}(A) = m$, donde m es la cantidad de lugares de la red.

- Limitación o acotado estructural: Una RdP es limitada estructuralmente si es limitada para cualquier conjunto finito de marcados iniciales M_0 .
- Conservabilidad: Una RdP es conservativa si existe un entero positivo $y(p)$, para cada lugar p tal que la sumatoria de marcas sea constante para cada $M \in R(N, M_0)$.
- Repetibilidad: Una RdP es repetible si existe un marcado M_0 y una secuencia de disparos s desde M_0 , tal que las transiciones se disparan infinitamente en la secuencia definida por s .
- Consistencia: Una RdP es consistente si existe un marcado M_0 y una secuencia de disparos reversible s desde M_0 hacia M_0 , tal que cada transición haya sido disparada al menos una vez en s [8].

1.4. SUPERVISORES BASADOS EN RDP

1.4.1. Definición

Desde SED, se plantea que el concepto de *supervisor* está asociado al concepto de *controlador*, haciendo una analogía con la teoría de control realimentado en Sistemas de variables continuas. En este sentido, la función de supervisión está asociada a generar acciones de control para habilitar y deshabilitar eventos controlables, con el objetivo de tener un desempeño del sistema en lazo cerrado, tal que la trayectoria de eventos esté siempre en un conjunto de cadenas de eventos deseados. Generando una política de control que garantiza que se evite el estado prohibido y se permite un máximo conjunto de transiciones de estado [14].

La teoría del control supervisor de Ramadge y Wonham [15], es hasta ahora la teoría más comprensible para el control de SED. Esta ha sido ampliamente utilizada para diseñar, analizar y sintetizar sistemas lógicos de control. Esta teoría propone representar el comportamiento del sistema a controlar (planta), con un lenguaje formal que debe ser generado por un SED. Dicho lenguaje debe ser restringido a un comportamiento requerido (especificación o restricción), mediante la habilitación/deshabilitación de la posible ocurrencia de eventos controlables en la planta, a través de un agente externo denominado *supervisor*.

Es importante diferenciar algunos conceptos para no generar confusiones con el tema.

1.- *Sistema de supervisión*: Sistema con la habilidad de medir un proceso y actuar en él, formado por componentes interactivos que razonan acerca del comportamiento del proceso, para proponer y ejecutar acciones apropiadas para mantener las condiciones de operación normal a un caso de fallas.

2.- *Control supervisorio*: es una teoría general para la síntesis de controladores, llamados *supervisores*, en sistemas de eventos discretos. También se aplica a los sistemas continuos cuando el supervisor se diseña desde los sistemas de eventos discretos. El supervisor de un SED cambia la entrada de control de acuerdo al estado actual del sistema, observa su estado y por cada estado admisible, una entrada de control debe ser aplicada en ese punto de funcionamiento. Un evento es admisible para un SED supervisado, si y solo si es físicamente posible y autorizado por la función de supervisión.

3.- *Control supervisado*: proceso sometido a un subsistema de control que interactúa directamente con las señales físicas restringiendo su comportamiento y una entidad de orden jerárquico superior que coordina las tareas del controlador. El control supervisado presupone una estructura jerárquica en la que en el bajo nivel se tiene un sistema controlado (discreto o continuo) y en el alto nivel un supervisor que actúa sobre el sistema controlado, presentando como ventaja que el supervisor sólo maneja la coordinación y no considera control directo de los subsistemas.

4. - *SCADA, Supervisory control and data acquisition*: según ISA (International Society of Automation) es la tecnología que habilita a un usuario para recoger datos desde una o más instalaciones distantes y le permite enviar instrucciones de control limitadas a esas instalaciones. Los SCADA requieren que un operario permanezca al frente del sistema o que visite frecuentemente la ubicación remota. La definición de la IEEE (IEEE 100, 2000) establece que los SCADA son sistemas que operan con señales codificadas sobre canales de comunicación a fin de proporcionar control remoto de equipos, obteniendo información sobre el estado de éstos para la visualización o registro de sus funciones. Puede concluirse entonces que los SCADA son sistemas de adquisición de datos y que junto con un operador humano u otro sistema de toma de decisiones, conforman un sistema de supervisión en el sentido de la definición dada anteriormente.

5.- *Monitoreo: Recolección de datos desde el proceso*. Determina el estado actual del sistema controlado y hace inferencias necesarias para producir datos adicionales como históricos o diagnóstico. El monitoreo se limita a la captura de datos del proceso y no tiene acciones directas en los modelos o en la evolución del estado [14].

Aclarando estos conceptos, se puede concluir que un supervisor basado en RdP o control supervisorio; es aquel capaz de deshabilitar las transiciones controlables del modelo en respuesta a los diferentes efectos de los eventos generados, garantizando que ninguno de los estados prohibidos sea alcanzado y lo debe hacer con la mínima intervención en el

comportamiento del sistema posible. En el capítulo 4 se describe con detalle la metodología adoptada para diseñar supervisores basados en RdP y el método de implementación en el software ARENA.

1.5. SOFTWARE PARA LA SIMULACIÓN DE SEDS

La simulación, en el contexto de la informática, es la imitación de un sistema real o hipotético a través de la utilización de un modelo lógico construido en una computadora, según ciertas condiciones particulares de operación.

La simulación es una colección de métodos muy potente que permite ayudar a diseñar y evaluar sistemas del mundo real, tanto existentes como teóricos. Esto nos permite imitar procesos industriales, servicios, procesos comerciales, procesos académicos y de diversa índole. Algunos ejemplos incluyen fábricas, con todos los procesos, materiales y actores intervinientes; servicios como bancos u hospitales; y procesos comerciales como un almacén de distribución o un restaurante de comida rápida.

La realización de una simulación supone la disponibilidad de un modelo que puede ser el producto del análisis de un sistema existente o el diseño de uno a construir. En ambos casos el objetivo de la simulación es el de conocer cómo se comporta el mismo, ya sea para compararlo con otro sistema o con otra configuración del mismo sistema.

Según Kelton [16], en un paso previo a la simulación, la construcción del modelo puede ser en sí misma un proceso enriquecedor, ya que nos vemos obligados a entender el sistema a estudiar y a formalizar y documentar su funcionamiento. A partir de este proceso se suelen generar alternativas para su configuración y optimización [17].

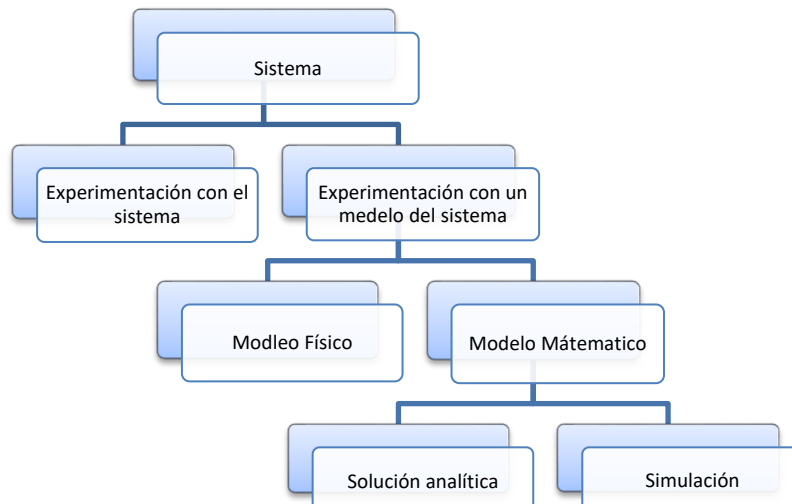


Figura 1.14 Formas de estudiar un sistema.

En el caso particular del enfoque orientado a eventos discretos, la simulación se utiliza para representar acciones instantáneas en tiempos no periódicos, que pueden cambiar el

estado actual del sistema en estudio [18]. Según Thesen y Travis (1991), cuando elegimos modelar un sistema del mundo real usando simulación con eventos discretos renunciamos a la posibilidad de capturar un grado de detalle solo describable como un cambio suave y continuo. Pero así conseguimos una simplicidad que nos permite capturar las características importantes de muchos sistemas que son demasiado complejos para ser modelados usando modelos continuos [17].

Para explicar cómo funciona una simulación de eventos discretos, conviene describir algunos términos, ya que su conocimiento resulta útil a la hora de comprender y analizar modelos de sistemas.

Entidad: son objetos dinámicos en la simulación; es el objeto sobre lo que actúa el proceso. Usualmente se crean, atraviesan el sistema y luego son desechadas. Las entidades representan personas, objetos o cosas, bien sean reales o imaginarias, cuyo movimiento en el sistema provoca cambios de estado, afectan y son afectados por otras entidades, y el estado del sistema y afectan las medidas de desempeño de los resultados [18].

Atributo: propiedad de una entidad. En un sistema pueden existir muchos tipos de entidades y cada una tendrá unas características propias llamadas Atributos. Todas las entidades tienen el mismo conjunto de atributos, pero con distintos valores [18].

Variables: variable o variable global es información que refleja alguna característica de un sistema, sin importar cuantos o que tipos de entidades haya alrededor [19]. Las variables son al sistema lo que los atributos a una entidad. Pueden ser accedidas y modificadas por cualquier entidad. El reloj del sistema, encargado de llevar el tiempo simulado es una variable global [17].

Actividad: representa un periodo de tiempo de duración específica [18].

Recurso: Estos son elementos necesarios para realizar una tarea [2]. Son quienes dan servicio a las entidades para procesos u otros trabajos; representan cosas, como personal, equipo o espacio de área de almacenaje de tamaño limitado. Una entidad se aprovecha de un recurso cuando está disponible y lo libera cuando termina [19]. Se llama capacidad de un recurso al número de unidades de recurso idénticas disponibles para dar un servicio [18].

Colas: Espacio donde permanece una entidad mientras espera a que un recurso esté disponible o mientras espera a formar un grupo con otras entidades [18].

Acumuladores estadísticos: Para que la salida de una simulación contenga información útil para quien realiza la simulación se necesita acumular datos durante el proceso. Para esto se crean los acumuladores que llevan la contabilización de distintos parámetros del modelo que estarán disponibles al final de la prueba [17].

Eventos: Un evento es una acción o la ocurrencia de un cambio instantáneo en alguna parte del sistema (simulado) que genera una serie de cambios en los estados del sistema, puede cambiar atributos, variables, o acumuladores estadísticos [17].

Reloj de simulación: Es la variable más importante. Contiene el tiempo actual simulado. Considera el valor del último evento ejecutado en lugar de correr tomando valores continuos como un reloj normal [17].

Las herramientas de simulación orientadas a eventos discretos ofrecen una plataforma que permite abordar con éxito un proceso de mejora continua de sistemas complejos para los cuales las técnicas analíticas clásicas basadas en el uso de cálculo diferencial, teoría de probabilidades y métodos algebraicos, no pueden ser utilizadas para analizar de modo sencillo la complejidad de los procesos [6].

1.5.1. Herramientas utilizadas para el desarrollo de la aplicación.

En la elección de las herramientas software utilizadas en este trabajo, se tuvo en cuenta la existencia de información, tutoriales y soporte técnico disponibles en la internet. Existe gran cantidad de aplicaciones de RdP con herramientas de simulación para el modelado y la verificación, que pueden ser encontradas en la base de datos Petri Nets Tool [20]. La mayoría de este software ha sido desarrollado para investigaciones y propósitos académicos, razón por la cual muchos de ellos presentan dificultades entre ellas: problemas de instalación, problemas de funcionalidad, limitaciones con ciertas clases de RdP entre otros.

Teniendo en cuenta las necesidades y requerimientos de este trabajo, el software elegido para la simulación de los modelos en RdP y su control supervisorio es:

Pipe (Plataform Independent Petri Net Editor 2.5): es un recurso abierto, sirve para crear y analizar redes de Petri, incluyendo redes de Petri estocásticas generalizadas (GSPN). Esta implementada completamente en Java, posee una interfaz grafica de fácil uso, contiene varios módulos de análisis, incluyendo análisis avanzado de GSPN, maneja cientos de miles de estados y elimina estados que dejan de existir sobre la marcha. Pipe también ofrece un paquete completo de módulos de análisis para comprobar las propiedades de comportamiento, produce las estadísticas de rendimiento y algunas características menos comunes tales como la comparación y la clasificación de RdP [21]. Además, genera grafos reutilizables, permite ver la red simulada de forma animada con la característica de que se puede seleccionar el disparo de transiciones a conveniencia. Finalmente, es posible obtener la matriz de incidencia que puede ser exportada a Matlab para un posterior análisis, a través de la extensión HTML.

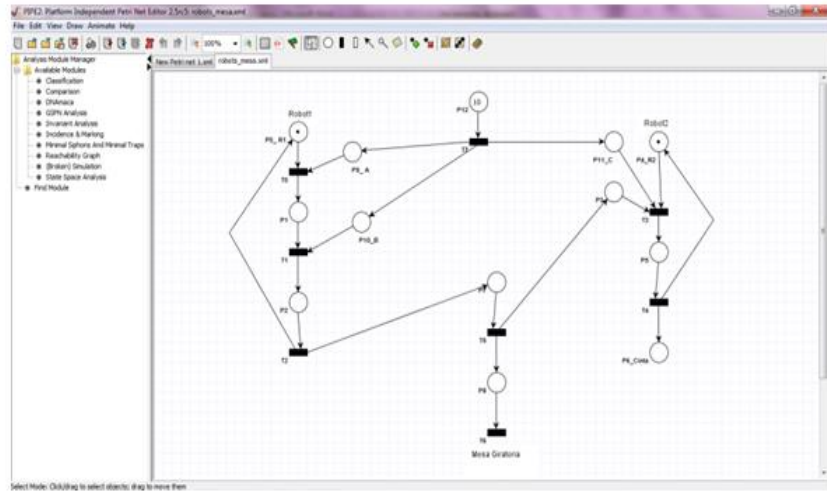


Figura 1.15 Logo PIPE (Izq), Interfaz gráfica de usuario PIPE2.5 (Der).

Matlab: es el nombre abreviado de “MATrix LABoratory”. Matlab es un programa de cálculo técnico y científico, diseñado para realizar cálculos numéricos con vectores y matrices. Es ampliamente usado por ingenieros de control para el análisis y diseño, posee una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, ingeniería entre otras. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa [22].

Por todas sus características Matlab esta herramienta es de vital importancia en este trabajo, ya que permite realizar los cálculos necesarios con las matrices que describen el comportamiento dinámico de las RdP, así como también facilitan el cálculo para el diseño de los supervisores basados en RdP.

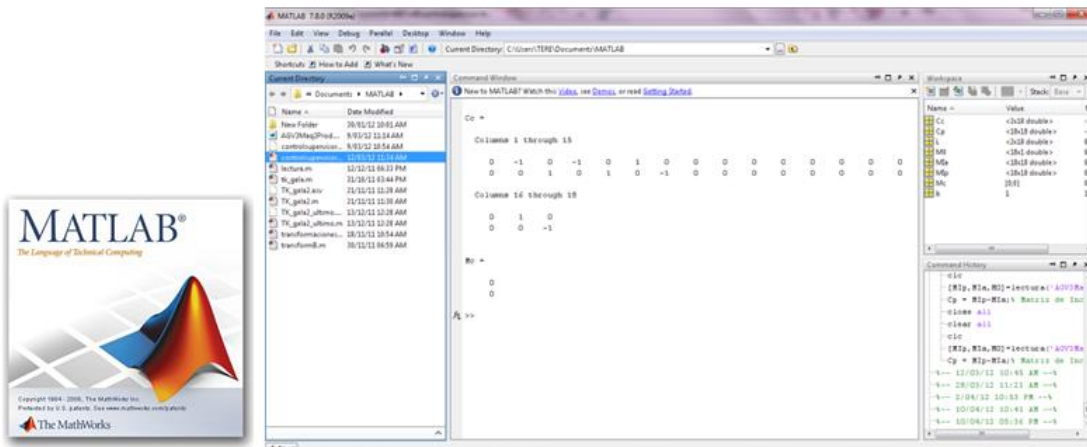


Figura 1.16 Logo Matlab (Izq), Interfaz gráfica de usuario Matlab (Der).

2. MODELADO DE PROCESOS CON SOFTWARE ARENA

En el presente capítulo se introduce a la herramienta de simulación ARENA. Esta se ubica en lo que anteriormente definimos como software de simulación para SEDs. El propósito de esta sección es describir sus características, componentes y aplicaciones; resaltando la importancia de usar este software para conocer los diferentes comportamientos que puede sufrir un determinado sistema. Finalmente a través de un ejemplo se mostrará por qué es importante modelar previamente sobre un formalismo como lo es RdP para evitar problemas en la lógica y estructura del sistema a simular sobre ARENA.

2.1. ARENA

El software de simulación ARENA de Rockwell Automation es una herramienta que permite construir el modelo de un sistema o proceso a estudiar de manera gráfica, mediante la utilización de una serie de módulos que representan el proceso modelado tipo “diagrama de flujo”, describiendo el flujo de las entidades en su paso por el sistema. Una vez realizado este, se introducen los datos de dichos módulos y se ejecuta la simulación [18]. Adicionalmente ARENA ofrece la posibilidad de crear representaciones animadas utilizando una librería gráfica ampliable.

ARENA es un ambiente completo de desarrollo de simulaciones. Provee la capacidad de generar modelos, generar datos aleatorios como entradas, correr los modelos bajo una serie de condiciones parametrizables, generar animaciones, generar reportes estándares y personalizados, comparar distintos modelos, buscar optimización de modelos e integrar todo el proceso con las aplicaciones más usadas del mercado como Microsoft Office, bases de datos y lenguajes de programación [17].

El software ARENA ha sido diseñado para su empleo en todas las funciones de la empresa, ofrece la facilidad de uso, flexibilidad y capacidad de modelado que se requiere para representar cualquier proceso. Desde los procesos de aprovisionamiento, pasando por el almacenaje, fabricación, logística y distribución, hasta la gestión administrativa y el servicio de atención al cliente, así como el análisis de procesos en los que intervienen varias áreas funcionales [23].

La simulación en general y ARENA en particular es cada vez una herramienta más y más demandada por parte de clientes para validar proyectos de muy amplia índole antes de ser implantados de manera efectiva [23].

Los aportes de ARENA a nivel empresarial son:

- Disminución de riesgos empresariales.
- Soporte en toma de decisiones estratégicas.

- Soporte al diseño de procesos.
- Gestión y optimización de recursos empresariales.
- No interferencia de la experimentación.

Es importante dejar claro que ARENA no es un software para realizar supervisorios de procesos, ya que no permite interactuar al usuario directamente con el proceso a través de botones o instrucciones durante la simulación; la simulación de secuencias de operaciones obedecen al diseño que el programador haya realizado y su funcionamiento estará determinado por el comportamiento aleatorio que posee esta herramienta.

Las razones por las cuales se ha elegido este software para este trabajo son porque ARENA permite simular sistemas donde se requiera:

- Conocer la cantidad de productos fabricados en un lapso de tiempo.
- Observar la utilización de un recurso (máquina, operario etc.).
- Analizar el flujo de fabricación.
- Conocer el inventario actual.
- Conocer la tasa de llegada y salida de productos.
- Analizar los resultados a través de de la integración con otras herramientas como Word, Excel, HTML ...
- Observar la animación del sistema simulado.

Como se observa en esta descripción, Arena no es una herramienta para la simulación de redes de petri, entonces, ¿por qué simular una RdP en este software? La respuesta es porque Arena es de tipo industrial, con una madurez de desarrollo tecnológico que fácilmente le ha permitido ser usado en diferentes empresas exitosas a nivel nacional e internacional, lo cual, hace interesante abordar su potencial para traducir un formalismo ampliamente utilizado como lo son las RdP y dejar una base para posteriores desarrollos bajo esta herramienta.

2.1.1. Características del software

Arena comprende una familia de productos que permiten dar solución a diferentes partes de la empresa como se ha explicado anteriormente. Todos los productos de Arena comparten un software de base común, que dependiendo de las necesidades de la compañía que lo requiera se le puede agregar diferentes funciones. Para el caso particular de este trabajo se emplea *Arena Basic Edition* en su versión estudiantil; la cual, es la versión de introducción al mundo ARENA y permite la documentación, animación, y demostrar la variabilidad y la dinámica de un proceso [23].

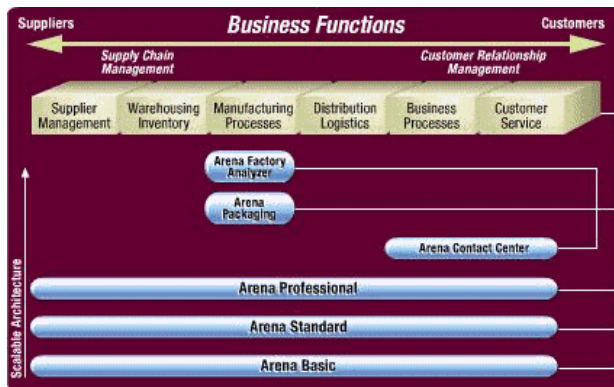


Figura 2.1 Productos De Arena [26].

ARENA combina la facilidad de uso de los simuladores de alto nivel con la flexibilidad de los lenguajes de simulación, y recorre todo el camino hacia abajo de los lenguajes de procesamiento de propósito general como el sistema de programación Microsoft® Visual Basic® o C. Lo hace así al proporcionar plantillas alternativas e intercambiables de modelación de simulación gráfica y módulos de análisis que se pueden combinar para desarrollar una amplia variedad de modelos de simulación [24]. Además ARENA se nos presenta como una herramienta *orientada al proceso* y *orientada a eventos discretos* por cuanto se modela de manera gráfica el flujo de las entidades a través de los módulos, siendo una forma más natural de descripción del proceso y es orientada a eventos discretos pues aunque durante el modelamiento no se observe la definición de eventos, estados, cálculos de variables, avance del reloj de simulación etc, estos están presentes en el software de manera encapsulada.

ARENA mantiene su flexibilidad de modelación al ser totalmente jerárquico, en cualquier momento puede utilizar módulos de bajo nivel del panel de *Bloques y elementos* y tener acceso a la flexibilidad del lenguaje de simulación SIMAN con módulos de más alto nivel de otras plantillas. SIMAN es un lenguaje de simulación de bajo nivel que proporciona los elementos básicos para SED descritos en el ítem 1.5 [26].

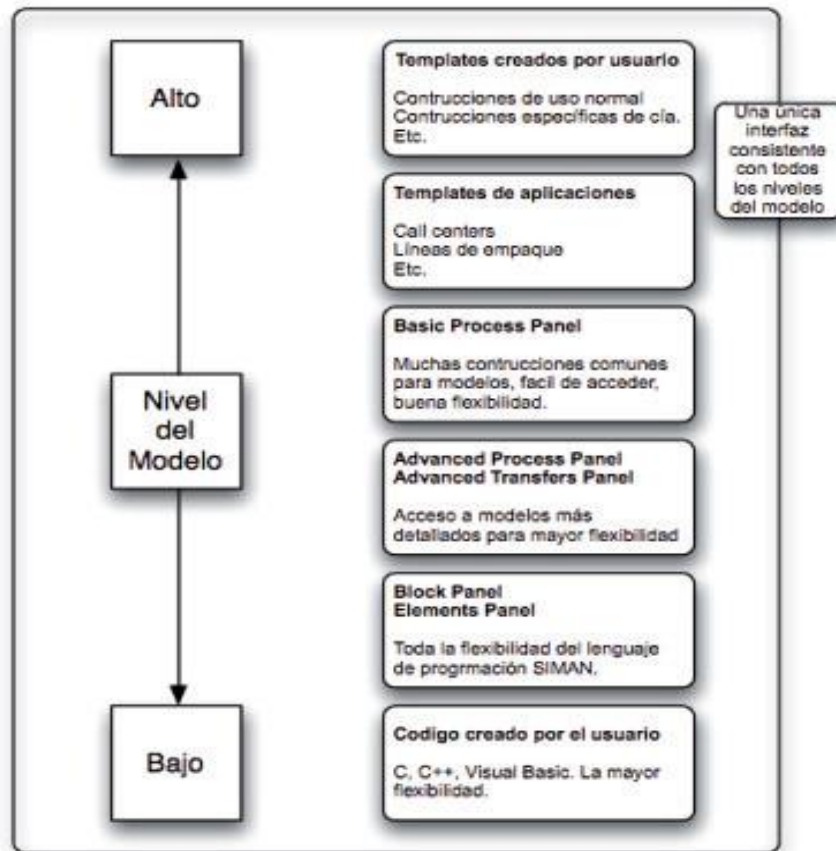


Figura 2.2 Estructura Jerárquica de Arena.

2.1.2. Entorno de ARENA

La ventana principal del software ARENA presenta tres regiones o ventanas correspondientes a:

- Barra de Proyectos
- Barra de Herramientas
- Ventana de diagrama de flujo o Modelo
- Ventana de hoja de cálculo

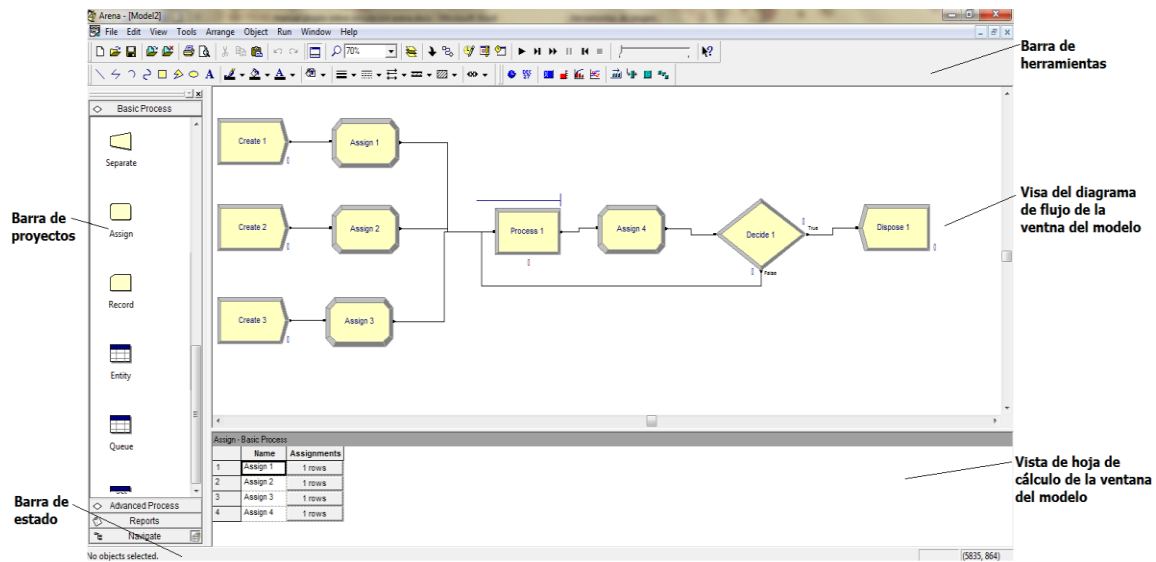


Figura 2.3 Entorno del Software ARENA.

La *Barra de Proyectos* presenta en el nivel más alto los *Templates*, estas son las construcciones de mayor complejidad que pueden ser creadas por los usuarios o ser incluidas en Arena. Normalmente representan comportamientos específicos generados para una industria o compañía en particular. Luego de los *Templates* en la escala jerárquica de Arena, encontramos a los *Paneles*. Estos contienen módulos con cierto nivel de complejidad que son comunes a gran cantidad de modelos y sirven como elementos básicos para la construcción de los mismos. El panel *Basic process* contiene módulos que permiten armar gran cantidad de modelos y están destinado a los usuarios que recién comienzan a utilizar Arena, sin necesitar demasiado desarrollo. El panel *Advanced Transfers* y el *Advanced Process* contienen en muchos casos, desagregaciones del panel Basic Process, que permiten a los modeladores generar comportamientos más específicos para actividades “menos estándar”. En un nivel aún más bajo encontramos a los *Blocks* y los *Elements*, estos módulos son espejos directos del lenguaje SIMAN. Y por tanto son los que mayor flexibilidad de uso presentan, pero al mismo tiempo son los más complejos de utilizar y requieren de usuarios con mayores conocimientos.

2.1.3. Módulos y bloques

Existen dos tipos de módulos en ARENA:

Módulos de flujo (iconos de color amarillo): se utilizan para construir el modelo, y para ello se arrastran de la *Barra de Proyectos* a la *Ventana de Modelo*, y se conectan de acuerdo al sistema que se desea construir.



Figura 2.4 Módulo de flujo.

Módulos de datos (iconos rectangulares azul y blanco) no se ubican en la ventana de Modelo, sino que se editan mediante un mecanismo similar a las hojas de cálculo y se visualizan en la ventana inferior a la ventana del Modelo, llamada *Ventana de Datos*. Estos módulos sirven para definir las características de los diferentes módulos del proceso como son las colas, recursos y variables [18].

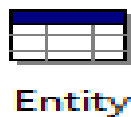


Figura 2.5 Módulo de datos.

Panel Basic Process

El panel *Basic Process* es usado para la construcción de modelos y consta de ocho módulos de flujo y seis módulos de datos que se describen a continuación:

- **Create:** genera una entidad, recurso o pieza, en el campo Entity Type se escribe el nombre del tipo de la entidad que se va a generar. [18].
- **Dispose:** Elimina la entidad o recurso. Es el final de un proceso o almacenamiento.
- **Process:** Este módulo se ha diseñado como el principal método de procesamiento de las entidades en la simulación. Dispone de las opciones *Seize*, *Delay*, *Release* ('capturar' 'demorar' y 'liberar') para cualquier recurso.
- **Decide:** Este módulo permite realizar procesos de decisión en el sistema de simulación; esto incluye opciones de toma de decisiones basadas en una o más condiciones o basado en una o varias probabilidades (p.e. 75% verdadero; 25% falso). Las condiciones se pueden basar en los valores de los atributos, valores de las variables, tipo de entidad o en una expresión.
- **Batch:** Este módulo constituye un mecanismo para el agrupamiento de entidades en el modelo de simulación. Las agrupaciones pueden ser permanentes o temporales. Estas últimas requerirán que se utilice un módulo *Separate* para separar las entidades agrupadas. Los agrupamientos pueden realizarse basados en un número específico de entidades o basados en un atributo determinado. Las entidades que llegan a un módulo *Batch* se sitúan en una cola hasta que se

acumule el número requerido de entidades. Una vez acumulados se creará una entidad representativa de dicho agrupamiento [18].

- **Separate:** Este módulo se utiliza para copiar una entidad entrante en múltiples entidades o para separar una entidad previamente agrupada mediante el módulo *Batch*. En este segundo caso, la entidad temporal representativa desaparece y se recuperan las entidades originales que constituían el agrupamiento.
- **Assign:** Este módulo se utiliza para asignar a las entidades que entren al módulo nuevos valores a variables, atributos de entidades, tipos de entidades, dibujos de entidades y otras variables del sistema. Se pueden realizar múltiples asignaciones en un único módulo *Assign*.
- **Record:** Este módulo se utiliza para recoger las estadísticas de la simulación del modelo. Los tipos de estadísticas disponibles incluyen tiempo de salida del módulo, estadísticas de las entidades (tiempo, coste, etc.), observaciones generales y estadísticas de intervalos de tiempo.

Panel Advanced Process

Se compone por 14 bloques los cuales se describen con mayor detalle a través del botón HELP, aquí se describirán los bloques usados para este trabajo.

- **Size:** Bloque utilizado con el propósito de ocupar o capturar algún recurso, generalmente compartido como: máquinas, robots etc.
- **Release:** libera un recurso, el recurso compartido.
- **Delay:** modela un retardo, se usa para simular una espera en un proceso.
- **Match:** es un componente que nos permite emular una transición de la RdP ya que si hay dos o más condiciones previas que se cumplan se da paso a la etapa siguiente.
- **Hold:** Este bloque mantiene una entidad en una cola esperando por una señal, por una condición específica que llegue a ser verdadera (scan) o podría mantenerla infinitamente.

Panel Blocks

Proporciona un acceso completo al lenguaje de simulación SIMAN, se compone por 69 bloques de los cuales se puede encontrar su descripción a través de HELP.

- **Branch:** Bloque utilizado con el fin de clasificar las entidades en base a unas condiciones en particular. Si tal entidad cumple con la expresión escrita en ese *Branch* la misma será enviada a una lógica diferente que aquellas que no cumplen con tal condición.
- **Duplicate:** Este bloque crea duplicados de una entidad de llegada y los envía a un bloque destino, se debe agregar la cantidad de duplicados que se desean,

especificando: cantidad a duplicar (Quantity to Duplicate) y el costo y tiempo de la duplicación asignada (Duplicate Allocation) igualmente.

- **Scan:** Este bloque permite monitorear alguna entidad dentro del modelo, se usa para dejar que continúe o detener dicha entidad hasta que se cumpla la condición dada, ya sea en una fila (*Queue*) creada dentro del mismo bloque *Scan* o una fila creada por el modelador.

Variables en ARENA

ARENA posee un conjunto de variables predeterminadas que pueden ser usadas o referenciadas, algunas de estas variables no están disponibles en todas las versiones de ARENA. Estas variables son frecuentemente necesarias para recoger estadísticas, datos, estado de recursos y particularmente en el presente trabajo construir condiciones para conocer el mercado de los lugares. Para obtener más información se puede revisar la guía de variables de ARENA.

2.1.4. Ejemplo de Aplicación en ARENA

Fabricación de un producto

En este ejemplo se desea simular el proceso de fabricación de un producto que está compuesto por tres elementos: dos tapas (la superior y la inferior) y el interior. Se requiere calcular cuantas unidades del producto final es posible fabricar en un mes (30 días) con jornadas de 8 horas.

Las tapas llegan a la línea de fabricación según un proceso de Poisson de media 5 tapas/hora. El 50% son tapas superiores y el otro 50% inferiores. Una vez recibidas, es necesario pintarlas para lo que pasan de una en una; por un proceso de pintura cuya duración es independiente de la clase de tapa que se trate; se ha comprobado que se distribuye según una triangular de tiempo mínimo 6, medio 9 y máximo 12 minutos. Hay un control de calidad del proceso de pintura que separa las tapas correctamente pintadas (el 95%) de las defectuosas, las cuales vuelven al proceso de pintura de nuevo.

Por otra parte, el elemento interior del producto final, llega a la línea de fabricación empaquetado en cajas de 3 unidades, siguiendo una distribución exponencial de media 64 minutos. El proceso de desempaqueado lo realiza una máquina que tarda en realizar el trabajo un tiempo que se distribuye según una uniforme entre 30 y 50 minutos. Además, esta misma máquina separa las unidades defectuosas (el 10%) y las envía a chatarra.

Posteriormente, se tiene una máquina que hace el ensamblaje de una tapa superior, una inferior y un elemento interior para construir el producto final. El tiempo de ensamblado se distribuye según una normal de media 15 minutos y varianza 10 minutos.

En Arena hay más de una forma para modelar un problema, este sistema se ha visualizado como proceso, donde las partes (entidades dinámicas), se desplazan por el modelo y compiten por capturar un pintor y dos máquinas: el primer recurso para pintar las tapas, una máquina para desempaquetar y la otra para el ensamble del producto final. El flujo del proceso puede verse en la siguiente figura.

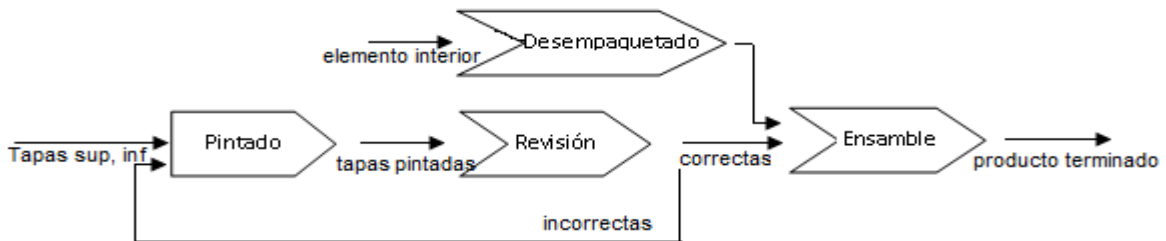


Figura 2.6 Secuencia del proceso de aplicación en ARENA.

A partir de este diagrama se procede a modelar el sistema en ARENA, haciendo uso en primera instancia de los bloques *Create* para crear las entidades que representan las diferentes partes del producto, en dicho bloque se configuran las especificaciones del problema con respecto a la distribución; seguidamente se agrega un módulo *Assign* para agregar un atributo a las entidades, en este caso se asigna un tipo a cada parte: a las tapas inferior y superior *Tipo 1* y la parte interior de *Tipo 2*, se le asigna un tiempo actual (TNOW) de llegada que servirá para visualizarlo al final de la simulación en los *displays*; además de un dibujo para identificar las partes.

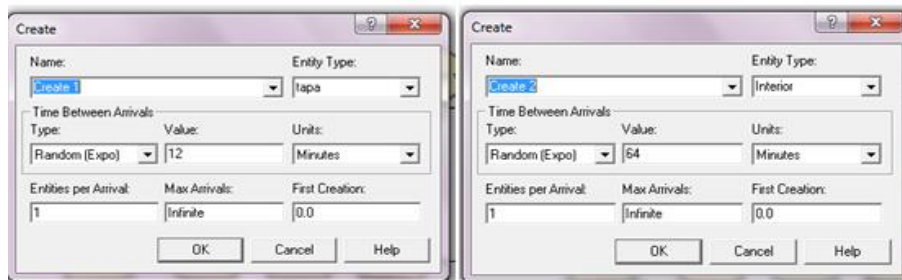


Figura 2.7 Configuración bloques *Create*.

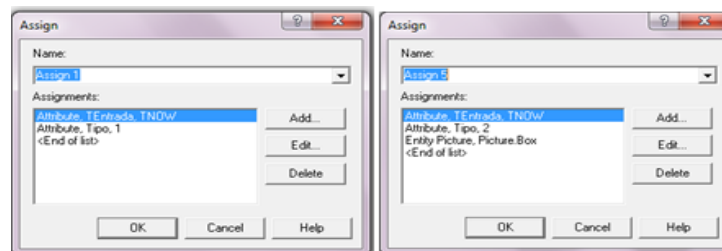


Figura 2.8 Configuración primeros módulos *Assign*.

Por la rama en que entran las tapas superior e inferior se usa un módulo *Process*, donde se realiza el pintado, este se debe configurar de tipo (*Seize Delay Release*) con el propósito de capturar, demorar y liberar el recurso pintor, a continuación un módulo *Decide* hace la función del control de calidad, sus salidas representan las partes correctas y las incorrectas que serán devueltas al proceso de pintura, el siguiente módulo es un *Assign* para incorporar una variable que permite diferenciar las tapas, 50% inferiores y 50% superiores, y en el bloque *Decide* se dividen y se les asigna un nuevo dibujo a cada parte con el bloque *Assign*.

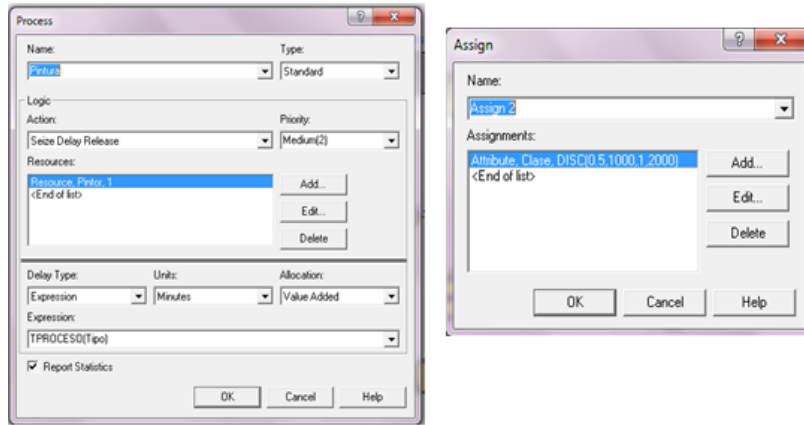


Figura 2.9 Configuración bloque *Process* y segundo bloque *Assign*.

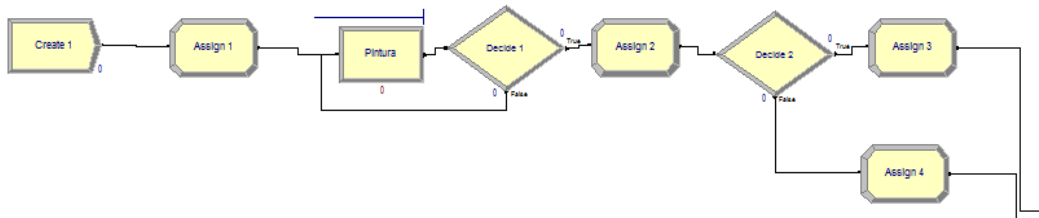


Figura 2.10 Modelado de la evolución de las tapas.

El complemento del modelo consta de la entrada de la parte interior, quien entra empaquetada en caja de tres y necesita de un proceso de desempaqueado que usa el recurso máquina 1 el cual es configurado en un módulo *Process*, luego se usa un módulo *Separate* que de una entidad original permite separarla al número de copias necesarias, para este caso se desean 2 copias mas la original, para emular que se han desempacado, se asigna un dibujo a través de un *Assign* y se usa un *Decide* para separar las correctas de las defectuosas.

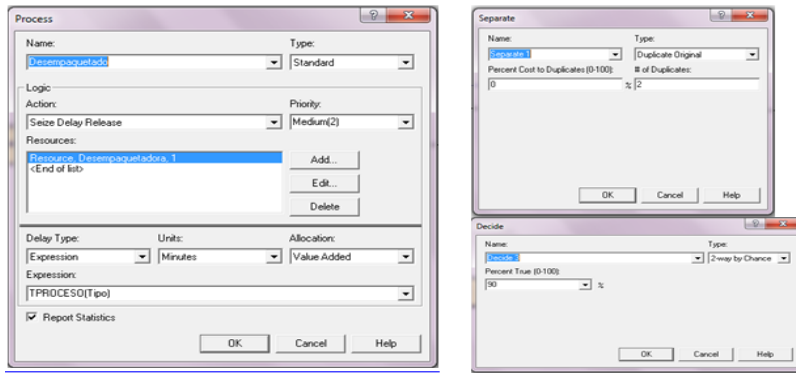


Figura 2.11 Configuración módulos *Process*, *Separate* y *Decide*.

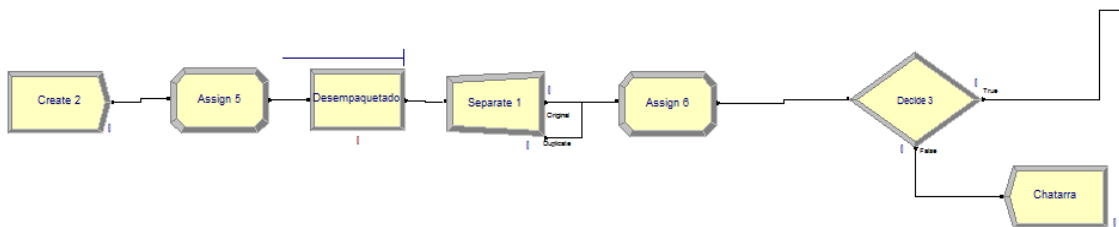


Figura 2.12 Modelo de la evolución de la parte interior.

Finalmente las partes pasaran a en ensamblarse; en un bloque *Match* se reciben las tapas y la parte interior del producto, este bloque solo permitirá el paso al ensamble hasta que haya un componente de cada uno en cola, de ahí pasará al bloque *Batch* que agrupa las tres entidades en una sola para proceder al *Process*, que permite realizar el ensamblado usando como recurso la máquina 2, en un bloque *Record* se guardan los datos del tiempo de fabricación del proceso y en un bloque *Assign* de tipo variable se calcula el tiempo del proceso ($T_{fabricacion} = T_{NOW} - T_{Entrada}$), finalmente se obtiene el producto final y se almacena en un *Dispose*.

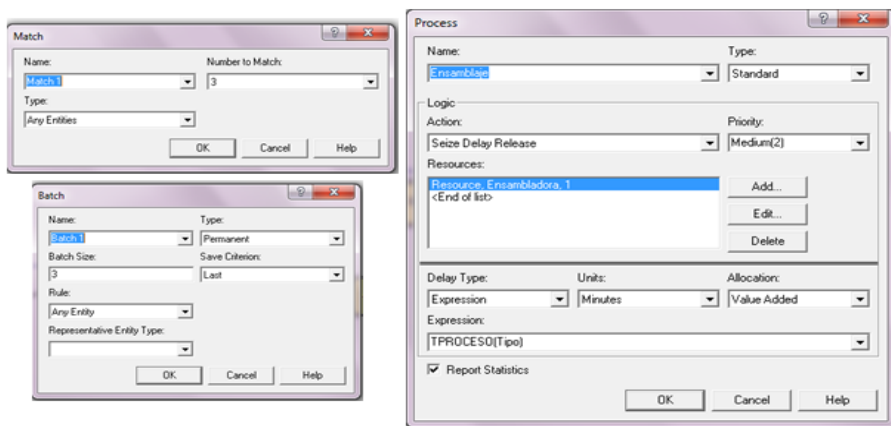


Figura 2.13 Configuración para bloques *Match*, *Batch* y *Process*.

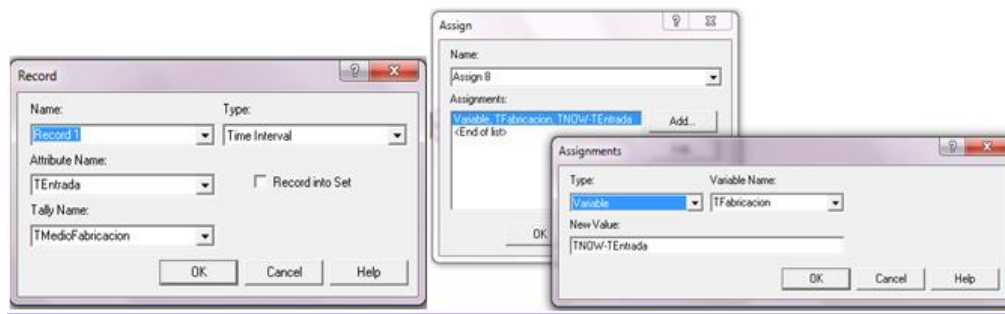


Figura 2.14 Configuración para bloques *Record* y *Assign*.

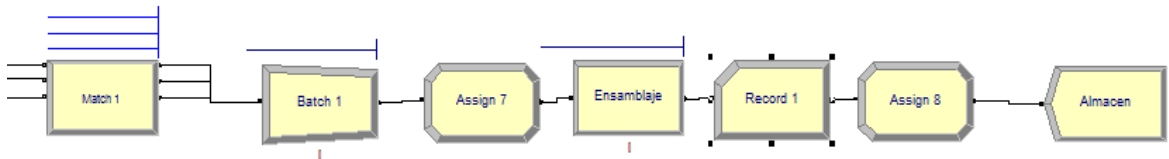


Figura 2.15 Modelo final de etapa de ensamblado para producto terminado.

Para poder ver los resultados finales del modelo se le agregan unos *displays* ubicados en la barra de herramientas, donde se pueden observar durante la simulación algunos datos importantes:



- Calendario y hora
- Número de entidades
- Tiempo de fabricación
- Número de unidades en cada cola del bloque match

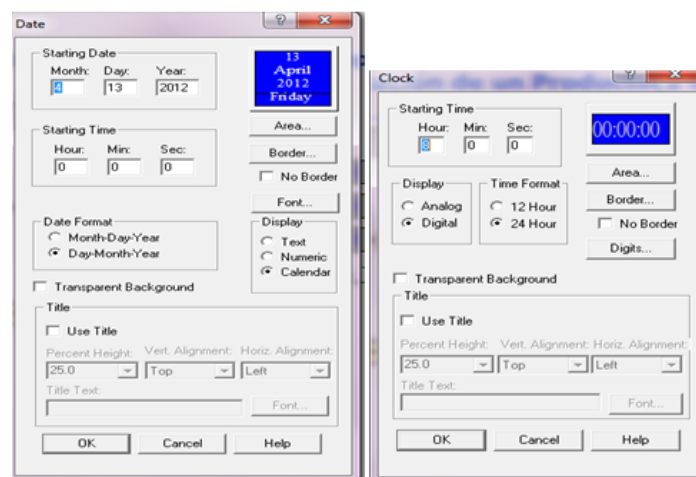


Figura 2.16 Configuración calendario y reloj.

Para la configuración de los displays de las variables, en el espacio *Expression* se indica lo que se desea mostrar: para el número de entidades se incluye, el número de salidas en el *Dispose* llamado "Almacen", para conocer el tiempo de fabricación, en *build expression* de un nuevo cuadro variable se asigna "Tfabricacion" y para conocer todas las entidades en las colas, se crea un nuevo cuadro variable; ubicándose en *Queue* se realiza una sumatoria de todas las colas existentes en el sistema. El modelo está listo para ser ejecutado, bajo las condiciones que se muestran en la figura y se pueden ver los resultados en los diferentes displays.

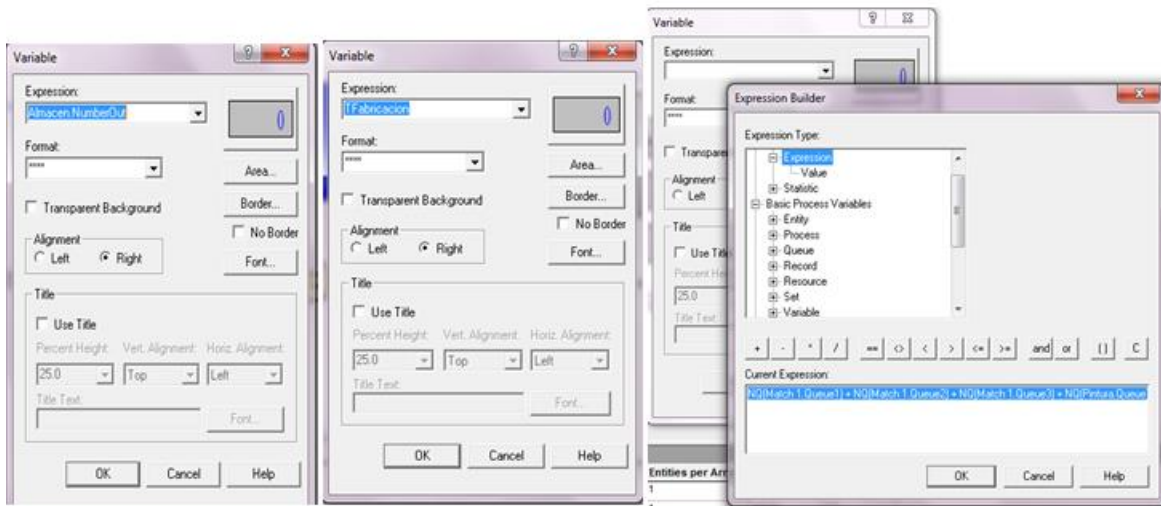


Figura 2.17 Configuración de las variables para los *displays*.

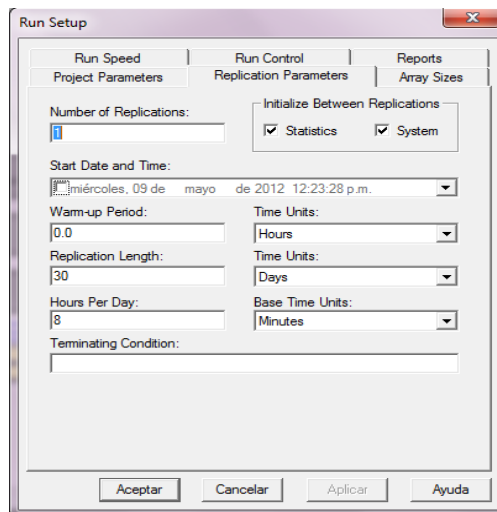


Figura 2.18 Configuración para el *Run Setup* de la ejecución.

El modelo final que describe este ejemplo es:

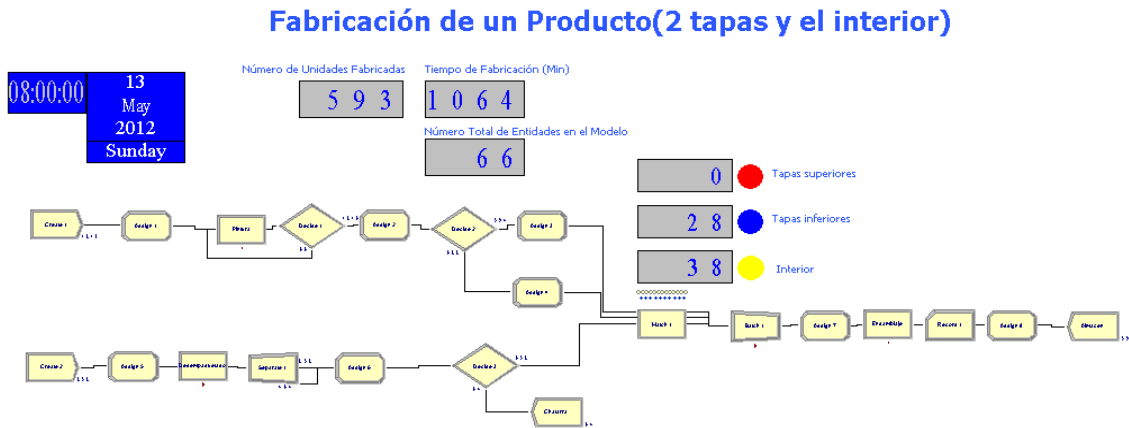


Figura 2.19 Aplicación simulada ejemplo en ARENA.

La estrategia para modelar este ejercicio, ha dependido del conocimiento que tiene el modelador sobre los bloques y módulos del software ARENA, razón por la cual este ejercicio puede presentar serias dificultades entre programadores que apenas empiezan a conocer el software. Entre las desventajas del método usado para construir este modelo se encuentran: existe más de una manera de obtener un modelo que representa al sistema en dependencia del grado de conocimiento que el programador tenga acerca de ARENA, en la construcción del modelo se pierde la estructura lógica del problema, y no es posible saber si el modelo cumple los requerimientos sino hasta luego de correr varias simulaciones y si es del caso, de realizar correcciones al modelo; de ahí, la importancia de modelar sobre un formalismo de fondo que permita conocer la lógica de un proceso y su estructura, identificar bloqueos, errores en la dinámica, entre otros.

2.1.5. ¿Por qué modelar previamente en Red de Petri?

Según Kelton [16], en un paso previo a la simulación, la construcción del modelo puede ser en sí misma un proceso enriquecedor, ya que nos vemos obligados a entender el sistema a estudiar y a formalizar y documentar su funcionamiento. A partir de este proceso se suelen generar alternativas para su configuración y optimización.

El análisis de la estructura de RdP permite alcanzar conclusiones rápidas sobre el comportamiento de un sistema, relacionando propiedades estructurales y de comportamiento. Por eso, capturar los elementos estructurales de la RdP es el primer paso a realizar, una vez el modelador desea estudiar la dinámica de su modelo de simulación.

Las ventajas de los modelos conceptuales en RdP permiten recoger de forma detallada y precisa las relaciones dinámicas entre los diferentes elementos del proceso en estudio, por tanto, constituyen en sí mismos una especificación del modelo del proceso o planta

que se desea analizar a través de la simulación. Adicionalmente el modelo conceptual facilita el dialogo y la coordinación entre todas las partes implicadas en el estudio, constituyendo una representación independiente de la herramienta de simulación escogida [12].

Hallar un método de programación para construir un modelo de simulación en ARENA a partir de una RdP, le permitiría al programador adquirir cierta confianza en el uso de herramientas de simulación para no caer en la tentación de programar directamente un modelo sin previamente haber formalizado el problema, es decir, se busca que el programador no caiga en la problemática de realizar su simulación al “ensayo y error” donde se realizan parches sobre el programa original hasta tener una versión aparentemente correcta. El ejemplo mostrado en esta sección formalizado a través de RdP evita la aparición de problemas añadidos debidos a una mala estructuración de la programación [27].

3. IMPLEMENTACIÓN DE RED DE PETRI EN ARENA

En la literatura se han encontrado trabajos en los cuales se proponen métodos para implementar modelos de RdP en ARENA [26], [27]. Sin embargo, al aplicar los métodos de implementación del modelo RdP en ARENA, en nuestra experiencia se encontró que aquellos llegaban a ser subjetivos, debido a que no se modela el proceso por medio de un estándar, llevando a la obtención de resultados diferentes en la traducción de una misma RdP a ARENA.

En este capítulo se plantea un nuevo método en el cual, se definen estándares de equivalencia entre los diferentes componentes de las RdP y los módulos y módulos de ARENA. Las ventajas que brinda este método son:

- Fácil y rápida implementación del modelo en ARENA
- Permite visualizar la estructura de la RdP en ARENA, conservando así la estructura lógica del problema.
- Ahorro de tiempo al modelar.
- Permite que el programador no caiga en la problemática de realizar su modelo al ensayo y error.
- Elimina la subjetividad a la hora de implementar un modelo RdP en ARENA.

3.1. ASIMILACIÓN DEL FLUJO DE MARCAS EN RDP AL FLUJO DE ENTIDADES EN ARENA

Luego de haber conocido el software, es notable que el flujo de las diferentes entidades por los módulos sea quien genere la dinámica del modelo en ARENA y es de recordar que el comportamiento dinámico de un modelo RdP es debido al flujo de sus marcas a través de la activación de las transiciones [8]. De ahí surge la hipótesis de hacer una analogía entre el flujo de marcas en los diferentes lugares de una RdP, con el flujo de entidades que recorren los módulos en un modelo de ARENA.

Es así como el cambio de una marca de un lugar llamado P1 a otro lugar llamado P2, que ocurre al dispararse una transición cumpliéndose unas condiciones dadas, puede ser modelado en ARENA como el cambio de posición de una entidad de un bloque llamado P1 a otro llamado P2 dándose un conjunto de condiciones entre ellos. Esta situación se ilustra en la figura 3.1.

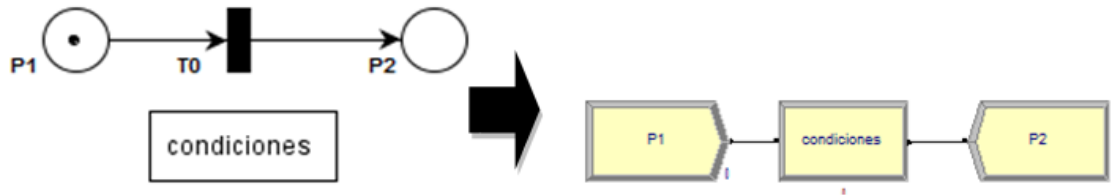


Figura 3.1 Comparación entre el flujo de marcas y el flujo de entidades.

Teniendo en cuenta esta analogía se generan dos inconvenientes: ¿Cómo modelar un lugar que anide entidades hasta que se cumplan las condiciones dadas en el software ARENA? y ¿Cómo modelar una transición que incluya un conjunto de condiciones dadas y que éstas permitan o no el flujo de una entidad entre los módulos en ARENA?

3.2. MODELO DE UN LUGAR DE UNA RDP EN ARENA

En la propuesta que presentamos en este trabajo, el modelo de un lugar de una Rdp en ARENA debe incluir dos características: una variable con la cual se pueda conocer la cantidad de marcas (entidades) presentes en un instante dado; y una condición que mantenga las entidades en dicho lugar mientras no se haya activado una transición que modifique el marcado del lugar.

3.2.1. Lugar con uno o múltiples arcos de entrada y uno de salida

Se propone para la implementación de un lugar de una Rdp con uno o múltiples arcos de entrada y uno de salida, hacer uso de los módulos *Process* y *Hold* conectados en serie como se muestra en la figura 3.2. El módulo *Process* modelará el espacio de anidamiento de marcas y uso de recursos cuando sea necesario, mientras que el bloque *Hold* modelará las condiciones que deben darse para que una marca (entidad) deje el lugar. La variable que definirá el marcado del lugar de la Rdp está dada por:

$$\text{Marcas_en_Px} = \text{Process Px.WIP} + \text{NQ}(\text{Hold Px.Queue})$$

Esta expresión nos indica que el marcado del lugar Px es igual al trabajo en proceso en el bloque *Process Px* sumado con la cantidad de entidades en cola (Queue) del bloque *Hold Px*. Esta variable no se debe definir en ninguna parte o bloque de esta etapa, sin embargo es importante conocer desde este momento que se puede verificar el marcado en cada lugar, ya que será usada en la verificación y validación del modelo, escribiendo esta sentencia en los módulos *Record* que se describen con detalle en el Capítulo 7.

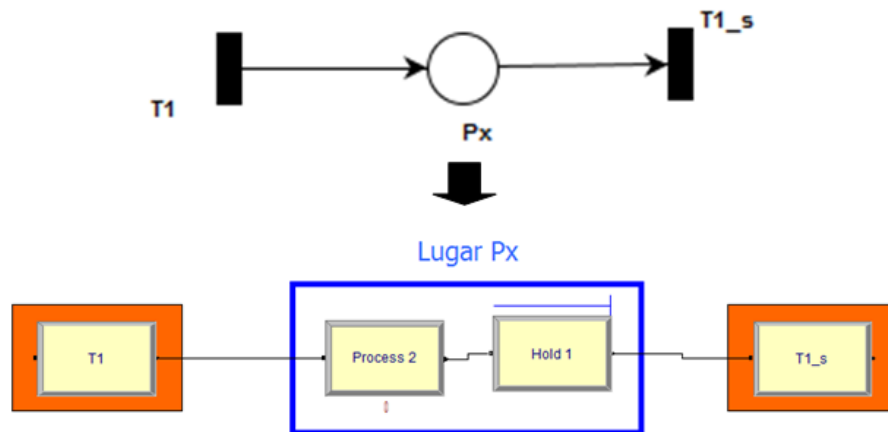


Figura 3.2 Modelo del lugar de una RdP en ARENA.

Marcado inicial.

Para inicializar el marcado de un lugar de una RdP modelada en ARENA se propone el uso de un módulo *Create* al inicio del modelo como entrada al lugar, este debe ser configurado de la siguiente manera. Detalles de la configuración en el Anexo B.

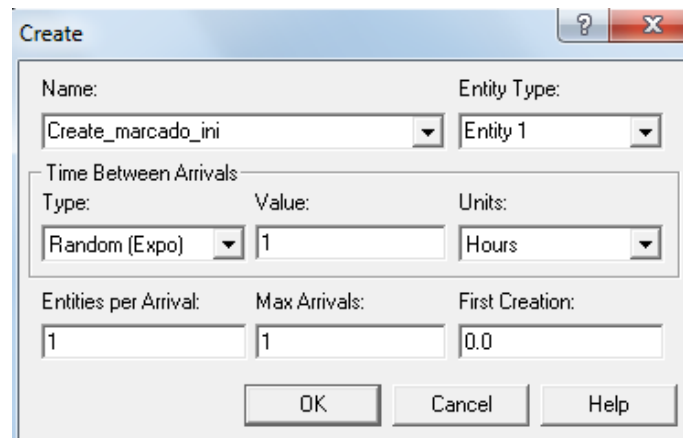


Figura 3.3 Configuración del bloque *Create* para el marcado inicial de un lugar.

Luego de presionar doble clic sobre el módulo *Create* se abrirá una ventana como la de la figura 3.3. En esta debe ponerse un máximo de arribos (Max Arrivals) de 1, y en entidades por arribo (Entities Per Arrival) debe configurarse la cantidad de marcas que se desean como iniciales.

3.2.2. Lugar con múltiples arcos de entrada y salida

Para el modelado de un lugar con múltiples entradas y salidas, se propone el uso de un módulo *Process* y múltiples bloques *Decide*. Si k es la cantidad de salidas del lugar, serán necesarios $k+1$ bloques *Decide*, uno que definirá de manera aleatoria la ruta que tomará la marca o entidad y los restantes verificarán que el camino seleccionado pueda ser tomado por la entidad; es decir, que la transición a la salida puede ser disparada.

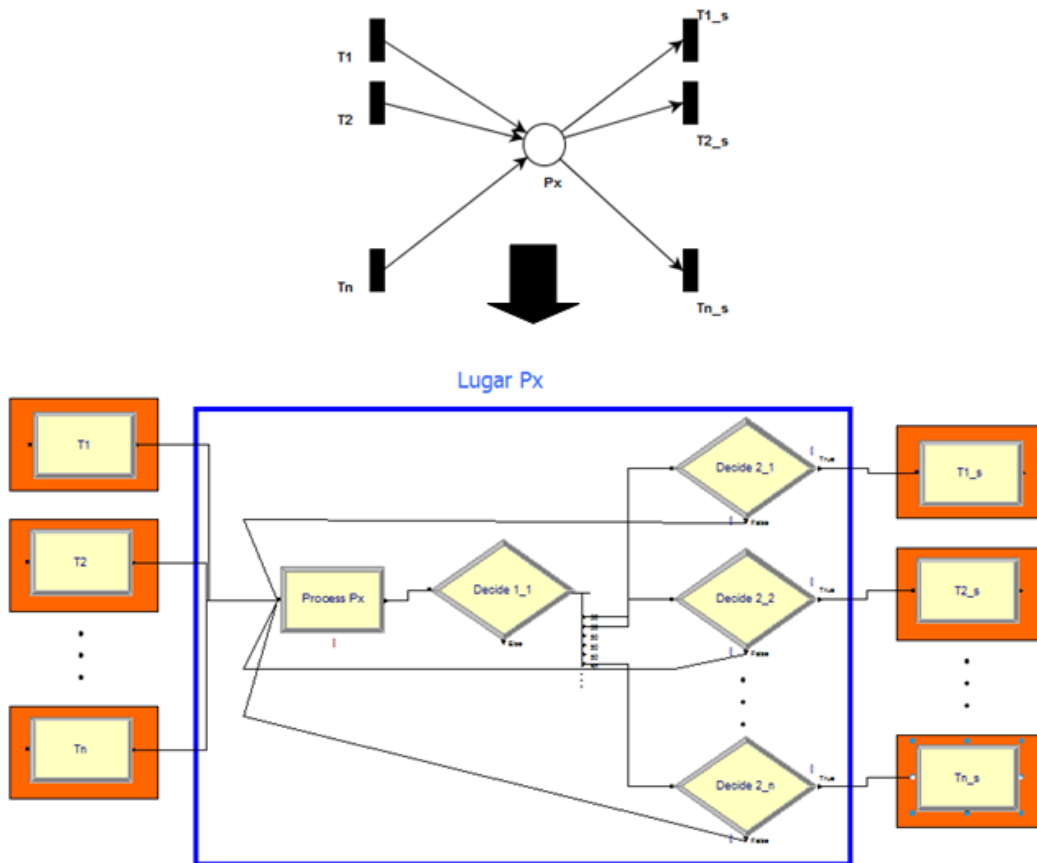


Figura 3.4 Modelo de un lugar con múltiples entradas y múltiples salidas.

El bloque *Decide 1* debe ser configurado del tipo N-vias por probabilidad (N-ways by chance) y se deben agregar tantas ramas como salidas sean necesarias y su probabilidad debe ser la misma. Los restantes deben ser configurados del tipo 2-vias por condición (2-ways by condition), con el fin de verificar que el camino seleccionado sea el correcto (verificar que la transición siguiente a esa salida pueda ser disparada) de no ser correcto este retornará la marca al lugar. La variable que define el marcado de este tipo de lugares es:

$$\text{Marcas_en_Px} = \text{Process Px.WIP}$$

Para modelar las múltiples entradas a un lugar de una RdP no es necesaria mayor modificación, ya que el bloque *Process* posee como característica ventajosa que se le pueden conectar múltiples rutas a su entrada. Al igual que en el primer tipo de lugar esta variable solo se usara en la verificación y validación.

3.2.3. Lugar asociado a un recurso

Para el modelado de un lugar asociado a la disponibilidad de un recurso (robots, máquinas, personal, etc), no se crea un lugar como los descritos anteriormente, ya que, ARENA posee la capacidad de modelarlos directamente como recursos por su característica de ser un simulador de SED's y esta habilidad debe ser aprovechada, ya que esta herramienta permite animar el recurso en sus posibles estados; libre (*Idle*), ocupado (*Busy*), inactivo (*Inactive*) y fallido (*Failed*). Además de utilizar los módulos dedicados para esta función capturar (*Seize*) y liberar (*Release*). El modelo de un lugar de este tipo será un módulo *Resource* definido en ARENA, el cual, debe ser configurado en el módulo de datos con el mismo nombre, de tipo Capacidad fija (*Fixed Capacity*) y en la capacidad debe definirse la cantidad de marcas iniciales que se desean en este lugar como se muestra en la siguiente figura.

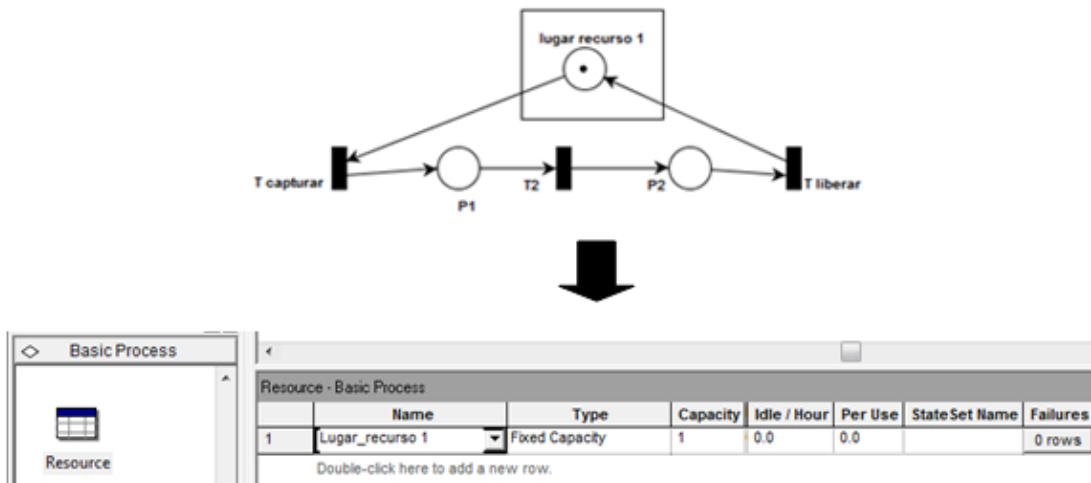


Figura 3.5 Configuración de un lugar asociado a un recurso.

La variable que define el marcado de este tipo de lugares es:

$$\text{Marcas_en_Lugar_recurso} = \text{capacidad} - \text{NR}(\text{Lugar_recurso } 1)$$

Esta expresión nos indica que el marcado en el lugar que modela el recurso es igual la capacidad configurada menos la cantidad del recurso en uso. Esta variable no se debe configurar en este punto, ya que será usada en la verificación y validación Capítulo 7.

3.2.4. Lugares con pesos en sus arcos de entradas o salidas

Para el modelado de este tipo de lugares se propone el uso de dos modelos diferentes: uno para los lugares con peso en los arcos de salida y otro para el peso en los arcos de entrada.

Lugar con peso en el arco de salida

Para el modelado de este lugar se usa un módulo *Process* para generar el espacio donde se realizaran las actividades asociadas al lugar y al anidamiento de marcas, además es necesario usar un bloque *Hold* para incluir las restricciones de avance, finalmente usará un bloque *Batch* el cual agrupará las entidades entrantes en una sola, para modelar el peso del arco en la salida.

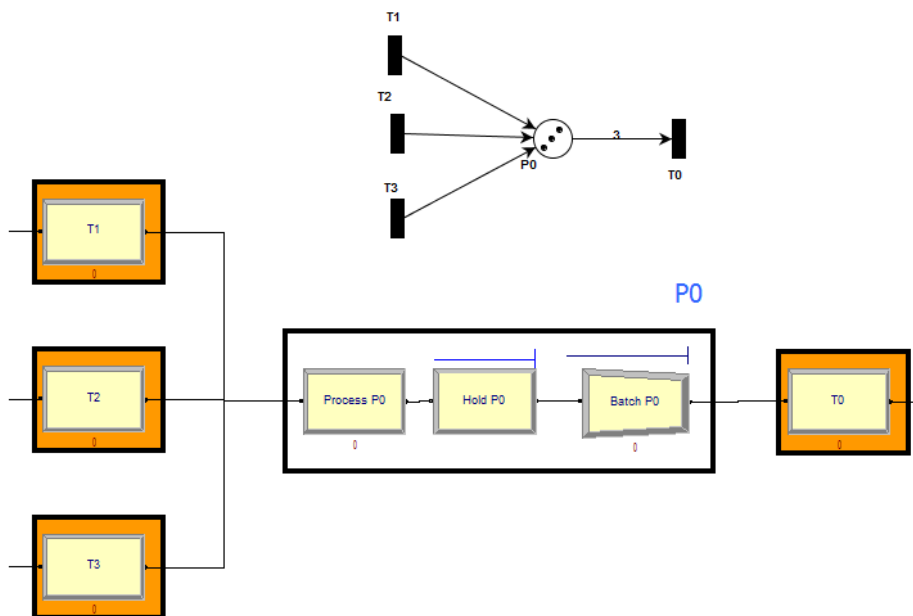


Figura 3.6 Modelo de lugar con peso en el arco de salida.

La configuración de las restricciones de avance, se realizan de acuerdo a las entidades necesarias o a los recursos necesarios para la activación de la transición a la salida del lugar. La variable que definirá el marcado del lugar de la RdP está dada por:

$$\text{Marcas_en_P0} = \text{Process P0.WIP} + \text{NQ}(\text{Hold P0.Queue})$$

Lugar con peso en el arco de entrada

Para el modelado de este lugar se usa un bloque *Separate* el cual multiplicará la entidad entrante y modelará el peso del arco en la entrada, seguido de un módulo *Process* para generar el espacio donde se realizarán las actividades asociadas al lugar y al anidamiento de marcas, además se usa un bloque *Hold* para incluir las restricciones de avance de las marcas.

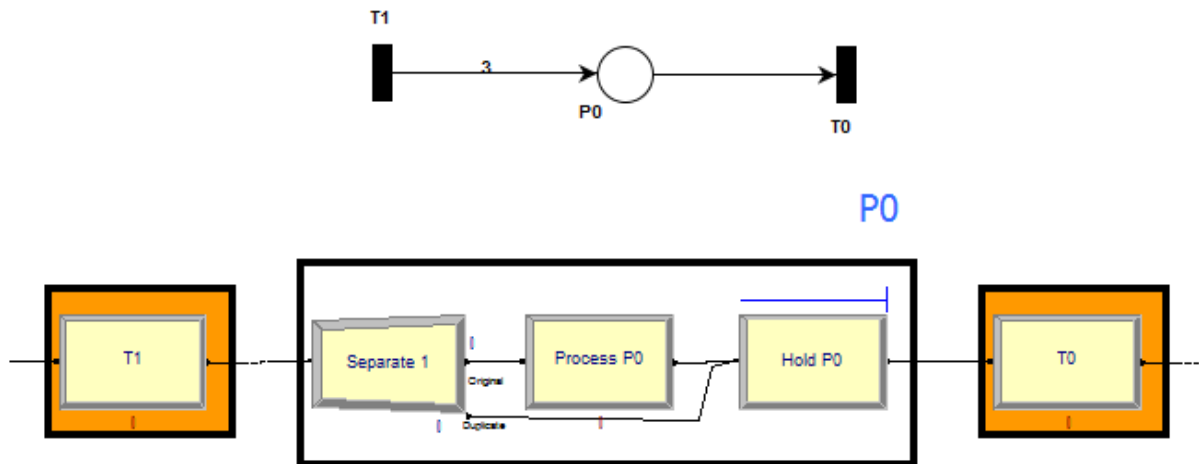


Figura 3.7 Modelo de un lugar con peso en el arco de entrada.

La configuración de las restricciones de avance se hace de acuerdo a las entidades o a los recursos necesarios para la activación de la transición a la salida del lugar y el bloque *Separate* debe ser configurado del tipo duplicar original (*Duplicate original*) y el valor configurado será igual al peso del arco de entrada - 1. La variable que definirá el marcado del lugar de la RdP está dada por:

$$\text{Marcas_en_P0} = \text{Process P0.WIP} + \text{NQ}(\text{Hold P0.Queue})$$

3.3. MODELO DE UNA TRANSICIÓN EN ARENA

Para modelar una transición de una RdP en ARENA se tuvieron en cuenta dos aspectos, primero la cantidad de arcos de entradas y salidas de la transición y segundo el uso de recursos (capturar o liberar). Teniendo en cuenta esto se propone el uso de varios tipos de transiciones.

3.3.1. Transición simple de una entrada y una salida

Esta es la transición más sencilla a modelar, consta de una entrada y una salida implicando la existencia de un evento, pero no tiene condiciones para que ocurra su disparo; se modela a través de un bloque *Delay* con un tiempo de 0, que emula la ocurrencia del evento, para que avance la entidad.

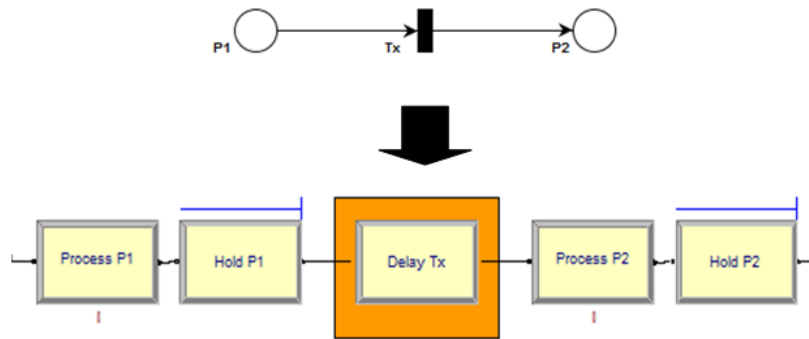


Figura 3.8 Transición de una entrada y una salida.

3.3.2. Transición de múltiples entradas y salidas

La segunda transición identificada es la de múltiples entradas y salidas, esta será modelada por bloques *Match* y *Duplicate*, los cuales deben ser configurados dependiendo de la cantidad de entradas y salidas que tenga la transición de la red de petri.

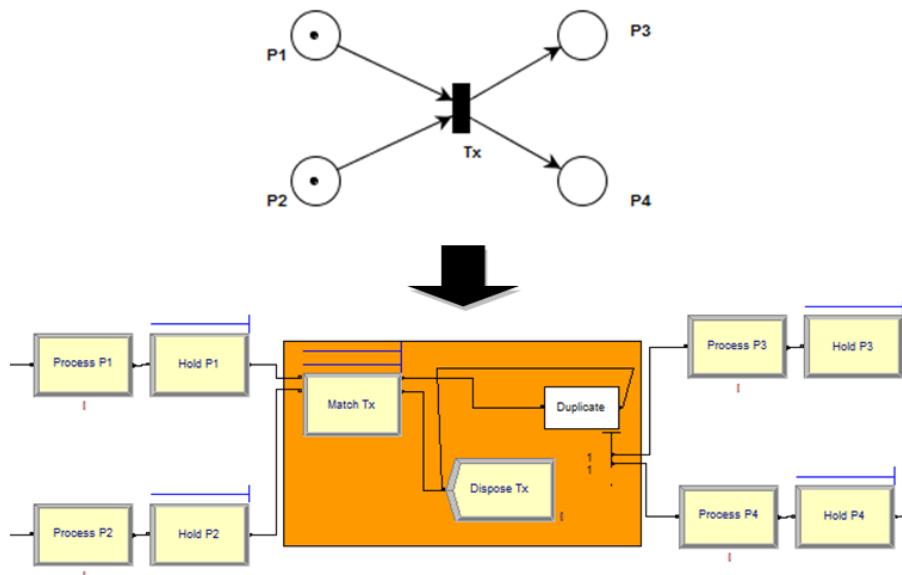


Figura 3.9 Modelo de una transición con múltiples entradas y salidas.

El bloque *Match* servirá para modelar las múltiples entradas a una transición pues este no permite el avance de las entidades hasta haya al menos una entidad en cada una de sus colas (el numero de colas es igual al de las entradas). Las restricciones de avance de los lugares previos deben ser configuradas en los bloques *Hold* al ser definidos del tipo *scan for condition* y las condiciones estarán dadas por:

Condición en el bloque *Hold P1*: $NQ(\text{Hold P2.Queue}) \geq 1$

Condición en el bloque *Hold P2*: $NQ(\text{Match Tx.Queue1}) \geq 1$

Las cuales indican que una entidad en el bloque *Hold P1* no podrá avanzar mientras que no haya al menos una entidad en el bloque *Hold P2*, la segunda restricción indica que una entidad en el bloque *Hold P2* no avanzará si no hay al menos una entidad en la cola1 del bloque *Match Tx*. El uso de estas dos condiciones genera un efecto cascada, el cual, garantiza que las entidades avancen en un mismo instante de tiempo, aunque en la simulación parezca que lo hace una después de otra. Si existen más entradas a la transición las condiciones deberán incluir la sentencia *AND (&&)* en el *Hold P1*, es decir, la condición debe revisar que haya al menos una entidad en cada cola de los bloques *Hold* distintos a la propia, los demás módulos tendrán la misma condición del bloque *Hold P2*.

Por otra parte el bloque *Duplicate* modela las múltiples salidas de la transición ya que cada vez que llegue una entidad a este bloque, se pueden crear varias copias de la entidad para ser enviadas a diferentes lugares. Lo único que se debe configurar es la cantidad de salidas deseadas presionando doble clic sobre el bloque y agregando en (*Add*) tantas ramas como salidas sean necesarias.

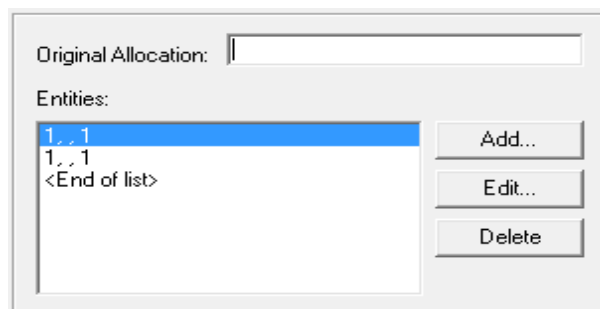


Figura 3.10 Configuración del bloque *Duplicate* para múltiples salidas.

3.3.3. Transición de uso de recurso

Esta transición es de las más sencillas, es la de uso de recurso y será modelada por un módulo *Seize* o *Release* que como se explico en el literal 3.2.3 son propias de ARENA

para el uso de recursos, el cual debe configurarse para capturar o liberar un recurso cuando sea necesario.

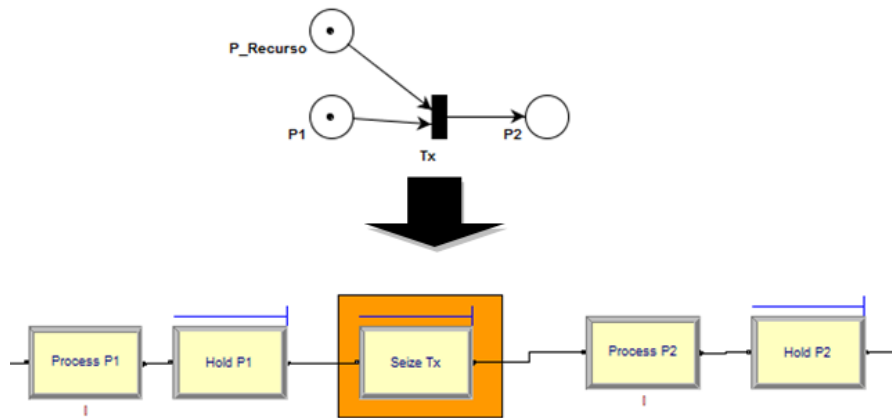


Figura 3.11 Modelo de una transición que captura un recurso.

En la figura 3.9 los bloques *Process* y *Hold* representan lugares de una RdP, el módulo *Seize* modela una transición la cual estará habilitada cuando el bloque *Hold P1* contenga al menos una entidad en cola y el recurso que se desee capturar esté disponible, en este caso la condición del bloque *Hold P1* debe ser configurada del tipo escanear condición (scan for condition) y la condición estará dada por:

$$\text{STATE}(\text{Resource } 1) == \text{IDLE_RES}$$

La expresión indica que el avance de la entidad desde el bloque *Hold P1* hacia la transición estará limitada a la disponibilidad del recurso, es decir, para que la transición pueda ser disparada debe haber una marca en cada lugar previo a esta.

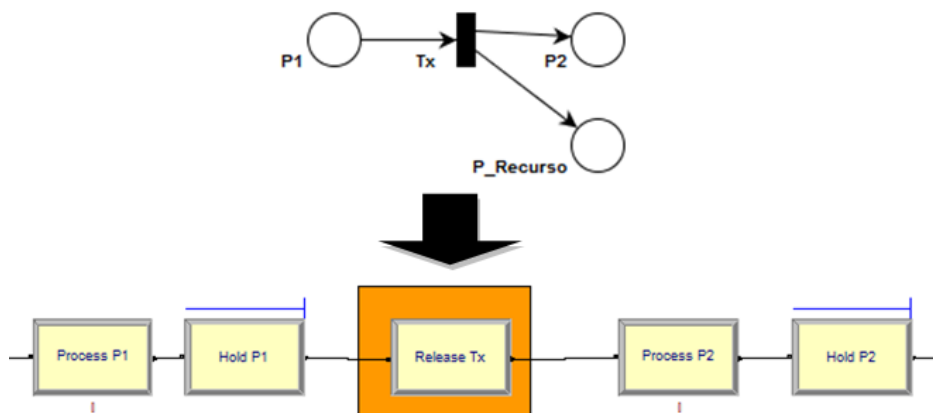


Figura 3.12 Modelo de una transición que libera un recurso.

Para una transición que libera un recurso, el bloque *Hold P1* debe ser configurado del tipo escanear condición (scan for condition) y la condición deberá ser igual =1, esto indica que

la entidad es libre de avanzar a través de la transición. Una vez la entidad pase por el bloque *Release* este permitirá que la entidad pase al lugar P2 y se libere el recurso, modelando el esquema del paso de las marcas a los lugares P2 y P_Recurso de la figura 3.10.

3.3.4. Transición temporizada

El último tipo de transición modelada en este trabajo es la transición temporizada, la cual incluye un retardo (delay) entre la habilitación y el disparo de la transición. Para el modelado de esta transición, solo es necesario agregar un bloque *Delay* al inicio de la transición con el tiempo requerido. Si la transición es de múltiples entradas será necesario agregar el bloque *Delay* después del bloque *Match*.

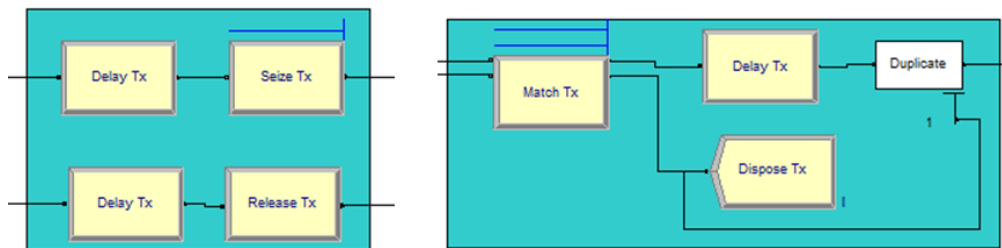


Figura 3.13 Modelos de transiciones temporizadas: (Izq) uso de recurso (Der) múltiples entradas y salidas.

3.4. ESTRUCTURAS BÁSICAS DE LAS RDP Y SUS EQUIVALENTES EN ARENA

Para dejar más claro el método planteado para la traducción de RdP en ARENA, se implementan algunas estructuras típicas de la teoría en RdP [11], con su respectiva equivalencia en este software, haciendo uso de los lugares y transiciones expuestos con anterioridad para ARENA.

Tabla 3.1 Modelo en ARENA de selección.

Estructura RdP	Selección: Selecciona el proceso a ejecutar.
-----------------------	---

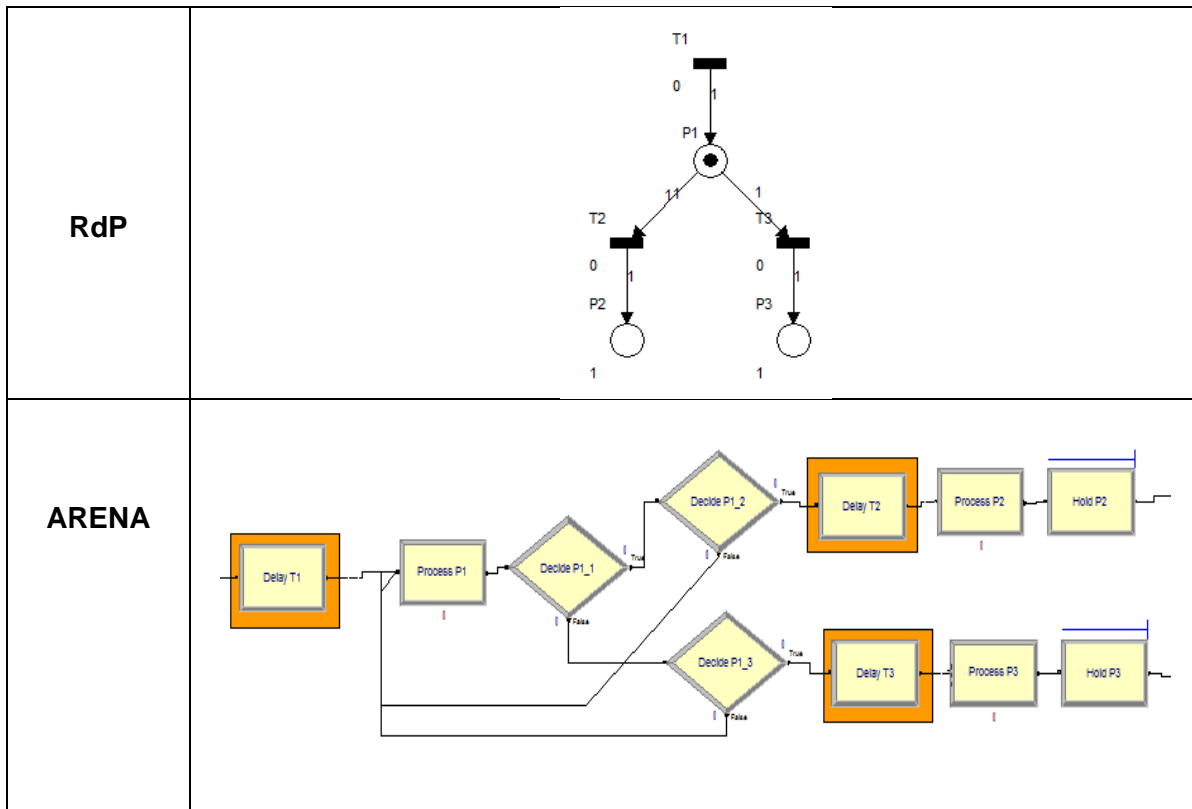
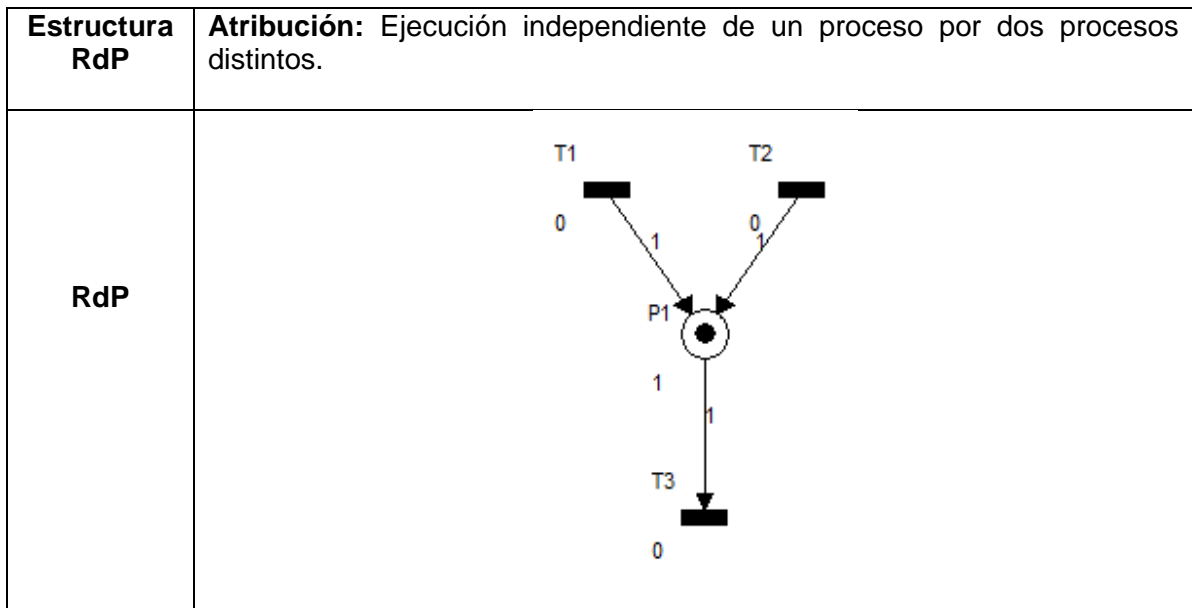


Tabla 3.2 Modelo en ARENA de atribución.



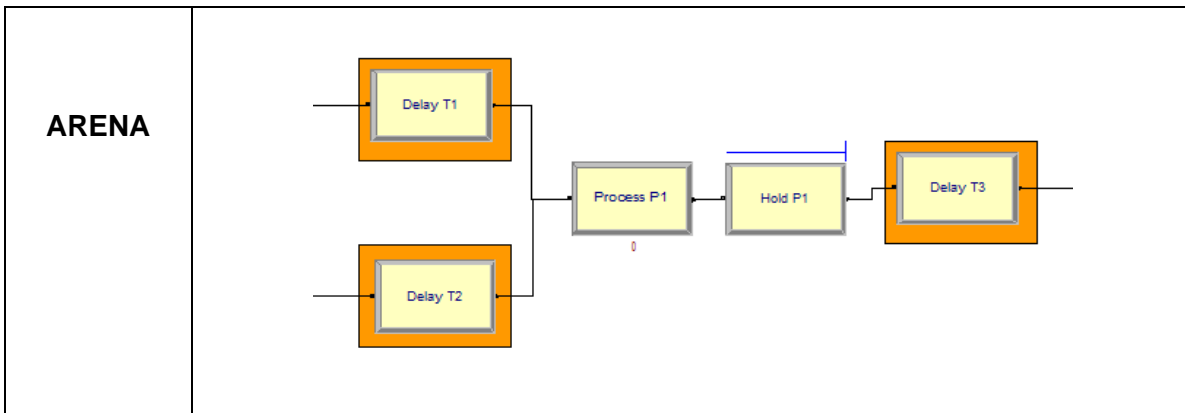


Tabla 3.3 Modelo en ARENA de distribución.

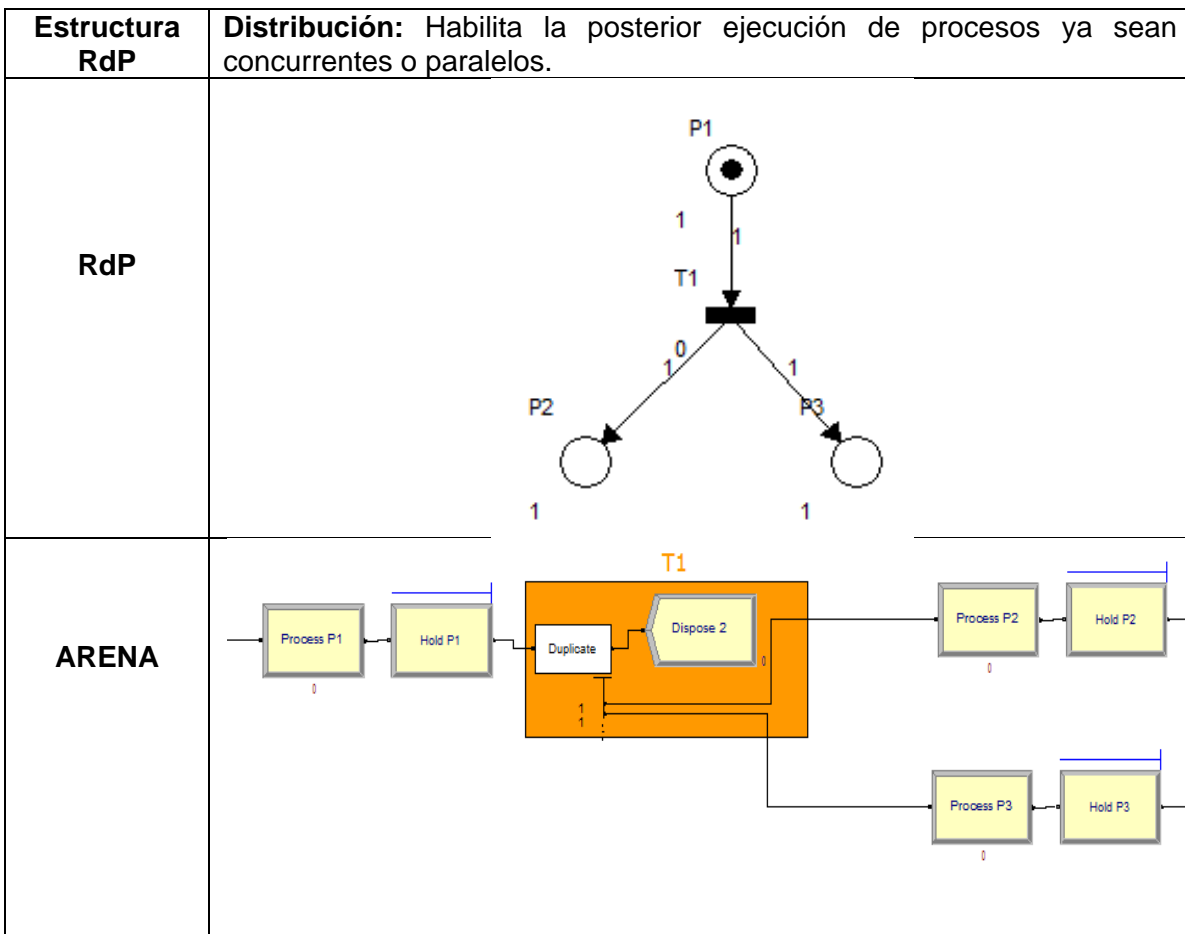


Tabla 3.4 Modelo en ARENA de conjunción o sincronización.

Estructura	Conjunción o Sincronización: Sincronización de procesos en paralelo.
-------------------	---

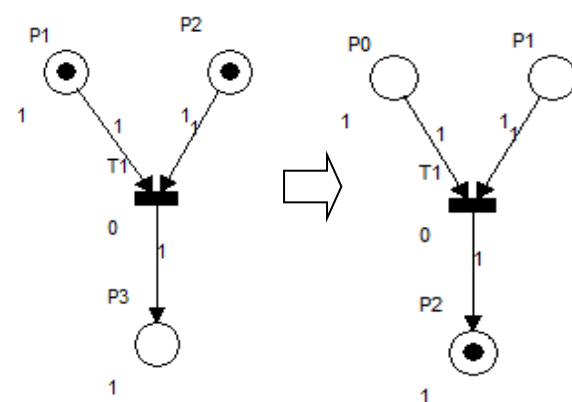
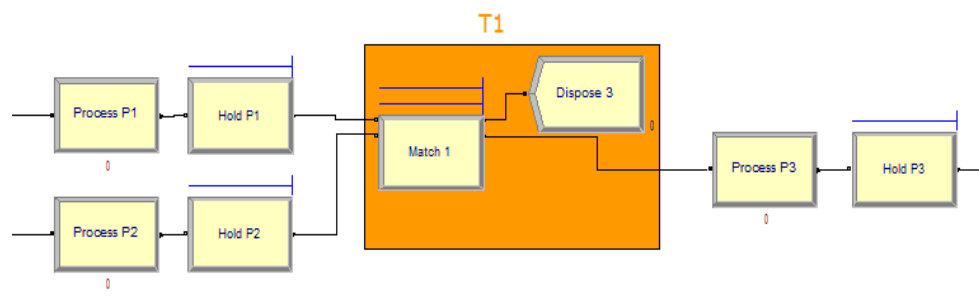
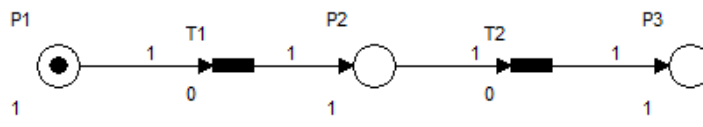
<p>RdP</p>	<p>La transición t_1 estará habilitada para ser disparada si todos los nodos de entrada de la transición t_1 poseen al menos una marca en cada uno de ellos. Esto ocurre principalmente en la electrónica. Ej: El en circuito AND las dos entradas producen una salida lógica.</p>
<p>RdP</p>	
<p>ARENA</p>	

Tabla 3.5 Modelo en ARENA de ejecución secuencial.

<p>Estructura RdP</p>	<p>Ejecución Secuencial: La transición t_2 puede ser disparada si solo si se ha disparado antes t_1.</p>
<p>RdP</p>	

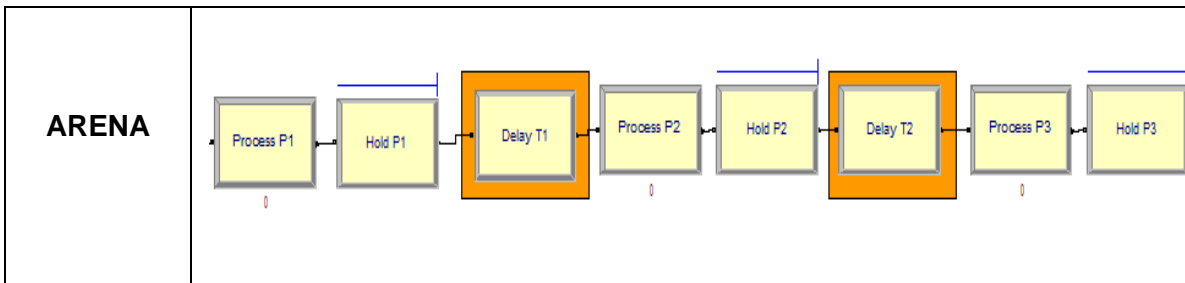


Tabla 3.6 Modelo en ARENA de concurrencia.

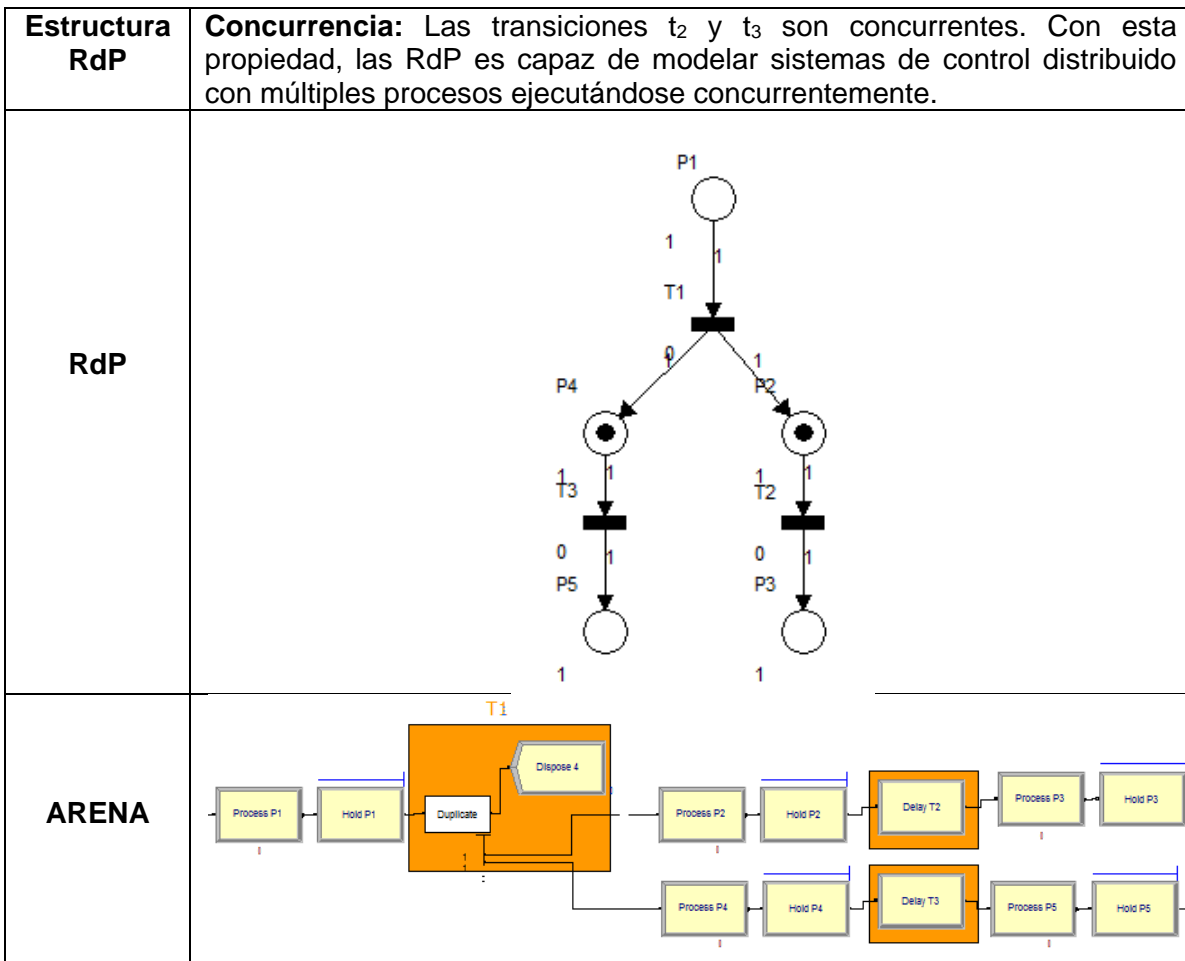


Tabla 3.7 Modelo en ARENA de conflictos.

Estructura RdP	<p>Conflictos: Tanto la transición t_1 como t_2 están listas para ser disparadas, pero el disparo de alguna de ellas produce que la otra transición quede inhabilitada para ser disparada.</p>
-----------------------	---

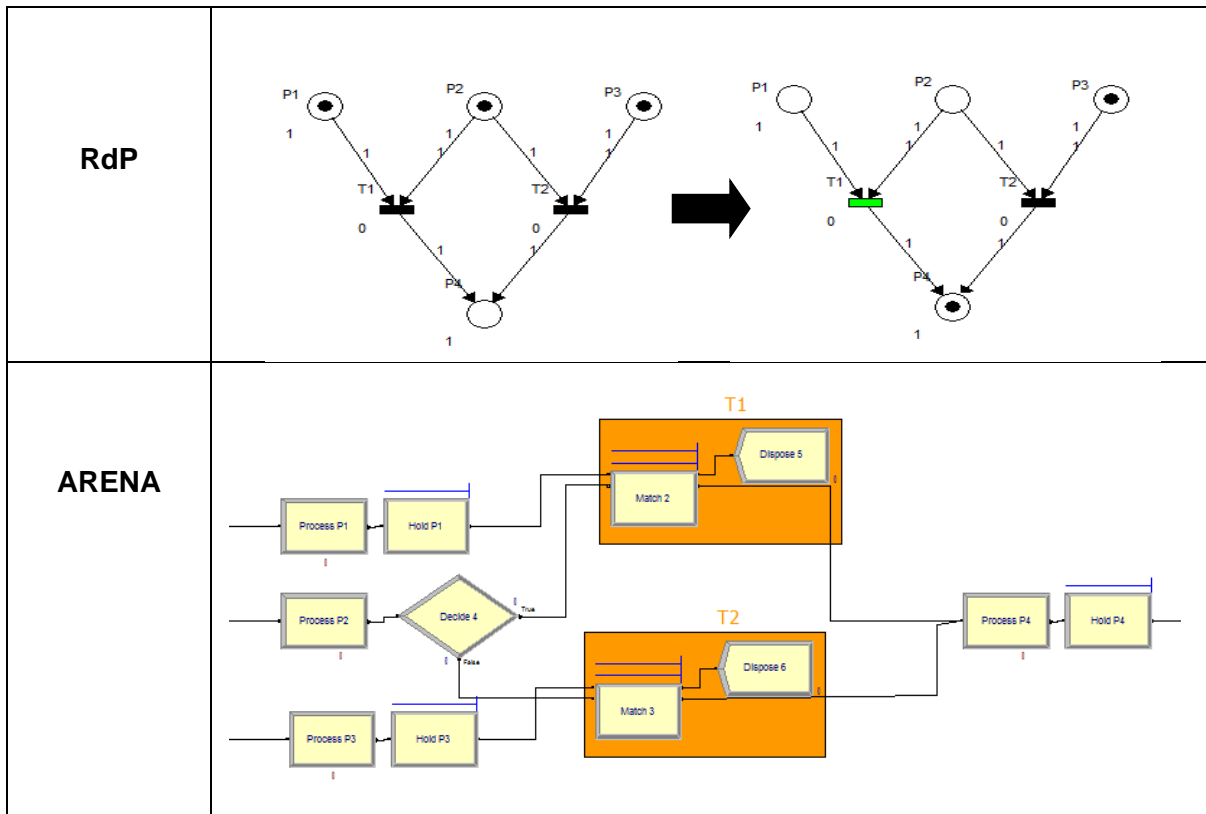
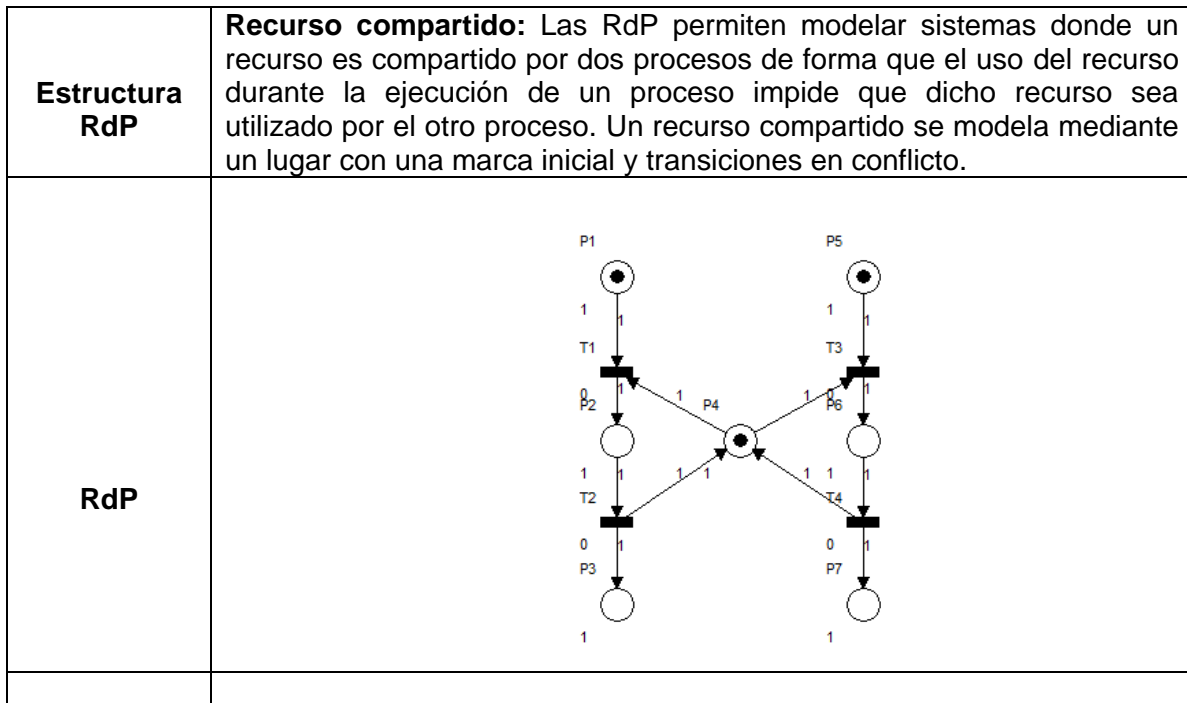
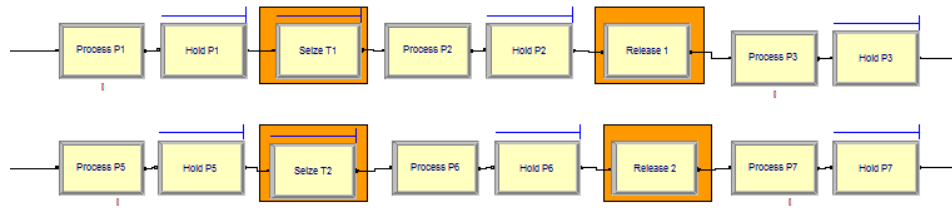


Tabla 3.8 Modelo en ARENA de recurso compartido.



ARENA



Resource - Basic Process			
	Name	Type	Capacity
1	P4_recurso	Fixed Capacity	1

4. METODOLOGÍA PARA EL CÁLCULO E IMPLEMENTACIÓN DE SUPERVISORES BASADOS EN RDP

De las metodologías encontradas en la literatura para el diseño de supervisores basados en redes de Petri se destacan las que usan:

- Arcos inhibidores [27].
- las Redes extendidas GHENesys [28].
- Invariantes de lugar [29].

Siendo elegido para este proyecto el trabajo desarrollado por Moody y Antsaklis [28]. El cual define el supervisor de una manera formal y matemática, haciendo uso de invariantes de lugar, que son ampliamente conocidas e incluyen la controlabilidad y observabilidad de las transiciones, además de permitir agregar restricciones al disparo de las transiciones. Adicionalmente esta selección se ha realizado pensando en la implementación del supervisor en el software ARENA, ya que con esta metodología bastará con agregar lugares estándar de control, mientras que para las otras formas de diseño de supervisores era necesario agregar conceptos poco conocidos como: matrices de inhibición y diferentes tipos de lugares para las redes GHENesys.

La metodología seleccionada será descrita en el presente capítulo con el objetivo de hallar un supervisor basado en RdP que cumpla con un conjunto de restricciones dadas y garantice el desempeño y la seguridad del sistema modelado, en esta parte, se usa como ejemplo un proceso de dosificación al cual se le aplican los diferentes métodos para hallar algunos de los lugares de control del supervisor, haciendo uso de las funciones creadas en Matlab. El ejercicio totalmente supervisado se puede encontrar en el Anexo C.

Finalmente, se realiza la descripción del método planteado por los autores de este proyecto para la implementación de supervisores basados en RdP en el software ARENA

4.1. METODOLOGÍA PARA EL DISEÑO DE UN SUPERVISOR EN RDP

A menudo es necesario regular o supervisar el comportamiento de los SED, con el fin de satisfacer los criterios de rendimiento o seguridad. Los supervisores SED se utilizan para asegurar que el comportamiento de la planta no viole ningún conjunto de restricciones bajo una variedad de condiciones de funcionamiento. Las acciones de regulación del supervisor están basadas en la observación de los estados de la planta, resultando en un control retroalimentado [28].

Un sistema de control supervisor para SED se compone de tres elementos: 1) el sistema que será controlado, comúnmente llamado *planta*, 2) el comportamiento requerido para la planta, al cual se le denomina *especificación o restricción*, y 3) un agente externo,

denominado *supervisor*, el cual manipula la posible ocurrencia de eventos de la planta, para que esta se comporte conforme a la especificación establecida [30].

Para encontrar el supervisor basado en RdP es necesario modelar la planta que se desea supervisar como una RdP de n lugares y m transiciones con matriz de incidencia $D_p \in Z^{n \times m}$, como se describe por Moody y Antsaklis en [28]. El supervisor hallado estará compuesto por los nuevos lugares y las transiciones de la RdP que describen la planta. Su matriz de incidencia es $D_c \in Z^{n_c \times m}$ donde n_c es el número de restricciones. La RdP supervisada deberá tener una matriz de incidencia $D \in Z^{(n_c+n) \times m}$ la cual está compuesta por la RdP que describe la planta y la del supervisor.

El objetivo del supervisor es forzar a la RdP a obedecer restricciones de la siguiente forma:

$$L\mu_p \leq b \quad (4.1)$$

Donde $\mu_p \in Z^n$, $\mu_p \geq 0$ es el marcado de la planta, $L \in Z^{n_c \times n}$ y $b \in Z^{n_c}$ son una matriz y un vector para describir las restricciones.

4.1.1. Invariantes de Lugar

Es una de las propiedades estructurales de las RdP y no depende de su marcado inicial. Las invariantes de lugar son conjuntos de lugares cuyas marcas permanecen constantes para todos los posibles marcos. Una invariante de lugar es representada por un n-vector columna, donde n es el número de lugares de la RdP cuyas entradas diferentes de cero corresponden a los lugares que pertenecen a la invariante en particular [32]. Teniendo claro este concepto se procede con la metodología para hallar un supervisor por invariantes de lugar.

La desigualdad (4.1) puede ser transformada en una igualdad, agregando lugares que representan variables estacionarias convirtiéndola en:

$$L\mu_p + \mu_c = b \quad (4.2)$$

Una ecuación de la forma (4.2) define un invariante de lugar ya que expresa que sin importar las secuencias de disparos en la RdP, el número de marcas en los lugares está sujeto a una restricción que debe cumplirse siempre, y que en este caso es forzada mediante los lugares de control. Esto puede verse como una analogía entre el control de sistemas dinámicos de variable continua, donde las variables de control mueven el punto

de equilibrio de la planta original a un nuevo punto de equilibrio forzado. Mediante el invariante de lugar en (4.2), los invariantes de lugar de la descripción original se ven forzados a modificarse a nuevas posiciones en virtud de las necesidades del diseño.

Donde $\mu_c \in Z^{nc}$, es el vector del marcado del supervisor ya que por cada variable estacionaria agregada se crea un lugar en el supervisor, note que $\mu_c \geq 0$ porque el número de marcas en la RdP no puede llegar a ser negativo. El sistema supervisado tiene la siguiente estructura [28]:

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix} \quad (4.3)$$

$$\mu = \begin{bmatrix} \mu_p \\ \mu_c \end{bmatrix} \quad (4.4)$$

Síntesis de un supervisor basados en invariantes de lugar

Siendo μ_{p0} el marcado inicial de la RdP de la planta y $b - L\mu_{p0} \geq 0$, entonces el supervisor de la RdP $D_c \in Z^{nc \times m}$, con marcado inicial $\mu_{c0} \in Z^{nc}$, puede ser calculado así [28]:

$$D_c = -LD_p \quad (4.5)$$

$$\mu_{c0} = b - L\mu_{p0} \quad (4.6)$$

Por las limitaciones de espacio en este trabajo, los resultados tales como las ecuaciones (4.5) y (4.6) se escriben sin demostración pero se deja al lector interesado en estos detalles la referencia donde se puede encontrar información complementaria.

Ejemplo - Unidad de Dosificación

A continuación se detalla la aplicación de la metodología para el diseño de un supervisor basado en RdP por invariante de lugar, en un sistema de tanque haciendo uso de las herramientas PIPE y MATLAB, para mejor comprensión de la teoría anteriormente explicada. El ejemplo ha sido adaptado de [33] y tomado como un problema interesante de estudio por que se trata de un sistema no controlable, aunque puede obtenerse un supervisor sin bloqueo.

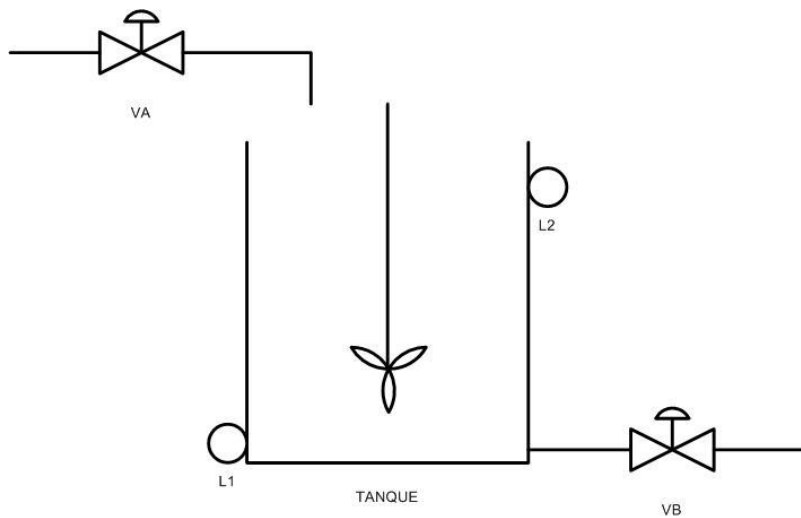


Figura 4.1 Unidad de dosificación.

La unidad de dosificación de la figura 4.1. Consiste de un tanque, una válvula de alimentación A, una válvula de salida B, y dos sensores, que indican cuando el tanque de dosificación está lleno o vacío. Los dos sensores, L1 en el fondo, y L2 en la superficie del tanque, los cuales pueden sensor encendido o apagado. Cuando ambos están apagados, el tanque está vacío. El tanque esta en un estado intermedio cuando el sensor L1 esta encendido y el sensor L2 está apagado. El tanque está lleno cuando ambos sensores están encendidos. Los eventos de los sensores son no controlables. Inicialmente el tanque esta vacío. Las válvulas A y B pueden estar abiertas o cerradas. Los eventos para abrir y cerrar las válvulas son controlables. El fluido debe ser agitado ya que podría volverse gelatina, detectado por el evento no controlable "gel", cuando la sustancia no está en movimiento. El agitador puede estar encendido o apagado, eventos que son controlables. Inicialmente el agitador está apagado. El proceso de gelatinización puede ocurrir cuando ambas válvulas están cerradas, el tanque lleno, y el agitador no está en marcha. Lo que ocurre cuando la sustancia se vuelve gelatina no se modela ya más adelante se desea construir un supervisor que evite el proceso de gelatinización. La RdP que modela este sistema se muestra en la figura 4.2.

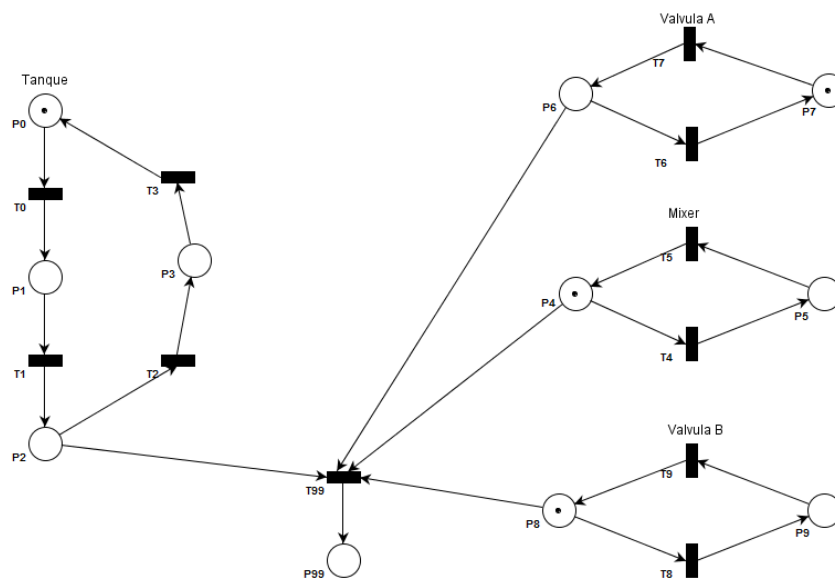


Figura 4.2 Modelo RdP de la unidad de dosificación.

Tabla 4.1 Descripción de lugares y transiciones para el ejemplo tanque dosificación.

Nombre	Tipo	Descripción
P0	Lugar	Tanque vacío
T0	Transición	Sensor L1 encendido
P1	Lugar	Tanque intermedio_llenando
T1	Transición	Sensor L2 encendido
P2	Lugar	Tanque lleno
T2	Transición	Sensor L2 apagado
P3	Lugar	Tanque intermedio_vaciando
T3	Transición	Sensor L1 apagado
P4	Lugar	Mixer apagado
T4	Transición	Apagar mixer
P5	Lugar	Mixer encendido
T5	Transición	Encender mixer
P6	Lugar	Válvula A cerrada
T6	Transición	Abrir válvula A
P7	Lugar	Válvula A abierta
T7	Transición	Cerrar válvula A
P8	Lugar	Válvula B cerrada
T8	Transición	Abrir válvula B
P9	Lugar	Válvula B abierta
T9	Transición	Cerrar válvula B
P99	Lugar	Mezcla gelatinizada (el objetivo es no caer en P99)
T99	Transición	Gel

Para realizar el supervisor de este modelo es necesario definir las restricciones del sistema:

- No se puede abrir la válvula Va y la válvula Vb al mismo tiempo.

Se desea que el supervisor fuerce al sistema a cumplir con dicha restricción:

$$\mu_7 + \mu_9 \leq 1 \quad (4.7)$$

Para agilizar las tareas de encontrar la matriz de incidencia del modelo en RdP y calcular el supervisor se propone el uso de las herramientas PIPE 2.5 y Matlab. Con la primera podemos calcular la matriz de incidencia y el marcado inicial del sistema, estos datos se pueden almacenar en un archivo de extensión *.html* como se indica en el Anexo A. Con la segunda podemos leer la matriz de incidencia y el marcado inicial del sistema usando una función creada para este fin llamada *lectura.m*, con la que se obtienen las matrices MIp (Matriz de incidencia posterior), MIa (Matriz de incidencia previa), up0 (Marcado inicial). La función tiene por argumento el nombre del archivo *.html* que debe estar ubicado en la carpeta de Matlab, en este caso toma el nombre: *"tK_gel.html"*.

Con las matrices de la planta, se define la restricción en (4.7) siendo $L = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0]$ Y $b = [1]$. Luego, en Matlab se pueden calcular las ecuaciones del controlador (4.5) y (4.6)

```
close all
clear all
clc
%obtener matrices de PIPE
[MIp,MIa,up0]=lectura('tK_gel.html');
Dp=MIp-MIa;
L = [0 0 0 0 0 0 0 1 0 1 0];
b=[1];
%calculo del supervisor
Dc1=-L*Dp
uc0=b-(L*up0)
```

Obteniendo:

$$Dc1 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ -1 \ 1 \ -1 \ 1 \ 0] \quad (4.8)$$

$$uc0 = 0 \quad (4.9)$$

Donde *Dc1* es la matriz de incidencia del supervisor y permite identificar el sentido que toman los arcos desde las transiciones de la RdP al nuevo lugar de control y viceversa, así por ejemplo, un 1 indica que un arco va desde una transición de la RdP hacia el lugar de control y un -1 indica que un arco va desde el lugar de control hacia la transición.

uc0 es el marcado inicial del nuevo lugar de control.

La herramienta PIPE ordenó las transiciones de la siguiente manera:

$$T = [T_0 \ T_1 \ T_2 \ T_3 \ T_4 \ T_5 \ T_6 \ T_7 \ T_8 \ T_9 \ T_{99}] \quad (4.10)$$

El lugar de control descrito en esta parte del ejemplo se muestra en la figura 4.3.

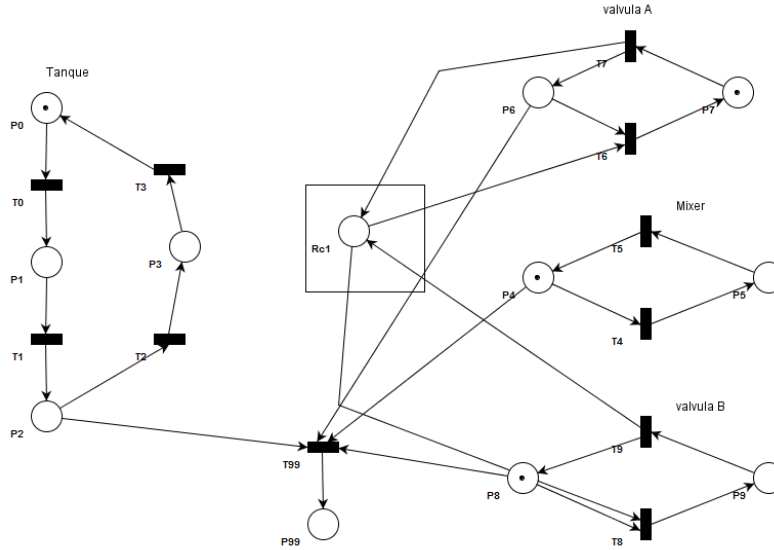


Figura 4.3 RdP con un lugar de supervisión basado en invariante de lugar.

El resultado obtenido cumple con la restricción (4.7), el supervisor completo de este ejemplo puede ser visto en el anexo C.

4.1.2. Restricciones que incluyen el vector del disparo

Ciertas metas de control pueden incluir intervenir sobre el vector de disparo de la RdP, por ejemplo se puede necesitar que una transición no pueda dispararse cuando ciertos lugares contengan marcas. Estas restricciones pueden ser escritas de la forma [32]:

$$\mu_i - q_j \leq 1 \quad (4.11)$$

Se dice que la transición j debe estar deshabilitada cada vez que el lugar i contenga una marca. Para escribir esta restricción de forma que solamente contenga elementos del vector de disparo, el sistema a supervisar debe ser transformado como sigue:

La transición j debe ser remplazada por dos transiciones y un lugar entre ellas, como se muestra en la Figura 4.4. Esta transformación es artificial y no afectará el modelo en RdP del sistema. Su único propósito es introducir un lugar μ_j el cual almacenará el disparo de

la transición t_j , luego de que el supervisor sea calculado, el sistema volverá a ser transformado a su estado original.

El marcado μ_j' de P_j' reemplaza a q_j en la ecuación (4.11) de la restricción, la cual se convierte en:

$$\mu_i - u_j' \leq 1 \quad (4.12)$$

La restricción ahora solo contiene μ_s y el controlador puede ser hallado a través de la forma descrita para invariantes de lugar. Luego de que este sea calculado, las dos transiciones y el lugar son contraídos al lugar original, mientras se mantiene el cumplimiento de la restricción original.

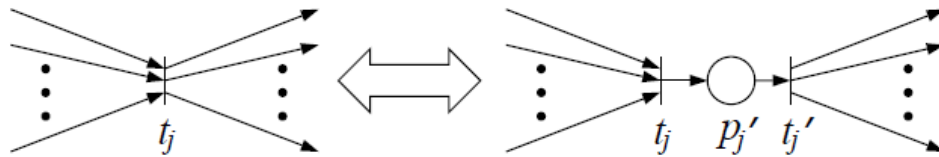


Figura 4.4 Transformación de una transición.

Los arcos que salen del controlador irán normalmente conectados a la transición t_j y los arcos que entran al controlador irán conectados a la transición t_j' . El acto de contraer la estructura transformada, volverá la RdP del sistema a su forma original y los arcos de entrada y salida irán conectados a la transición t_j . A diferencia de los supervisores basados en invariantes este tendrá auto bucles en las transiciones indicadas en las restricciones [32].

Síntesis de un supervisor con restricciones del vector de Disparo

Siendo la matriz de incidencia del sistema transformado $D_p' \in Z^{n+1 \times m+1}$, el supervisor basado en RdP $D_c \in Z^{1 \times m}$, con marcado inicial $\mu_{c0} \in Z^{n_c}$ puede ser calculado así:

$$D_c = -LD_p' \quad (4.13)$$

$$\mu_{c0} = b - L\mu_{p0} \quad (4.14)$$

Ejemplo - Unidad de dosificación con restricción de disparo.

Se desea agregar al ejemplo de la figura 4.2. La siguiente restricción:

- No abrir la válvula Vb hasta que el tanque este lleno

Esta restricción puede traducirse en que la transición T_9 no se dispare, si los lugares P_0 o P_1 contienen marcas.

$$\mu_0 + \mu_1 + q_8 \leq 1 \quad (4.15)$$

Luego de aplicar la transformación descrita en el literal 4.1.2 se tendrá una restricción que solo incluirá el vector de marcados de la RdP ampliada así:

$$\mu_1 + \mu_2 + \mu_{999} \leq 1 \quad (4.16)$$

La transformación hecha a este sistema se puede observar en la figura 4.5. Y debe dibujarse en la RdP original sobre la herramienta PIPE, para obtener la matriz de incidencia del sistema ampliado como se describió en el literal anterior.

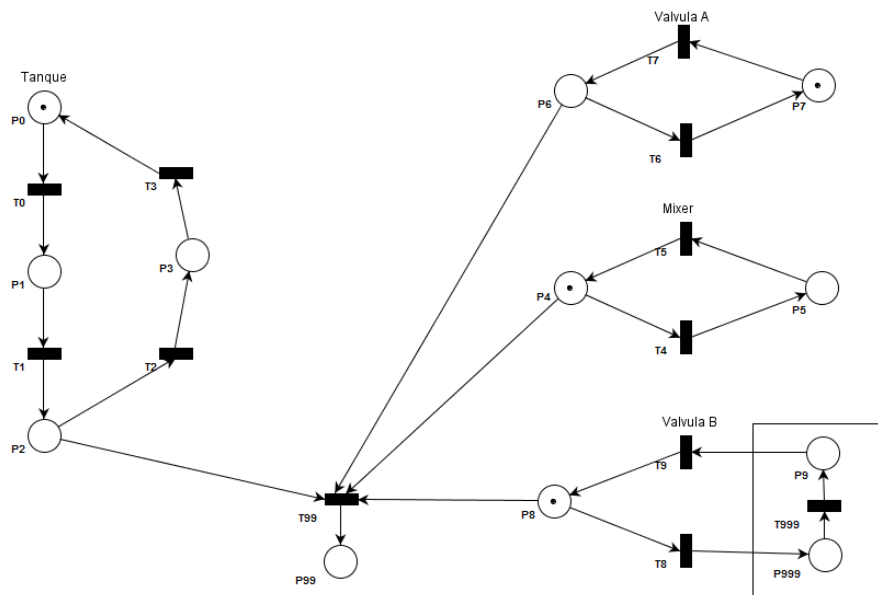


Figura 4.5 RdP del sistema transformado (vector de disparo).

A continuación, usando de nuevo la función desarrollada en *Matlab* llamada *lectura.m*; y después de haber modificado el modelo en RdP del sistema transformado, se obtienen las matrices M_{lp} , M_{la} , up_0 . La función tiene por argumento en este caso, el nombre del archivo *tk_gelRD1.html*.

```
%RESTRICCION 1 DE DISPARO   R3
close all
clear all
clc
%obtener matrices de PIPE
[Mlp,Mla,up0]=lectura('tk_gelRD1.html');
```

```

Dp=MIp-MIa
%matrices L y b
L = [1 1 0 0 0 0 0 0 0 0 0 0 1];
b = [1];
Dc=-L*Dp
uc0=b-(L*up0)

```

Después de ejecutado el archivo se obtiene:

$$Dc = [0 \ 1 \ 0 \ -1 \ 0 \ 0 \ 0 \ 0 \ -1 \ 0 \ 0 \ 1] \quad (4.17)$$

$$uc0 = 0 \quad (4.18)$$

Donde Dc es la matriz de incidencia del supervisor e indica el sentido de los arcos desde las transiciones al lugar de control y viceversa, como ya se explico anteriormente. Esta matriz a diferencia de la (4.8) tiene 12 transiciones, siendo la ultima la artificial agregada para el cálculo del supervisor y los arcos dirigidos hacia esta transición irán hacia la transición original en este caso T8.

El sistema supervisado y contraído a su forma original se observa en la figura 4.6.

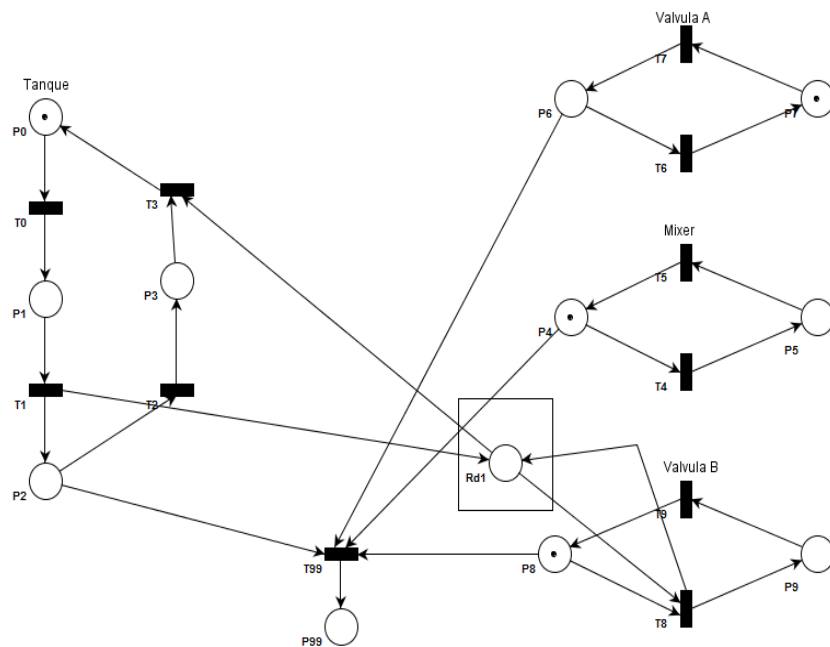


Figura 4.6 RdP del sistema supervisado y contraído

Según la RdP resultante la transición T8 solo se puede disparar cuando los lugares P2 y P3 contengan al menos una marca, es decir que haya una marca en el lugar Rd1

4.1.3. Transiciones No controlables y No observables

Una transición es llamada no controlable si el disparo de esa transición puede o no ser inhibido por una acción externa y una transición es llamada no observable si el disparo de esa transición no puede ser directamente detectado o medido [28].

Para un supervisor basado en RdP, los arcos de entrada y salida de las diferentes transiciones son usados para disparar cambios en el estado del supervisor. Un supervisor basado en RdP no puede tener conexiones con transiciones no observables; todas las transiciones no observables son implícitamente no controlables, sin embargo, las transiciones no controlables pueden o no ser no observables.

Admisibilidad de mercado y de restricciones [28]

Un mercado es admisible si:

$$L\mu_p \leq b, \quad y \quad (4.19)$$

para todos los mercados μ'_p alcanzables desde μ_p a través del disparo de una transición no controlable se cumple: $L\mu'_p \leq b$.

Si cualquiera de estas condiciones no se cumple, entonces el mercado es inadmisibile.

Dada una planta con mercado inicial $\mu_p(0) = \mu_{p0}$, una restricción admisible satisface dos condiciones:

$$L\mu_p \leq b, \quad y \quad (4.20)$$

para todo $\mu_p(N)$ alcanzable desde $\mu_p(0)$ a través de cualquier camino de mercados consecutivos alcanzables $\mu_p(0) \rightarrow \mu_p(1) \rightarrow \dots \rightarrow \mu_p(N)$, donde

$$L\mu_p(i) \leq b, \text{ para } 1 \leq i \leq N, \quad (4.21)$$

$\mu_p(N)$ es admisible.

Si una restricción no satisface ambas condiciones, entonces es inadmisibile. Las condiciones anteriores indican, que el disparo de transiciones no controlables nunca puede llevar de un estado que satisface la restricción a un nuevo estado que la viole.

Si $L\mu_p \leq b$ es inadmisibile, entonces es oportuno encontrar otra restricción $L'\mu_p \leq b'$ tal que $L'\mu_p \leq b'$ sea admisible y que además para todos los mercados μ_p que cumplan con la nueva restricción, también se cumplan con la original. La nueva restricción debe ser admisible y todos los estados alcanzables que satisfacen la restricción original deben ser alcanzados también por la nueva restricción [28].

Admisibilidad para las restricciones asociadas a una Transición no observable

Sea D_{uc} la matriz de incidencia de la parte no observable de la RdP (compuesta por las columnas de la matriz de la planta). $LD_{uc} \leq 0$ Implica admisibilidad. Dada una planta con transiciones no controlables descritas por la matriz de incidencia D_{uc} y una restricción $l^T \mu_p \leq b$, si:

$$l^T D_{uc} \leq 0 \quad (4.22)$$

La restricción es admisible para el sistema dado.

Admisibilidad para las restricciones asociadas a una Transición no controlable

Sea la matriz D_{uo} la matriz de incidencia de la parte no observable de la red de petri (compuesta por columnas de la matriz del sistema o planta)

$l^T D_{uo} = 0$ Implica admisibilidad. Dada una planta con transiciones no observables descritas por la matriz D_{uo} y una restricción $l^T \mu_p \leq b$, si:

$$l^T D_{uo} = 0 \quad (4.23)$$

La restricción es admisible para la planta dada.

Por tanto las restricciones que no cumplan con estas condiciones deben ser transformadas para poder ser cumplidas [28].

Transformación de Restricciones

Suponiendo un conjunto de restricciones $L\mu_p \leq b$, construimos las matrices LD_{uc} y LD_{uo} y observamos si se cumplen las condiciones mencionadas, de no cumplirse es posible que una transformación lineal de la ecuación pueda transformar las restricciones a $L'\mu_p \leq b'$ de tal manera que se cumplan las restricciones originales $L\mu_p \leq b$.

Para solucionar este problema se crea la función de *matlab* llamada *transformB*, la cual, tiene por argumentos el conjunto de restricciones originales (L y b), las matrices de la parte no controlable y no observable (D_{uc} y D_{uo}) y el marcado inicial del sistema. Como resultado nos entrega las matrices L_p y b_p que nos servirán para el cálculo del nuevo supervisor que cumple con las restricciones originales sin estar influenciado por la parte no controlable y no observable del sistema. Por limitaciones de espacio, en esta tesis no se detalla el cómo realizar una transformación lineal, solo se aclarará que en [28] se presentan dos opciones: una a partir de un proceso de operaciones por fila que va detectando el cumplimiento de las condiciones de admisibilidad y en caso de no cumplirse busca reordenar la matriz L y modificar el invariante en b conservando la restricción original. El algoritmo presentado en [28] para este enfoque no fue satisfactorio en nuestra forma de implementación en MatLab y se desconoce si el error está en el algoritmo

reportado o en la implementación realizada. En su lugar, se utilizó el enfoque también planteado en [28] de convertir el problema de admisibilidad en uno de optimización, en el cual se tiene como función objetivo el logro de la ecuación del invariante, sujeto a las restricciones de admisibilidad. Existen algoritmos numéricos que logran la solución de ecuaciones con restricciones. La dificultad adicional está en que el resultado de la solución que se busca, en este caso D_c y u_{c0} , no puede ser de números reales, deben ser enteros ya que están asociados con el balance de marcas de la red. En MatLab existe una solución al problema de optimización con restricciones en el caso de números enteros, pero solo en la versión de enteros binarios mediante la función `bintprog`. Queda una limitante por resolver y es cómo extender este resultado al caso entero en general.

Ejemplo- unidad de dosificación con transformación de restricciones.

Se presenta un nuevo problema al ejemplo planteado anteriormente, se dice ahora que las transiciones t_0 , t_1 , t_2 , y t_3 son no controlables y se desea hacer que el sistema obedezca las siguientes restricciones:

- Nuevamente no abrir las válvulas V_a y V_b al tiempo.
- No debe estar el tanque lleno y el Mixer apagado al mismo tiempo.

Las cuales serán escritas como:

$$\mu_2 + \mu_4 + \mu_6 + \mu_8 \leq 3 \quad (4.24)$$

$$\mu_7 + \mu_9 \leq 1 \quad (4.25)$$

A partir de estas especificaciones, definimos la matriz L así:

$$L = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Ahora podremos revisar la admisibilidad de las restricciones calculando LD_{uc} y LD_{uo} .

$$LD_{uc} = \begin{bmatrix} 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.26)$$

$$LD_{uo} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (4.27)$$

Como se observa, la primera restricción viola la primera condición; por lo tanto, la restricción es no admisible y debe ser transformada usando la función `transformB.m`. El archivo `TK_gela2_2.m` muestra el procedimiento del uso de la función y los supervisores encontrados con y sin transformar el resultado obtenido es el siguiente:

Supervisor original:

$$Dc1 = 0 \quad -1 \quad 1 \quad 0 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 4 \quad (4.28)$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1 \quad 1 \quad -1 \quad 1 \quad 0$$

$$\begin{matrix} uc0 = 1 \\ 0 \end{matrix} \quad (4.29)$$

Supervisor transformado:

$$Dc = 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 4 \quad (4.30)$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1 \quad 1 \quad -1 \quad 1 \quad 0$$

$$\begin{matrix} uc0 = 0 \\ 0 \end{matrix} \quad (4.31)$$

El nuevo supervisor cumple con las restricciones originales sin conectarse con las transiciones no controlables del sistema y a pesar de que presenta un arco conectado a la transición T99 esta nunca estará habilitada para cualquier marcado que se dé en la planta. A continuación se muestra el sistema con su supervisor completo, para observar los cálculos y la definición de las restricciones ver el anexo C.

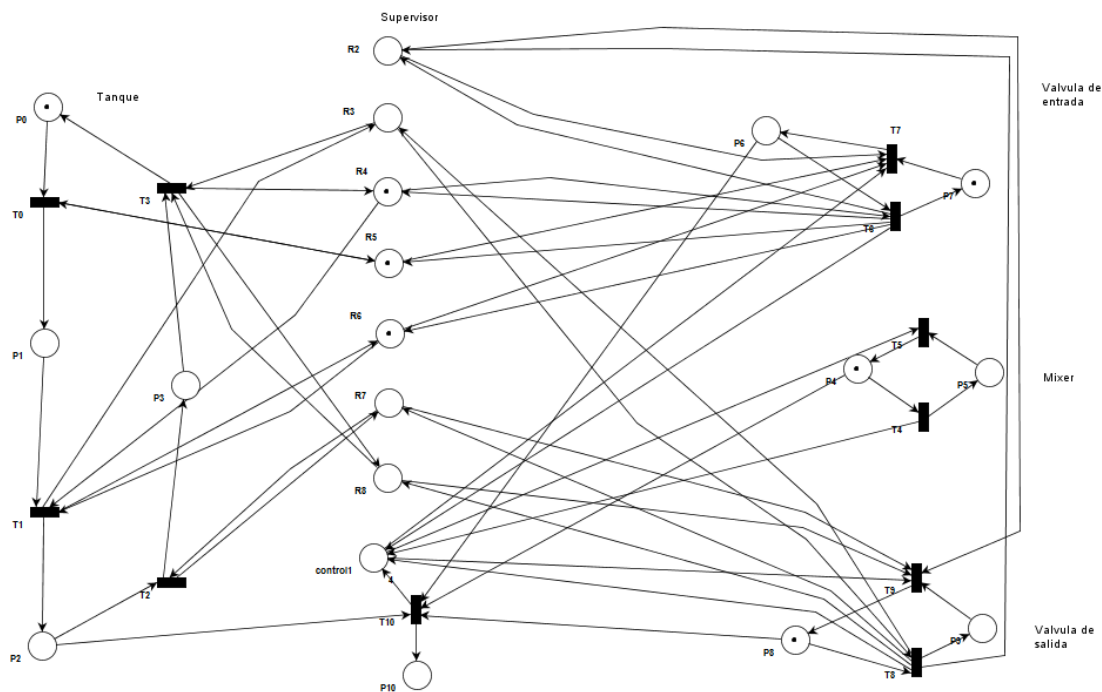


Figura 4.7 Unidad de dosificación supervisada.

4.1.4. Transiciones que incluyen tiempo

Los supervisores de RdP son generalmente conducidos por secuencias de eventos (disparos de transiciones) que ocurren en planta. Algunas transiciones se disparan antes

que otras y algunas pueden ser disparadas simultáneamente, respondiendo a una evolución que incluye tiempo.

Una Red de Petri temporizada trabaja como una RdP ordinaria pero incluye una nueva función de tiempo. En algunas transiciones la función indica la cantidad de tiempo requerido para que se disparen o la cantidad de tiempo que debe transcurrir entre la llegada de una marca a un lugar. El método de supervisor basado en invariantes de lugar puede ser usado sin ningún cambio para RdPT, si la información de temporización de la red se asocia con las transiciones. Debido a que el controlador no tiene nuevas transiciones propias, no es necesario establecer nuevas propiedades de tiempo para el controlador [28]. Esta aclaración no se aplica en el ejemplo de la Unidad de dosificación usado en este capítulo, pero debe tenerse en cuenta para el diseño del supervisor del caso de estudio del literal 5.3.

4.2. MÉTODO PARA LA IMPLEMENTACIÓN DE SUPERVISORES BASADOS EN RDP EN ARENA

Como ya se mencionó, los supervisores basados en RdP están compuestos por las transiciones de la red que se desean supervisar y los nuevos lugares agregados a partir del cálculo de las invariantes. Para introducir el supervisor a una RdP modelada en el software ARENA, es necesario modificar el comportamiento de sus transiciones para que estas se adapten a la nueva dinámica debida a la inclusión de los lugares de control. Por esto se propone el uso de modificadores de transición; los cuales afectaran las transiciones deseadas sin modificar la RdP original, solo sus conexiones.

Para implementar un supervisor en RdP en ARENA es necesario seguir los siguientes pasos:

- Tener el modelo de la RdP sin supervisar en este software.
- Agregar los modificadores de transición (PRE y POST) para incluir el efecto de Incidencia de los nuevos lugares y sus arcos.
- Agregar los lugares de control y realizar las debidas conexiones.
- Es necesario dar prioridad a los lugares de control, eso quiere decir que en los módulos *Hold* o en los *Decide* anteriores a una transición modificada (PRE), se debe agregar una restricción de avance, la cual debe verificar que en el *Match* siguiente al supervisor se esté dando vía libre al flujo de la entidad.

4.2.1. Lugares de supervisión en ARENA

El supervisor en RdP calculado a través del método descrito anteriormente, agrega lugares a la RdP del sistema que se desea supervisar; estos lugares pueden ser de múltiples formas dependiendo de su marcado inicial, el número de entradas o salidas y de

los pesos de sus arcos. Estos lugares son similares a los descritos con anterioridad para la implementación de una red de petri en ARENA, puesto que, van a contener módulos similares.

Lugar de control de una entrada y una salida con marcado inicial

Este tipo de lugar es el más sencillo de implementar y es muy parecido al lugar descrito en el literal 3.2.1, estará constituido por un bloque *Create* el cual inicia el marcado del lugar y un bloque *Process*, el cual solo servirá para conocer el marcado de un lugar en un instante determinado mediante la variable:

Marcado = Process X.WIP

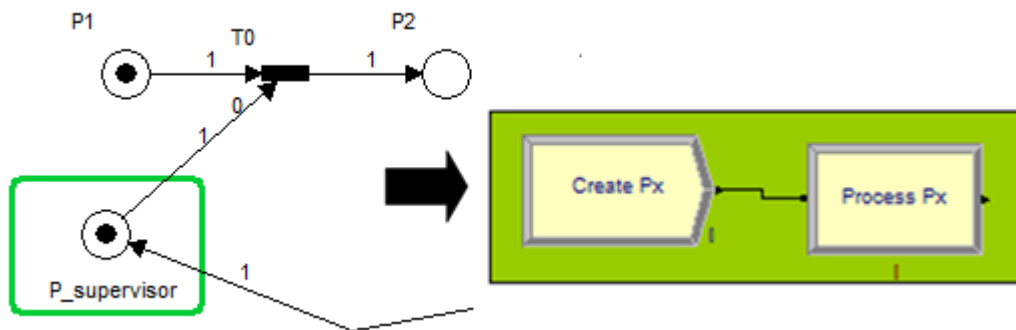


Figura 4.8 Modelo de lugar de supervisión con una entrada, una salida y marcado inicial.

La única configuración requerida para este lugar es la del marcado inicial, solo es necesario poner un máximo de arribos de uno, y en entidades por arriba la cantidad de marcas iniciales. El retorno de marcas provenientes de la RdP del sistema se hará antes del bloque *Process*.

Lugar de control de múltiples entradas y salidas

El modelo de un lugar de múltiples entradas y salidas está representado por un bloque *Process* y un bloque *Decide* conectados en serie, las múltiples conexiones de entrada del bloque *Process* modelan las múltiples entradas a un lugar de una RdP, por otra parte, el bloque *Decide* se configura con el número de salidas deseadas de manera *N-way-by Condition*, la cual permite simular las múltiples salidas de la RdP, dichas condiciones estarán basadas en la presencia o no de entidades en los modificadores de transición y/o en la disponibilidad de los recursos necesarios.

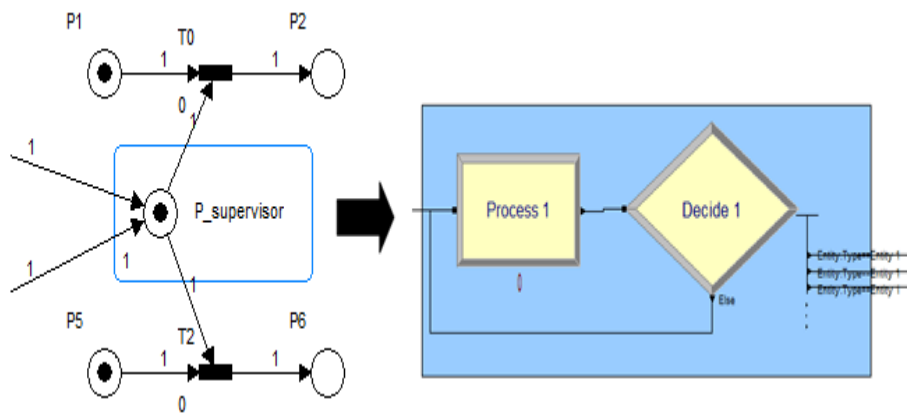


Figura 4.9 Modelo de lugar de supervisión con múltiples entradas y salidas

El retorno de marcas provenientes de la RdP del sistema, se hará antes del bloque *Process* que conforma el lugar de control y el marcado de un lugar en un instante determinado esta dado por la variable:

$$\text{Marcado} = \text{Process X.WIP}$$

Lugar de control de múltiples entradas y única salida con peso en el arco de salida

El modelo de un lugar de múltiples entradas y única salida con peso en el arco de salida, estará representado por bloques *Create*, *Process*, *Match*, y *Dispose*. Serán necesarios tantos bloques *Create* como marcas iniciales se requieran y deben ser configurados con un máximo de arribos de uno y tan solo una entidad por arribo, el bloque *Match* representará el peso en el arco de salida y debe ser configurado dependiendo de dicho peso; por ejemplo, si se desea un peso de tres en el arco de salida, se debe configurar el bloque *Match* de tres entradas y salidas, donde dos de sus salidas estarán conectadas a un bloque *Dispose* y la restante estará conectada hacia el modificador de transición que sea necesario. El retorno de las marcas que provienen de la RdP del sistema se hará una antes del bloque *Process* y las otras antes del bloque *Match* que representan el lugar de control.

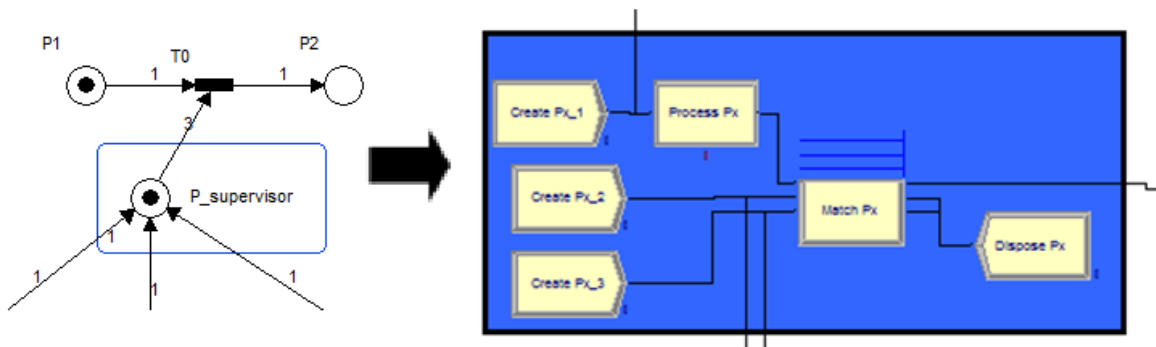


Figura 4.10 Lugar de múltiples entradas y única salida con peso en el arco de salida

El marcado de un lugar en un instante determinado, se puede observar mediante la variable:

$$\text{Marcado} = \text{Process X.WIP} + \text{NQ}(\text{Match Px.Queue 2}) + \dots + \text{NQ}(\text{Match Px.Queue } n)$$

4.2.2. Modificadores de transición en ARENA

El concepto de modificadores de transición, propuesto en este trabajo es necesario para agregar el supervisor a una RdP modelada previamente en Arena, estos deben cumplir dos funciones: habilitar el paso de una entidad teniendo en cuenta las restricciones deseadas y devolver las marcas a los lugares de control cuando sea pertinente. Estos modificadores modelarán la incidencia de las transiciones sobre los lugares de control, de tal manera que afecten el flujo de las entidades por la RdP logrando cumplir con las restricciones propuestas.

Modificador PRE

Un modificador de transición PRE, modela el efecto de un arco desde un lugar de control hacia una transición de la RdP, es decir, inhibirá o habilitará una transición cuando sea necesario, este efecto puede ser logrado mediante el uso de un bloque *Match* conectado previamente a la transición que se desea modificar; sobre la primera entrada y salida debe ser conectada la ruta original de la RdP, mientras que sobre la segunda entrada será conectada la salida del lugar de supervisión y la segunda salida del bloque *Match* deberá ir a un bloque *Dispose*.

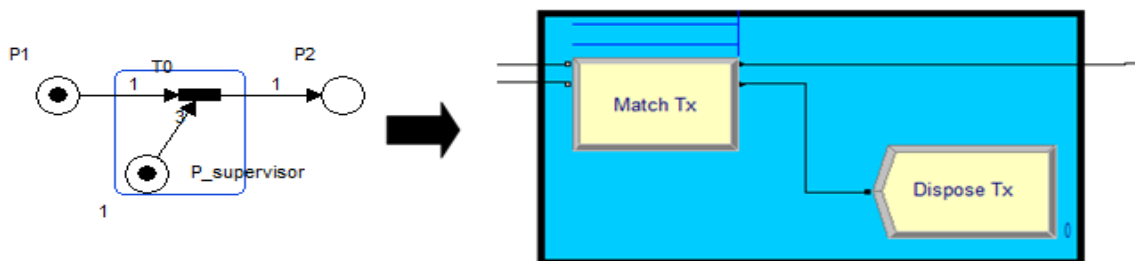


Figura 4.11 Modificador de transición PRE.

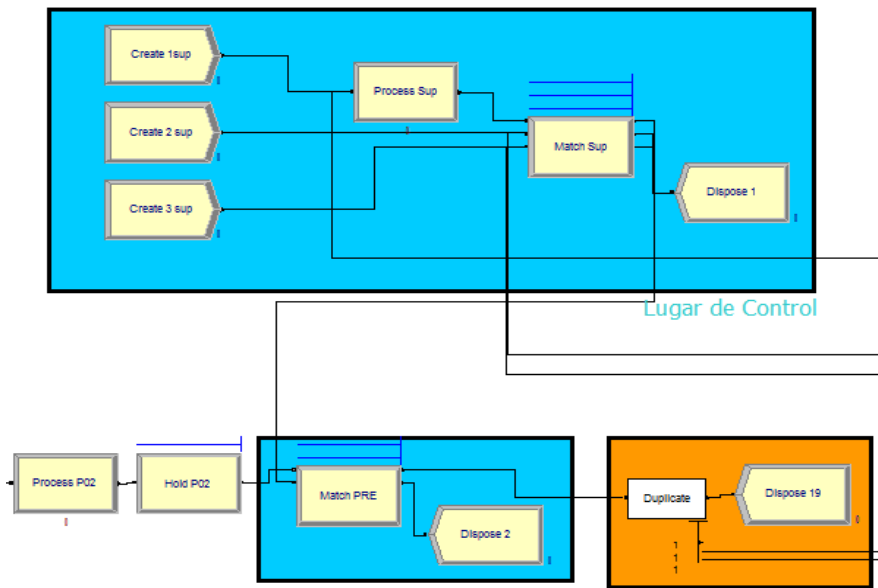


Figura 4.12 Modificador de transición PRE ampliado.

Modificador POST

Un modificador de transición POST, modela el efecto de un arco desde una transición hacia un lugar de supervisión de la RdP, es decir, devolverá una marca cuando sea necesario, esto se logra a través, de un bloque *Duplicate* conectado posteriormente a la transición que se desea modificar, sobre la entrada y salida superior se conectara la ruta original de la RdP, es decir, que la entidad continua por su camino normal, y se debe agregar una rama por la cual el duplicado de la entidad será devuelto al lugar de supervisión.

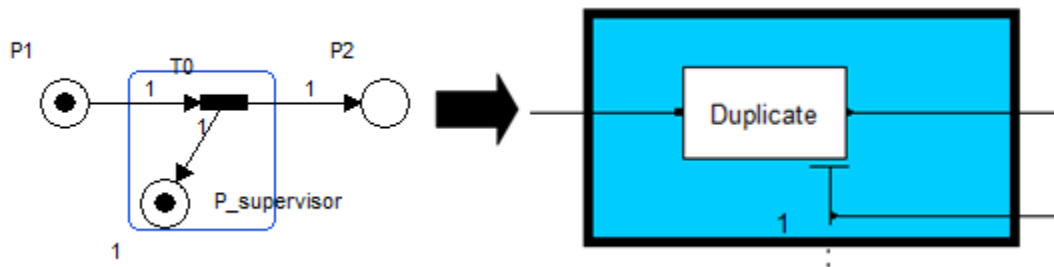


Figura 4.13 Modificador de transición POST.

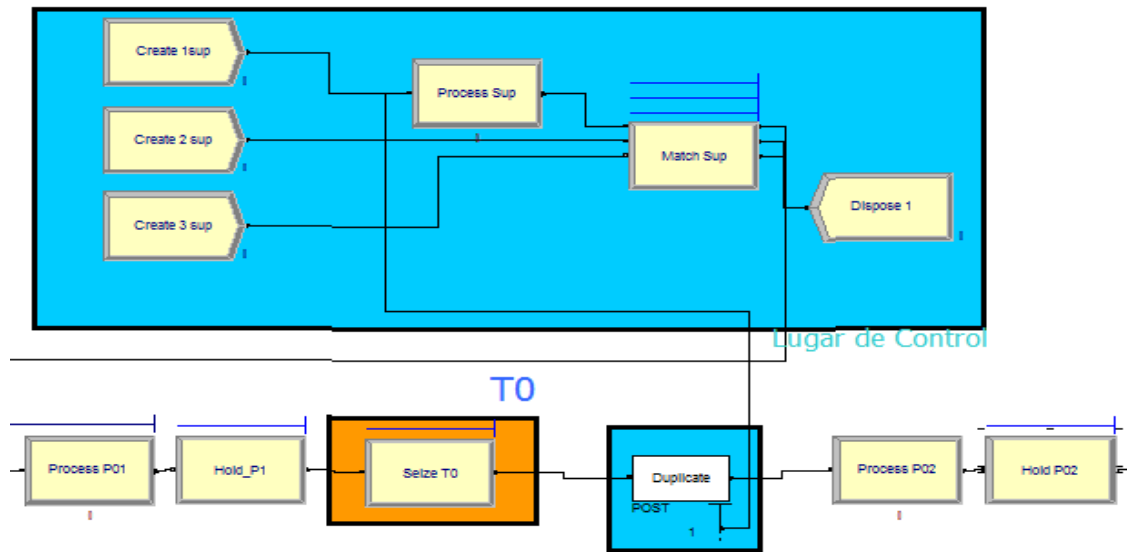


Figura 4.14 Modificador de Transición POST ampliado.

Así se finaliza con la método de implementación de supervisor basado en RdP sobre el software ARENA, para ver su aplicación más detallada, en el Anexo B se muestra como configurar de los bloques usados para el diseño e implementación de los casos de estudio.

5. DISEÑO E IMPLEMENTACIÓN DE LOS CASOS DE ESTUDIO

En este capítulo se muestran algunos prototipos de modelos en RdP cuyo comportamiento dinámico puede ser ejecutado para la simulación en ARENA mediante el método propuesto. Para mejorar la comprensión del modelado y de la utilización de la herramienta computacional ARENA, se presentan los modelos de manera ejemplificada, basados en casos académicos que puedan representar procesos de manufactura.

Los modelos se simulan en PIPE y ARENA, con el objetivo de mostrar la similitud entre los resultados de esta herramienta y un paquete comercial.

5.1. CASO DE ESTUDIO 1: ROBOTS Y MESA GIRATORIA

Este caso de estudio corresponde a un caso típico de secuencia de eventos discretos, es un ejemplo sencillo para iniciar con la aplicación del método de implementación de RdP

sobre ARENA y en la parte de implementación del supervisor permite incluir dos tipos de lugares de control: uno por invariantes de lugar y otro que contiene una restricción que incluye el vector de disparo.

5.1.1. Modelo conceptual

Este caso de estudio es un ejemplo de producción de tipo académico tomado de [32]: consiste en una celda de producción cuyo equipamiento lo conforman dos robots con instrumentos manipuladores, y una mesa giratoria. Los insumos son las piezas A, B y C. El primer robot toma las piezas A y B y los ensambla produciendo el subproducto AB, lo coloca en la mesa, la cual gira y en ese momento el segundo robot toma el subproducto AB y lo ensambla con la pieza C para producir ABC, luego lo coloca en la cinta transportadora. La mesa giratoria es un recurso compartido y mientras un robot está trabajando sobre ella no debe girar, requiriéndose un mecanismo de coordinación al igual que un mecanismo que no permita que circulen insumos de más en el proceso.

Un esquema simplificado muestra el proceso en la figura 1, no se muestran detalles sobre cómo los robots percibirán la presencia o ausencia de piezas en su entorno, ni quien se las proporciona.

La principal característica del sistema es que los recursos son compartidos (la mesa es un recurso utilizado por los dos brazos), y los brazos bajo ciertas condiciones en la secuencia de producción, operan a la vez.

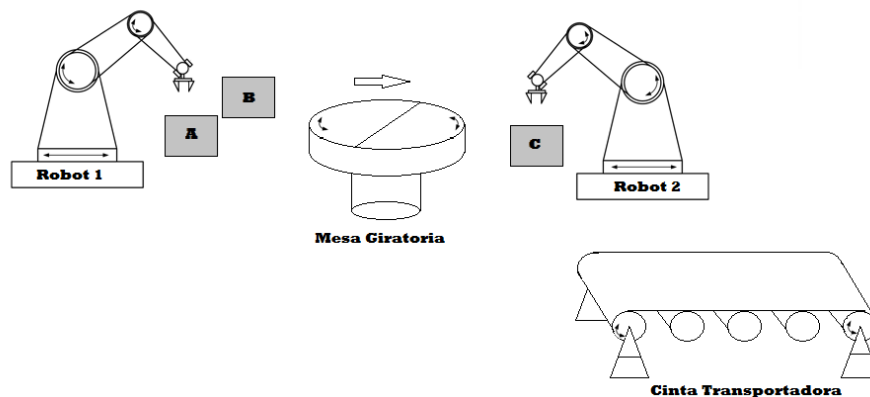


Figura 5.1 Esquema del proceso de fabricación.

Una descripción más detallada del ejercicio es:

- El brazo 1 toma una pieza A y la coloca sobre su lado de la mesa.

- El brazo 1 toma una pieza B, la coloca sobre su lado de la mesa y ensambla las piezas, obteniendo el subproducto AB.
- La mesa gira para que el subproducto quede del lado del brazo 2.
- El brazo 2 toma una pieza C, la coloca sobre su lado de la mesa y ensambla para obtener el producto terminado ABC.
- El brazo 2 toma el producto ABC y lo lleva a una cinta transportadora.

Este proceso tiene a su vez las siguientes restricciones para su funcionamiento:

- Ambos brazos solo pueden tomar un elemento a la vez.
- El brazo 1 solo coloca el elemento A cuando su lado de la mesa esta vacio.
- El brazo 1 solo coloca el elemento B, luego de colocar el elemento A.
- La mesa solo gira cuando el robot 1 ensambla AB, y el lado del brazo 2 esta vacio.
- El brazo 2 solo coloca el elemento C cuando la pieza AB esta de su lado de la mesa.

5.1.2. Modelo en RdP

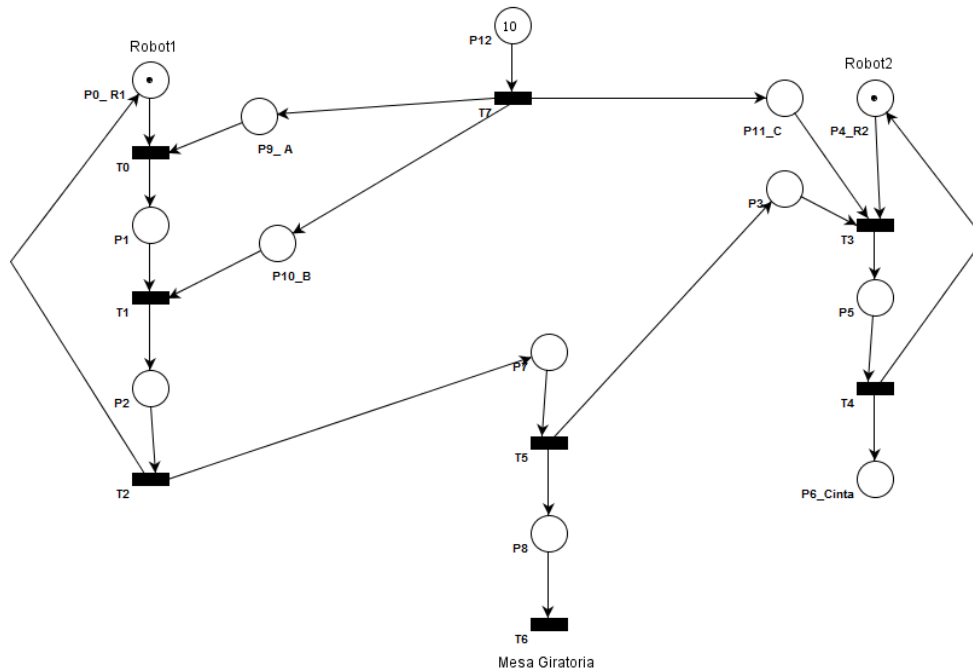


Figura 5.2 RdP del modelo de manufactura caso 1 sin supervisor en Pipe.

Tabla 5.1 Descripción de lugares y transiciones para el Robot1.

Nombre	Tipo	Descripción
P12	Lugar	Entrada de piezas, cantidad de unidades de producto ABC a producir (equivalente a cantidad de unidades de piezas A, B y C) Estos valores son iguales.
T7	Transición	Enviar insumos A, B y C a sus correspondientes posiciones para que los tomen los robots
P0	Lugar	Robot 1 libre para manipular insumo A
T0	Transición	El robot 1 toma pieza A y lo coloca sobre la mesa
P1	Lugar	El robot 1 ha colocado pieza A sobre la mesa
T1	Transición	El robot 1 toma pieza B y lo coloca sobre la mesa
P2	Lugar	El robot 1 ha colocado pieza B sobre la mesa
T2	Transición	El robot 1 ensambló las piezas A y B, obteniendo AB
P9	Lugar	Pieza A en posición para ser tomado por el robot 1
P10	Lugar	Pieza B en posición para ser tomado por el robot 1

Tabla 5.2 Descripción de lugares y transiciones para el Robot2

Nombre	Tipo	Descripción
P4	Lugar	Robot 2 libre para manipular insumo A
P11	Lugar	Pieza C en posición para ser tomado por el robot 2
P3	Lugar	Subproducto AB en posición para ser tomado por el robot 2
T3	Transición	El robot 2 toma la pieza C y el subproducto AB y los ensambla
P5	Lugar	El robot 2 ha ensamblado el producto ABC
T4	Transición	El robot 2 lleva el producto ABC a la cinta transportadora
P6	Lugar	El robot 2 ha colocado producto ABC en la cinta transportadora

Tabla 5.3 Descripción de lugares y transiciones relacionadas con la mesa giratoria.

Nombre	Tipo	Descripción
P7	Lugar	Subproducto AB esta sobre lado izquierdo de la mesa
T5	Transición	Gira la mesa de izquierda a derecha, para llevar subproducto AB al robot 2
P8	Lugar	Subproducto ABC esta sobre lado derecho de la mesa
T6	Transición	Gira la mesa de derecha a izquierda, para retornar a estado inicial

5.1.3. Implementación del modelo RdP en ARENA

Una vez modelado el sistema bajo el formalismo en RdP, se deben identificar los componentes que asemejan los elementos de la RdP con los módulos y bloques del entorno ARENA, el diseñador debe seguir el método planteado en el capítulo 3 para el modelado de RdP en ARENA.

La RdP cuenta con dos lugares asociados a recursos P0 y P4 que modelan la disponibilidad de los robots, Además de lugares de una entrada y una salida P1, P2, P3, P5, P6, P7, P8, P9, P10, P11 y P12. En cuanto a las transiciones se presenta una T7 de tipo una entrada y tres salidas; T0, T2 y T4 son de uso de recurso, T1 es de dos entradas y una salida, T3 es de dos entradas y una salida pero al mismo tiempo es de uso de recurso, T5 es de una entrada y dos salidas, T6 es una transición con una entrada y una salida que elimina una marca o entidad. Lo descrito aquí se muestra en la tabla 5.4.

Tabla 5.4 Clasificación de lugares y transiciones para caso 1.

Lugar	Tipo de lugar
P0 y P4	Con un arco de entrada y uno de salida
P1, P2, P3, P5, P6, P7, P8, P9, P10 y P11	Con múltiples arcos de entradas y salidas
P12	Un arco de salida y marcado inicial
Transición	Tipo de transición
T0, T2 y T4	Uso de recurso
T1	Dos entradas y una salida
T3	Dos entradas y una salida con uso de recurso
T5	Una entrada y dos salidas
T6	Una entrada y una salida que elimina una entidad
T7	Una entrada y tres salidas

El modelo en ARENA de los lugares P0 y P4 no se vera de manera física, basta con la creación de los recursos robot1 y robot2, fijando su capacidad en 1 para contar con una marca en el estado inicial en cada lugar.

Resource - Basic Process									
	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1	robot1	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
2	robot2	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>

Figura 5.3 Definición de los lugares P0 y P4.

El modelo en ARENA de los lugares P1, P2, P3, P5, P6, P7, P8, P9, P10 y P11 equivale a módulos *Process* y *Holds* conectados en serie y sus condiciones de avance serán configuradas luego de que el diagrama de flujo (lugares y transiciones) sea completado.

Para el marcado inicial del lugar P12 se debe agregar un bloque *Create* con un máximo de arribos de 1 y 10 entidades en dicho arribo.

Para la creación de las transiciones T0, T2 y T4 se utilizan bloques *Seize* y *Release* como se describe en la tabla 5.5.

Tabla 5.5 Modelo y configuración de las transiciones T0, T2 y T4 caso 1.

Transición	Módulo que lo modela	Configuración necesaria
T0	Seize T0	Resource,Robot1,1,
T2	Release T2	Resource,Robot1,1,
	Seize T2	Resource,mesa_IZQ,1,
T4	Release T4	Resource,Robot2,1,

La transición T1 cuenta con dos entradas, por tanto debe incluir un bloque *Match* de dos entradas y un bloque *Dispose* para eliminar una de las entidades que ingrese. La transición T3 cuenta con un bloque *Match* a la entrada, además de un bloque *Seize*, pues esta transición es una combinación de las descritas en la sección 3.3. Luego la transición T5 se modela con bloques *Duplicate*, *Dispose*, *Release* Y *Seize*. El bloque *Duplicate* debe ser configurado con dos ramas, los bloques *Release* y *Seize* son agregados para poder manipular el estado de los recursos llamados mesa_IZQ y mesa_DER, los cuales no hacen parte de la RdP pero fueron agregados con el fin de ver de manera grafica el estado del uso de la mesa giratoria. La transición T7 incluirá un bloque *Duplicate* configurado con 3 ramas. Finalmente para modelar la transición T6 se crea un bloque *Release* y un *Dispose* para liberar el estado de la mesa y eliminar la marca o entidad del modelo RdP.

Tabla 5.6 Modelo y configuración de las transiciones T1, T3, T5, T6 y T7 caso 1.

Transición	Módulo que lo modela	Configuración necesaria
T1	Match T1	Por defecto es 2 entradas
	Dispose T1	No necesaria
T3	Match T3	Por defecto es 2 entradas
	Seize T3	Resource,Robot2,1,
	Dispose T3	No necesaria
	Duplicate	Agregar 2 ramas
	Dispose T5	No necesaria

T5	Release T5	Resource,mesa_IZQ,1,
	Seize T5	Resource,mesa_DER,1,
T6	Release T6	Resource,mesa_DER,1,
	Dispose T6	No necesaria
T7	Duplicate	Agregar 3 ramas
	Dispose T7	No necesaria

Luego de agregar todos los bloques y conectarlos debidamente, es necesario incluir las condiciones de avance de las entidades usando los bloques *Hold*, para garantizar que las marcas se encuentren siempre sobre los lugares y su estadía en las transiciones sea solo pasajera. Primero es necesario definir todos los bloques *Hold* del tipo *scan for condition* luego se deben escribir las restricciones que se muestran en la tabla 5.7.

Tabla 5.7 Configuración de las restricciones de avance para el caso1.

Bloque	Condición	Descripción
Hold P1	NQ(Hold P10.Queue) >= 1	Espera una marca en P10
Hold P2	1	Ninguna restricción
Hold P3	NQ(Hold P11.Queue) >= 1 && STATE(robot2) == IDLE_RES	Espera una marca en P11 y que el robot2 este libre
Hold P5	1	Ninguna restricción
Hold P7	1	Ninguna restricción
Hold P8	1	Ninguna restricción
Hold P9	STATE(robot1) == IDLE_RES	Espera a que el robot1 este libre
Hold P10	NQ(Match_T1.Queue1) == 1	Espera una entidad en la cola 1 del Match_T1
Hold P11	NQ(Match T3.Queue1) == 1	Espera una entidad en la cola 1 del Match_T3

Además de incluir las transiciones y los lugares de la RdP, ARENA permite crear objetos de animación los cuales son asociados a los recursos o variables, en este caso, se agregaron 4 recursos los cuales representan los robots y los lados de la mesa, con el fin de poder observar gráficamente y de manera paralela al flujo de las marcas el estado de los recursos.

Ahora es posible simular el modelo de la RdP mediante el botón *Go* ubicado en la barra de herramientas o presionando la tecla F5 para observar como el flujo de las entidades se asemeja al flujo de las marcas en una RdP. Es conveniente definir un tiempo de simulación finito en *Run* → *Setup...* en la barra de menús, para este caso se configura una réplica de 26 horas.

Caso de Estudio 1: Robots y Mesa giratoria

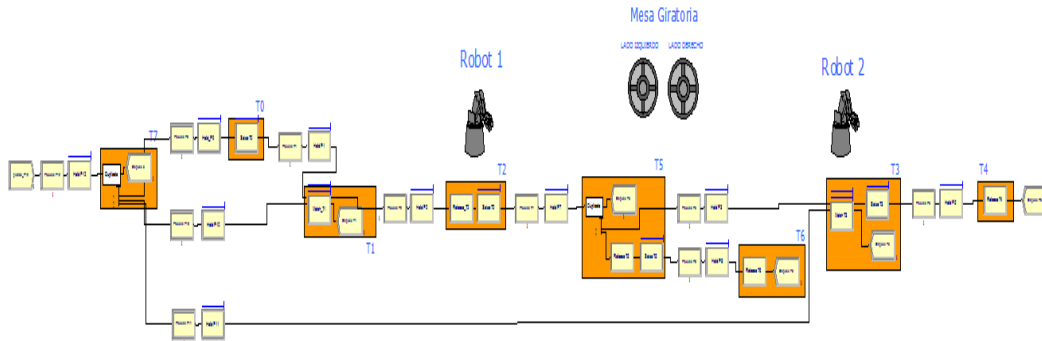


Figura 5.4 Modelo RdP en Arena del Caso de Estudio 1.

5.1.4. Supervisor basado en RdP para el caso de estudio 2

Siguiendo la metodología descrita en la sección 4.1. Para el diseño de supervisores, es necesario iniciar con:

CALCULAR LA MATRIZ DE INCIDENCIA

Para facilitarnos los cálculos manuales con las matrices que describen el comportamiento dinámico de la RdP del caso de estudio, la herramienta de simulación PIPE permite hallar la matriz de incidencia previa y posterior y la combinación entre ellas ($C = C^+ - C^-$), a través, del módulo disponible llamado *Incidence & Marking*, el cual, calcula la matriz de incidencia, obteniéndose:

	T0	T1	T2	T3	T4	T5	T6	T7
P0	1	0	0	0	0	0	0	0
P1	0	1	0	0	0	0	0	0
P10	0	1	0	0	0	0	0	0
P11	0	0	0	1	0	0	0	0
P12	0	0	0	0	0	0	0	1
P2	0	0	1	0	0	0	0	0
P3	0	0	0	1	0	0	0	0
P4	0	0	0	1	0	0	0	0
P5	0	0	0	0	1	0	0	0
P6	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	1	0	0
P8	0	0	0	0	0	0	1	0
P9	1	0	0	0	0	0	0	0

Combined incidence matrix /

Figura 5.5 Matriz de Incidencia del caso de estudio 1.

La herramienta PIPE ordena el archivo de lugares y transiciones que usa para este modelo en RdP del caso de estudio 1 de la siguiente manera:

$$\begin{aligned} \text{PCaso1} &= [P_0 P_1 P_{10} P_{11} P_{12} P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9] \\ \text{TCaso1} &= [T_0 T_1 T_{10} T_{11} T_{12} T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9] \end{aligned}$$

RESTRICCION DE CONTROL

- El material no debe avanzar hasta que se haya completado un ciclo en la RdP

$$\mu_{10} + \mu_{11} + \mu_9 \leq 1$$

$$L2 = [0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$$

RESTRICCION DE DISPARO

- El lado derecho de la mesa no debe desocuparse hasta que se halla transportado el producto ABC a la cinta.

$$\mu_3 + q_6 \leq 1$$

$$\mu_3 + \mu_{10} \leq 1$$

Para esta restricción de disparo se debe agregar un lugar y una transición adicional, el único propósito es introducir un nuevo lugar, el cual, graba el disparo de la transición; después de que el controlador ha sido calculado la planta debe ser transformada a su forma original. Para más detalles redimirse al ítem 4.1.2. Restricciones de Disparo.

$$L1 = [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1];$$

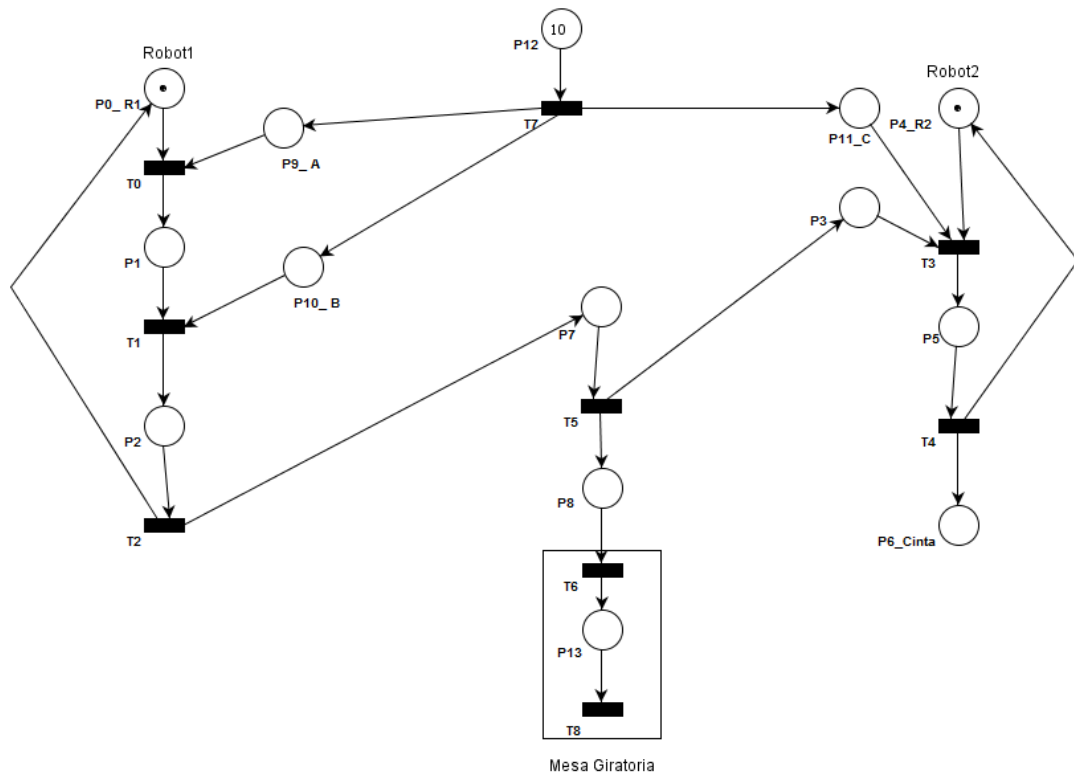


Figura 5.6 Red de Petri del modelo caso 1 ampliado.

Una vez obtenidas las correspondientes restricciones se introducen los datos a Matlab, se ejecuta el archivo que se encuentra detallado en el Anexo D, y se calcula el supervisor para el caso de estudio1, obteniéndose la siguiente RdP Supervisada, los lugares de control se enmarcan en el recuadro.

Tabla 5.8 Descripción de lugares y transiciones relacionadas con el supervisor caso 1.

Nombre	Tipo	Descripción
P13	Lugar	La mesa no se puede girar hasta que ABC sea llevado a Cinta
P14	Lugar	Controla el avance o la entrada de las piezas A, B y C completarse el ciclo

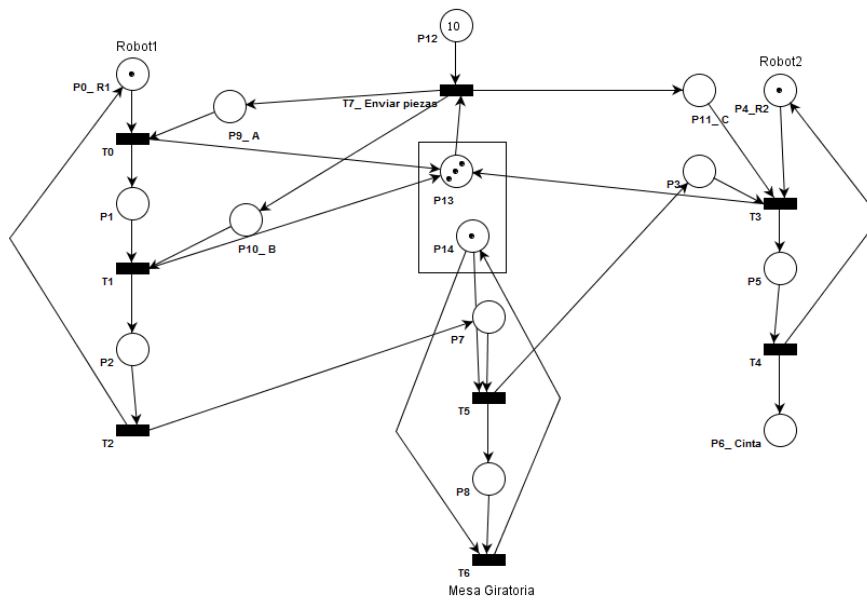


Figura 5.7 Supervisor basado en RdP del modelo de manufactura caso 1.

5.1.5. Implementación del Supervisor en RdP en ARENA para el caso de estudio 1

Se desea agregar el supervisor al caso de estudio en ARENA, con el fin de hacer cumplir las restricciones descritas; para esto iniciamos agregando los modificadores de transición PRE, los cuales serán necesarios antes de las transiciones T5, T6 y T7, luego agregamos los modificadores de transición POST después de T0, T1, T3 y T6 y realizamos las debidas conexiones. A continuación debemos agregar los lugares de supervisión, en este ejemplo tendremos dos de estos lugares uno de múltiples entradas y única salida con peso en el arco de salida y el otro de múltiples salidas y entradas. La configuración de sus bloques, para este caso en particular solo se deben hacer a las salidas del bloque *Decide* de N-way by Condition como se ve en la tabla 5.9.

Tabla 5.9 Configuración de las salidas del lugar de supervisión caso 1.

Condición	Descripción
$NQ(\text{Hold } P7.\text{Queue}) \geq 1$	Verifica que haya una entidad en P7, es decir, que haya una marca previa para poder disparar la transición T5
$NQ(\text{Hold } P8.\text{Queue}) \geq 1$	Verifica que haya una entidad en P8, es decir que haya una marca previa para poder disparar la transición T6

Para finalizar se deben agregar restricciones de avance en los bloques *Hold* previos a los modificadores de transición PRE agregados, es decir, agregar condiciones las cuales

darán prioridad a las decisiones tomadas por el supervisor, en este caso serán necesarias agregar condiciones en los bloques *Hold P7* y *Hold P8* como se muestra en la tabla 5.10.

Tabla 5.10 Configuración de las condiciones de avance caso 1.

Módulo	Condición	Descripción
Hold P7	$NQ(\text{Match } 9.\text{Queue}2) == 1$	Verifica que el supervisor este dando vía libre a la activación de T5
Hold P8	$NQ(\text{Match } 10.\text{Queue}2) == 1$	Verifica que el supervisor este dando vía libre a la activación de T6

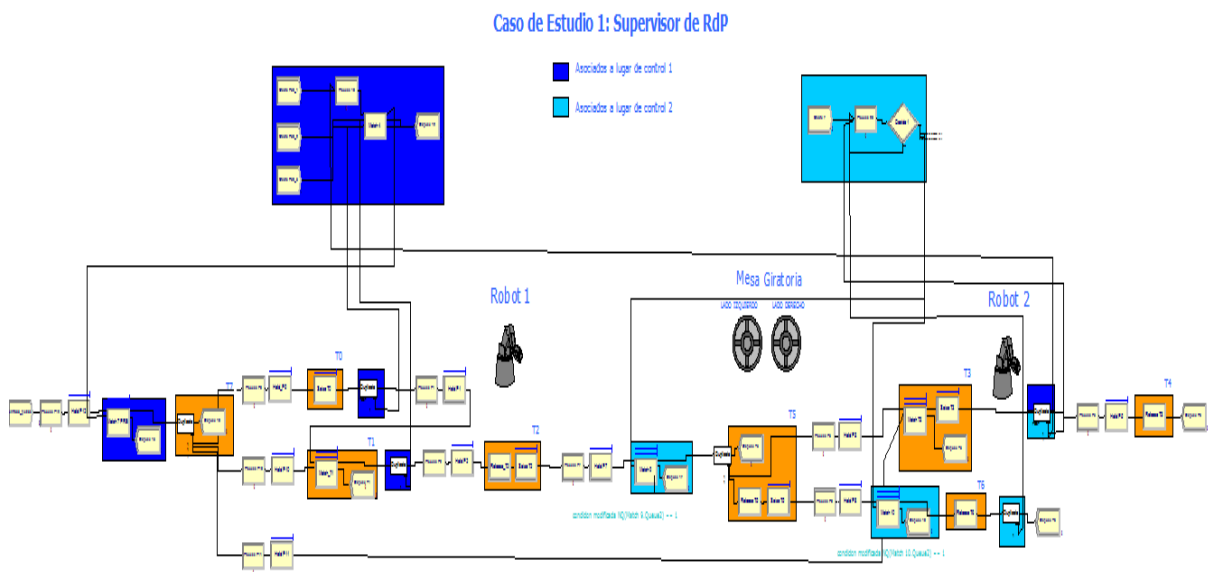


Figura 5.8 Modelo de RdP supervisada en ARENA para caso de estudio 1.

5.2. CASO DE ESTUDIO 2: TRES ROBOTS Y CUATRO MÁQUINAS

Este caso de estudio maneja un nivel de complejidad mayor, requiere de más recursos compartidos que proporcionan la flexibilidad del proceso, ha sido elegido porque incluye en su comportamiento dinámico un bloqueo, el cual, es el punto de partida para la elaboración del supervisor de RdP.

5.2.1. Modelo conceptual

Considere el Sistema de Manufactura flexible que se muestra en la figura 5.9, tomada de [19]. Este sistema consta de tres robots R1, R2 y R3, cada uno puede contener un producto al tiempo y cuatro máquinas M1, M2, M3 y M4 las cuales pueden procesar dos productos a la vez. Adicionalmente hay tres buffers de carga I1, I2, e I3 y tres buffers de descarga O1, O2 y O3 para cargar y descargar el SMF, respectivamente. Hay tres tipos de materia prima, para procesar los productos P1, P2 y P3. Para estos tipos de productos en bruto, las rutas de producción se muestran en la figura.1b. De acuerdo con las rutas de producción, la primera ruta es la del producto P1, el cual se toma de I1, a continuación la toma R1, quien pone el producto en M1 o en M3. Después de ser procesado en M1 (o M3) este se mueve entonces de M1 (o M3) a M2 (o M4) por medio de R2. Finalmente, después de haber sido procesado por M2 (o M4) entonces se pasa de M2 (o M4) para O1 por medio del R3. La materia prima para el producto P2 es tomada de I2 y es puesta en M2 por R2, después de ser procesado por M2, el producto pasa de M2 a O2 por R2. La materia prima del producto P3 se toma de I3 y se pone en M4 por R3. Posteriormente de ser procesado por M4 se pasa entonces de M4 a M3 por R2. Finalmente, después de haber sido procesado por M3, se mueve del M3 a O3 por R1.

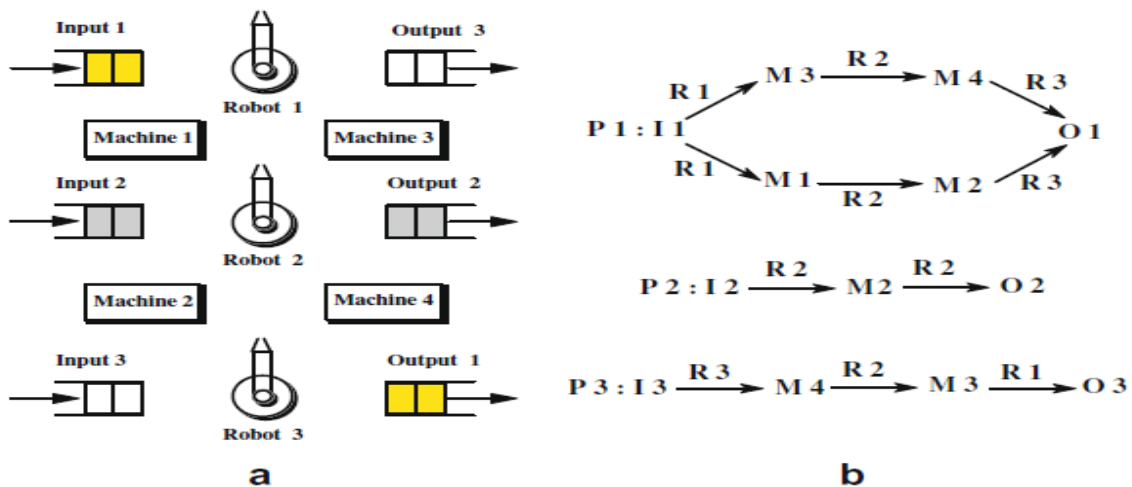


Figura 5.9 (a) La distribución de la FMS, y (b) Ruta de producción.

5.2.2. Modelo en RdP

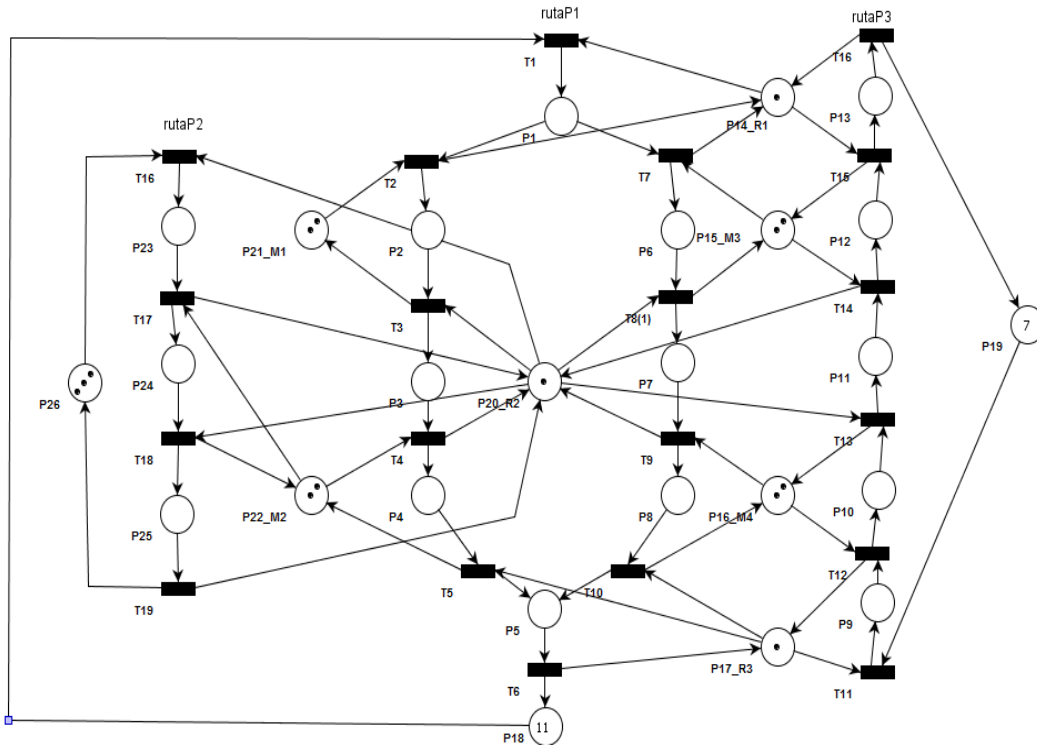


Figura 5.10 Red de Petri del modelo del caso 2 sin supervisor en Pipe.

Tabla 5.11 Descripción de lugares para la ruta P1

Nombre	Tipo	Descripción
P1	Lugar	El robot R1 carga las máquinas M1 o M3
P2	Lugar	La máquina M1 realiza operación sobre P1
P3	Lugar	El robot R2 descarga la máquina M1 y carga M2
P4	Lugar	La máquina M2 opera sobre P1
P5	Lugar	La máquina M4 realiza operación sobre P1
P6	Lugar	La máquina M3 realiza operación sobre P1
P7	Lugar	El robot R2 descarga la máquina M3 y carga M4
P8	Lugar	El robot R3 descarga la máquina M2 o M4

Tabla 5.12 Descripción de lugares para la ruta P2.

Nombre	Tipo	Descripción
P23	Lugar	El robot R2 carga la máquina M2
P24	Lugar	La máquina M2 realiza operación sobre P2
P25	Lugar	El robot R2 descarga M2

Tabla 5.13 Descripción de lugares para la ruta P3.

Nombre	Tipo	Descripción
P9	Lugar	El robot R3 carga la máquina M4
P10	Lugar	La máquina M4 realiza operación sobre P3
P11	Lugar	El robot R2 descarga la máquina M4 y carga M3
P12	Lugar	La máquina M3 realiza operación sobre P3
P13	Lugar	El robot R1 descarga la máquina M3

Tabla 5.14 Descripción de lugares Complementarios.

Nombre	Tipo	Descripción
P14	Lugar	El robot R1 está libre
P15	Lugar	La máquina M3 está libre
P16	Lugar	La máquina M4 está libre
P17	Lugar	El robot R3 está libre
P18	Lugar	Libre
P19	Lugar	Libre
P20	Lugar	El robot R2 está libre
P21	Lugar	La máquina M1 está libre
P22	Lugar	La máquina M2 está libre
P23	Lugar	Libre

Las marcas en los lugares $M_{P18}= 11$, $M_{P19}= 7$, $M_{P26}= 3$, representa el número de actividades simultáneas que puede tener lugar por tipo de pieza P1- P3, respectivamente.

A continuación se observa la RdP del caso 2 con una modificación al ejercicio original, teniendo en cuenta que al intentar modelar esta RdP en Arena se presentaron inconvenientes con respecto a las limitaciones presentadas en la versión estudiantil por la cantidad de módulos, bloques, elementos SIMAN que puede usar ARENA, razones por las cuales, el caso de estudio 2 se ha reducido, eliminando la ruta de procesamiento del Producto 2, obteniendo el sistema mostrado en la figura 5.11.

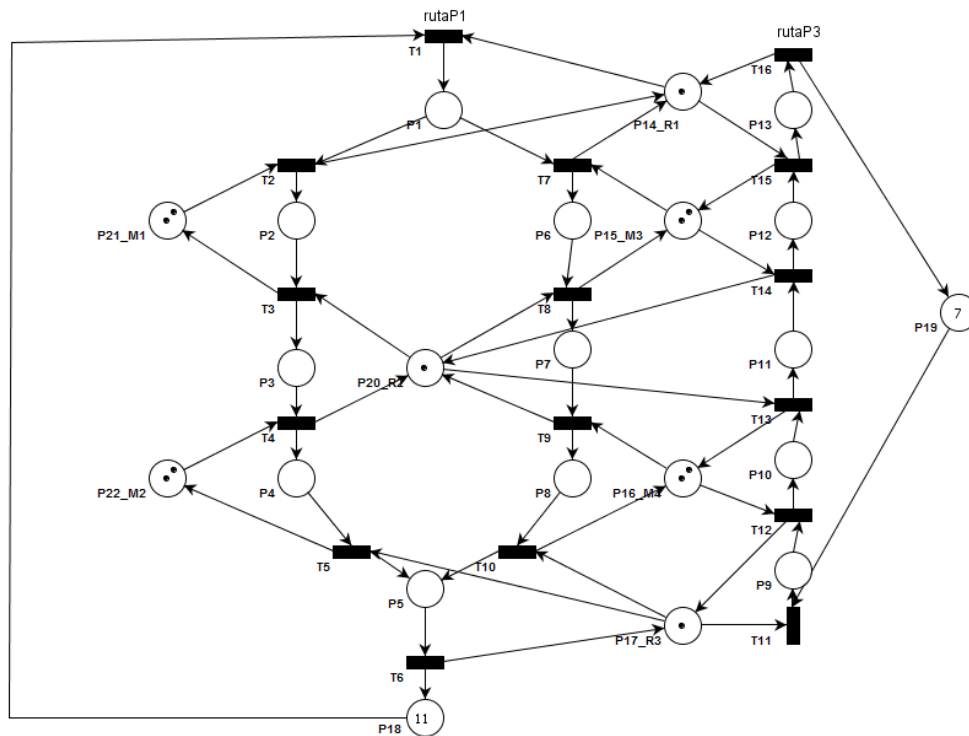


Figura 5.11 RdP del modelo del caso 2 disminuido en Pipe.

5.2.3. Implementación del modelo RdP en ARENA

Una vez modelado el sistema bajo el formalismo en RdP, se deben identificar los componentes que asemejan los elementos de la RdP con los módulos y bloques del entorno ARENA, el diseñador debe seguir el método planteado en el capítulo 3 para el modelado de RdP en ARENA.

La red de petri cuenta con siete lugares asociados a recursos P14, P15, P16, P17, P20, P21 Y P22. Los cuales serán representados por módulos *Resource* como se muestra en la figura 5.12.

Resource - Basic Process									
	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1	Robot1	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
2	Robot2	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
3	maquina1	Fixed Capacity	2	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
4	maquina2	Fixed Capacity	2	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
5	maquina3	Fixed Capacity	2	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
6	maquina4	Fixed Capacity	2	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
7	Robot3	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>

Figura 5.12 Definición de los lugares asociados a recursos en ARENA.

Además tiene un lugar con una entrada y múltiples salidas P1 y un lugar con dos entradas y una salida P5, que serán modelados con bloques *Process* y *Decide*. El resto de lugares son más básicos y su modelo será bloques *Process* y *Holds* conectados en serie

Tabla 5.15 Lugares con múltiples entradas y múltiples salidas caso 2.

Lugar	Módulos que lo modelan	Configuración sobre el módulo	Descripción
P1	Process P1	No necesaria	--
	Decide P1_1	2-way by Chance 50% y 50%	La marca puede salir por cualquiera de las transiciones con probabilidad de 50%-50%
	Decide P1_2	NR(máquina1) == 2	Verifica que la transición T2 se pueda disparar
	Decide P1_3	NR(máquina3) == 2	Verifica que la transición T7 se pueda disparar
P5	Process P5	No necesaria	--

Tabla 5.16 Lugares de una entrada y una salida caso 2.

Lugar	Módulos que lo modelan	Configuración sobre el módulo	Descripción
P2	Process P2	No necesaria	--
	Hold P2	STATE(Robot2) == IDLE_RES	Espera a que el robot 2 este libre
P3	Process P3	No necesaria	--
	Hold P3	NR(máquina2) < 2	Espera a que el máquina 2 este libre
P4	Process P4	No necesaria	--
	Hold P4	STATE(Robot3) == IDLE_RES	Espera a que el robot 3 este libre
P6	Process P6	No necesaria	--
	Hold P6	STATE(Robot2) == IDLE_RES	Espera a que el robot 2 este libre
P7	Process P7	No necesaria	--
	Hold P7	NR(máquina4) < 2	Espera a que el máquina 4 este libre
P8	Process P8	No necesaria	--
	Hold P8	STATE(Robot3) == IDLE_RES	Espera a que el robot 3 este libre
P9	Process P9	No necesaria	--
	Hold P9	NR(máquina4) < 2	Espera a que el máquina 4 este libre
P10	Process P10	No necesaria	--
	Hold P10	STATE(Robot2) == IDLE_RES	Espera a que el robot 2 este libre
P11	Process P11	No necesaria	--
	Hold P11	NR(máquina3) < 2	Espera a que el máquina 3 este libre
P12	Process P12	No necesaria	--
	Hold P12	STATE(Robot1) ==	Espera a que el robot 1 este

		IDLE_RES	libre
P13	Process P13	No necesaria	--
P18	Process P18	No necesaria	--
	Hold P18	STATE(Robot3) == IDLE_RES	Espera a que el robot 3 este libre
	Create P18	Max Arrivals = 1, Entities per Arrival = 7	Marcado inicial del lugar P18 es igual a 7
P19	Process P19	No necesaria	--
	Hold P19	STATE(Robot3) == IDLE_RES	Espera a que el robot 3 este libre
	Create P19	Max Arrivals = 1, Entities per Arrival = 11	Marcado inicial del lugar P11 es igual a 11

El modelo en ARENA de las transiciones estará dado por bloques *Sieze* y *Release*, ya que todas las transiciones estas asociadas al uso de un recurso.

Tabla 5.17 Modelo y configuración de transiciones en RdP en ARENA caso 2.

Transición	Módulos que lo modelan	Configuración los módulos	Descripción
T1	Sieze T1	Resource, Robot1,1	Captura el robot 1
T2	Sieze T2	Resource, Máquina1,1	Captura la máquina1
	Release T2	Resource, Robot1,1	Libera el robot1
T3	Sieze T3	Resource, Robot2,1	Captura el robot2
	Release T3	Resource, Máquina1,1	Libera la máquina1
T4	Sieze T4	Resource, Máquina2,1	Captura la máquina2
	Release T4	Resource, Robot2,1	Libera el robot2
T5	Sieze T5	Resource, Robot3,1	Captura el robot3
	Release T5	Resource, Máquina2,1	Libera la máquina2
T6	Release T6	Resource, Robot3,1	Libera el robot3
T7	Sieze T7	Resource, Máquina3,1	Captura la máquina3
	Release T7	Resource, Robot1,1	Libera el robot1
T8	Sieze T8	Resource, Robot2,1	Captura el robot2
	Release T8	Resource, Máquina3,1	Libera la máquina3
T9	Sieze T9	Resource, Máquina4,1	Captura la máquina4
	Release T9	Resource, Robot2,1	Libera el robot2
T10	Sieze T10	Resource, Robot3,1	Captura el robot3
	Release T10	Resource, Máquina4,1	Libera la máquina4
T11	Sieze T11	Resource, Robot3,1	Captura el robot3
T12	Sieze T12	Resource, Máquina4,1	Captura la máquina4
	Release T12	Resource, Robot3,1	Libera el robot3
T13	Sieze T13	Resource, Robot3,1	Captura el robot3
	Release T13	Resource, Máquina4,1	Libera la máquina4
T14	Sieze T14	Resource, Máquina3,1	Captura la máquina3

	Release T14	Resource, Robot2,1	Libera el robot2
T15	Sieze T15	Resource, Robot1,1	Captura el robot1
	Release T15	Resource, Máquina3,1	Libera la máquina3
T16	Release T16	Resource, Robot1,1	Libera el robot1

Para observar el estado de los recursos en tiempo de simulación se agregan indicadores para los Robots y las Máquinas. Para este caso se realiza una simulación de una réplica de 70 horas.

Caso de Estudio 2: 3 Robots y 4 Maquinas Rdp

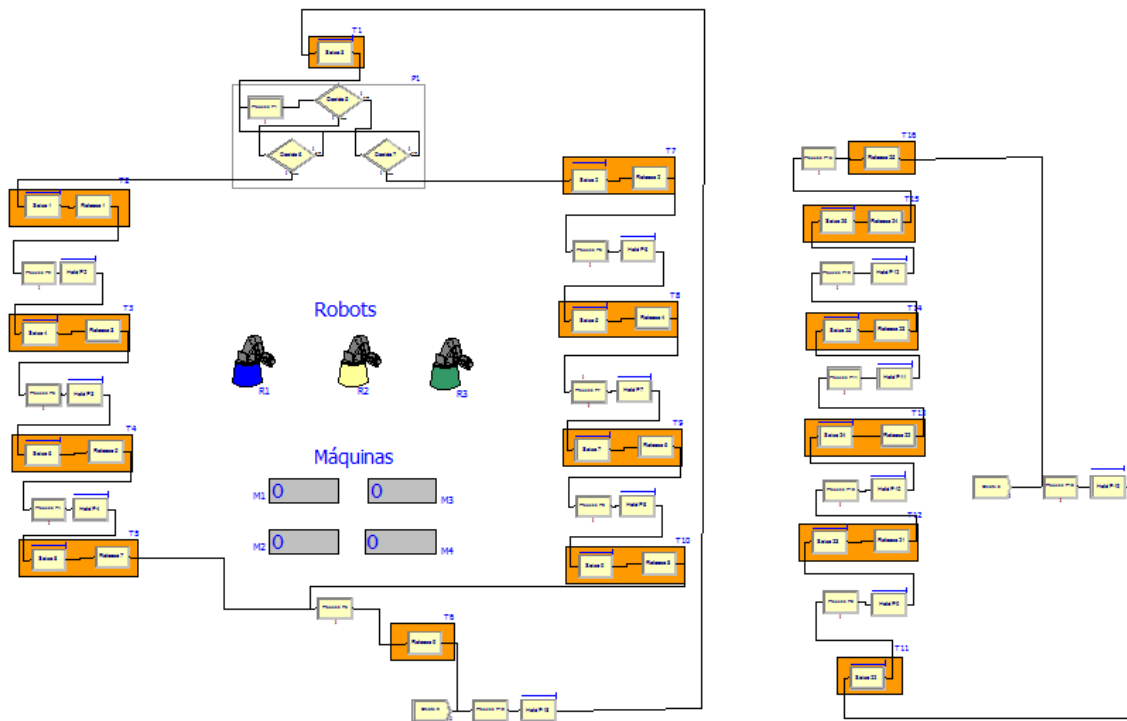


Figura 5.13 Modelo de Rdp en ARENA del caso de estudio 2.

5.2.4. Supervisor basado en Rdp para el caso de estudio 2

Siguiendo la metodología descrita en capítulo 4 para el diseño de supervisores, es necesario:

LA MATRIZ DE INCIDENCIA:

- Si la máquina M3 esta operando sobre el producto P3 no se debe tratar de cargar de nuevo por el R2 para procesar el mismo producto.

$$\mu_{11} + \mu_{12} \leq 1$$

$$L = [00110000000000000000]$$

Una vez descritas las restricciones, se procede al cálculo del supervisor como se muestra en el código de matlab en el Anexo D. Luego se procede a ubicar los lugares de control en la RdP original teniendo en cuenta el sentido de los arcos y el marcado de los lugares.

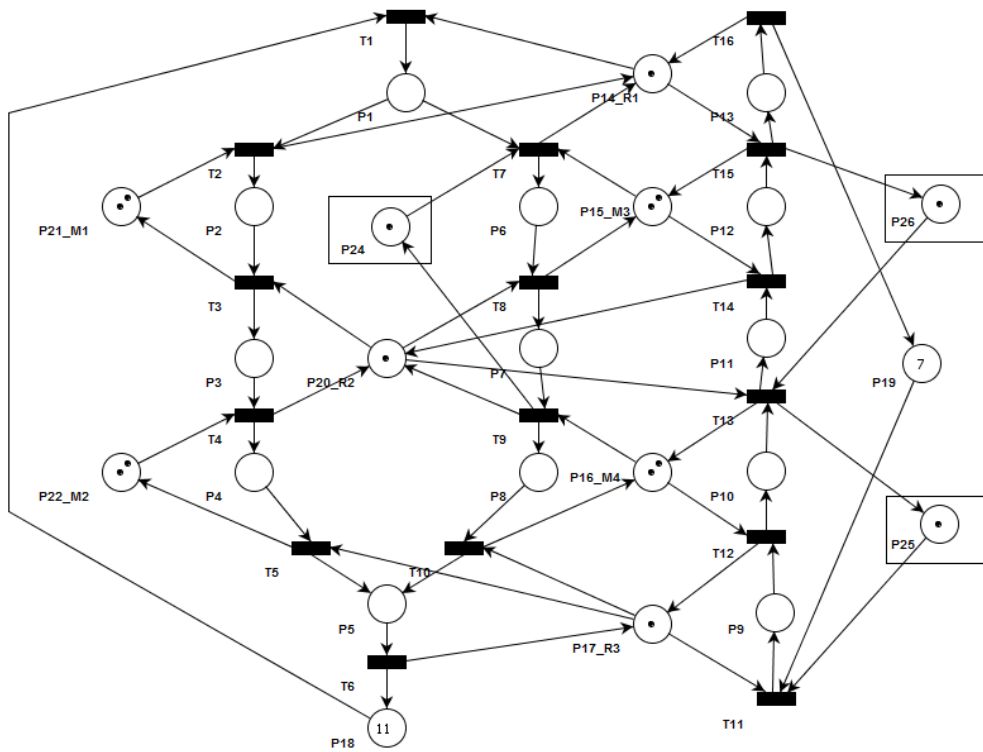


Figura 5.15 Supervisor del modelo RdP para el caso de estudio 2.

La agregación de los lugares de control, logran satisfactoriamente eliminar el bloqueo de la red, forzando la vivacidad de la RdP del presente sistema.

Tabla 5.18 Descripción de lugares y transiciones relacionadas con el supervisor caso 2.

Nombre	Tipo	Descripción
P24	Lugar de Control	Se controla que R2 no cargue a máquina M4 si M4 esta operando sobre dos productos.

P25	Lugar de Control	Si la máquina M4 esta operando sobre P3 en el lugar P ₁₀ no se debe tratar de cargar de nuevo la máquina M4 para dicho producto.
P26	Lugar de Control	Se controla que R2 no cargue máquina M3 si M3 esta operando sobre dos productos.

5.2.5. Implementación del Supervisor en RdP en ARENA para el caso de estudio 2

Para implementar el supervisor basado en RdP al caso de estudio 2 se inicia agregando los modificadores de transición PRE, necesarios antes de las transiciones T7, T11 y T13, luego agregamos los modificadores de transición POST después de T9, T13 y T15 y realizamos las debidas conexiones. A continuación debemos agregar los lugares de supervisión en este caso tendremos tres, todos de única entrada y única salida con marcado inicial de 1. Las configuraciones requeridas son solo para los marcados iniciales.

Tabla 5.19 Configuración de los lugares de supervisión caso 2.

Lugar Sup	Módulos que lo modelan	Configuración necesaria	Descripción
P24	Create	Max Arrivals = 1, Entities per Arrival = 1	Crea el marcado inicial
	Process	No necesaria	--
P25	Create	Max Arrivals = 1, Entities per Arrival = 1	Crea el marcado inicial
	Process	No necesaria	--
P26	Create	Max Arrivals = 1, Entities per Arrival = 1	Crea el marcado inicial
	Process	No necesaria	--

Finalmente es necesario agregar las restricciones de avance en los bloques *Hold* Y *Decide* previos a los modificadores de transición PRE, las nuevas restricciones se muestran en la tabla 5.20.

Tabla 5.20 Configuración de las restricciones de avance para incluir el supervisor caso 2.

Módulo	Condición	Descripción
Decide P1_3	NR(máquina3)<2 && NQ(Match 1.Queue2) == 1	Además de verificar el estado del uso de la máquina 3, revisa que el supervisor este dando vía libre a la activación de la transición T7.
Hold P10	STATE(Robot2) == IDLE_RES && NQ(Match	Además de verificar el estado del uso del Robot 2, revisa que el supervisor este dando

	$3.Queue2 == 1$	vía libre a la activación de la transición T13.
Hold P26	$STATE(Robot3) ==$ $IDLE_RES \ \&\& \ NQ(Match$ $2.Queue2) == 1$	Además de verificar el estado del uso de la Robot 3, revisa que el supervisor este dando vía libre a la activación de la transición T11.

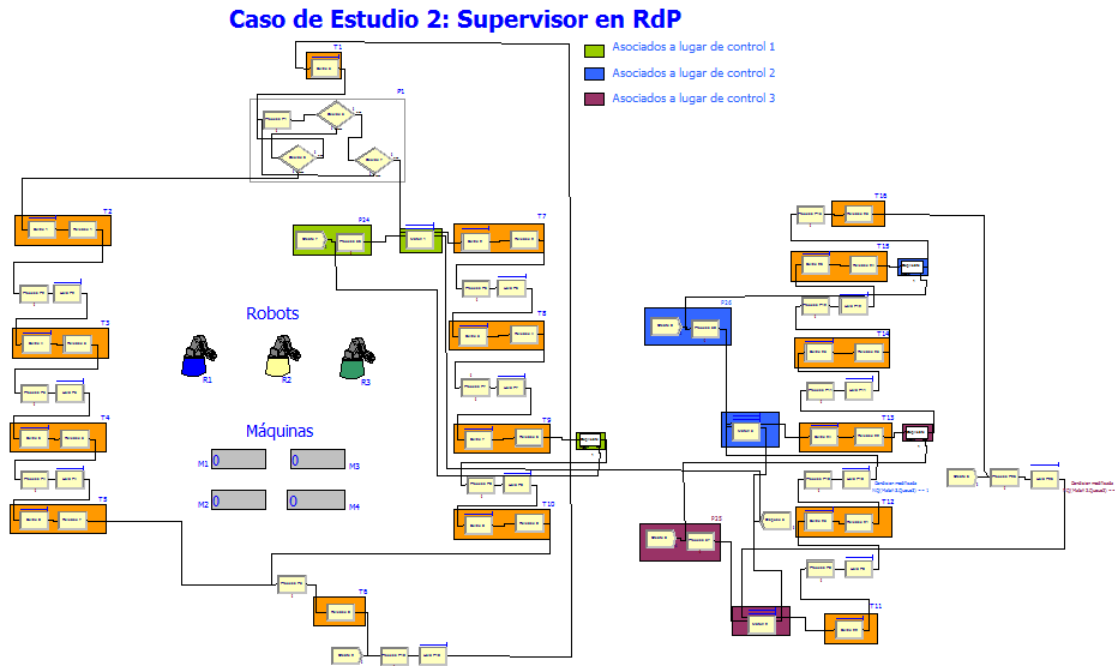


Figura 5.16 Modelo de supervisor de RdP del caso de estudio 2.

5.3. CASO DE ESTUDIO 3: AGVS

Finalmente se plantea este caso de estudio que incluye tiempo en el proceso, permite chequear que la metodología elegida para el diseño de un supervisor basado en RdP funciona en este tipo de red de petri temporizada, y se puede aprovechar las características de tiempo del software ARENA.

5.3.1. Modelo conceptual

El SMF consta de un AGV⁶, tres máquinas M1, M2 y M3 y dos productos A y B. Este modelo es descrito [35]. El producto A es procesado por las máquinas M1 y M3 y el producto B se procesa por la Máquina M2. Son necesarias tres operaciones de transporte

⁶ AGV: (Automated Guided Vehicle)Vehículo Auto Guiado

por parte del AGV. En primer lugar, la materia prima es entregada al inicio de cada máquina, posteriormente el producto manufacturado es tomado de cada máquina por el AGV. Finalmente, el tercer requerimiento de transporte es solo para el producto A entre las máquinas M1 y M2. Todos los productos son transportados sobre pallets.

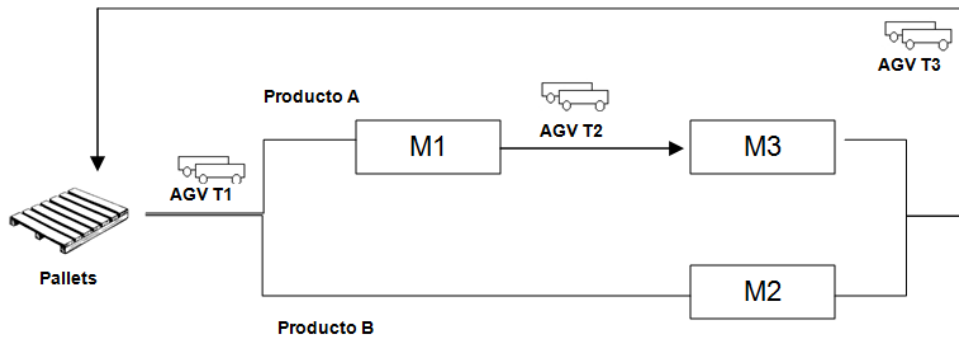


Figura 5.17 SMF con AGV.

El paso de una parte a través de las máquinas se describe así: Asumimos que la materia prima debe hacer el producto B, así que B espera por la máquina M2 hasta que esté disponible, al terminarse la operación por dicha máquina, el AGV toma el producto de la máquina M2. La manufactura del producto A es análoga, pero con dos máquinas y otra operación de transporte entre ellas.

Este proceso tiene a su vez las siguientes restricciones para su funcionamiento:

- El AGV puede transportar solo un pallet a la vez. El pallet espera a que el AGV este libre, a continuación, cuando el pallet esta sobre el AGV ocurre un retardo de transporte. El resto de operaciones de transporte se modelan similarmente.
- El proceso de manufactura puede elegir si hacer producto A o producto B con una probabilidad del 50% cada una.
- La máquina M2 puede hacer los mismos procesos de manufactura de la máquina M3 y viceversa. Así que el producto A y el producto B pueden cada uno ser manufacturados por M2 o M3. Sin embargo, la máquina M3 es óptima para la manufactura del producto A y la máquina 2 es óptima para el producto B, así que el producto A tiene un tiempo de manufactura en M2 más largo que en M3, el producto B viceversa. Por tanto, la parte A debe ser transportada a la máquina M3 si esta libre, y solo a la máquina M2 si la M3 está ocupada y la M2 está libre.

5.3.2. Modelo en RdP

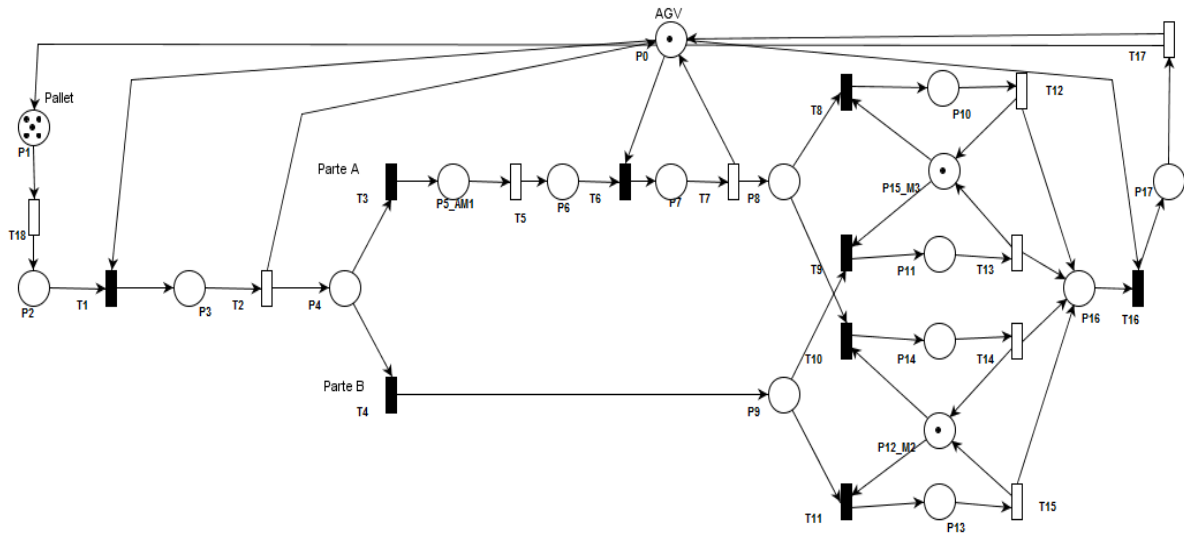


Figura 5.18 RdP del modelo del caso 2 sin supervisor en Pipe.

Tabla 5.21 Descripción de lugares caso 3

Nombre	Tipo	Descripción
P0	Lugar	AGV libre
P1	Lugar	El pallet esta libre
P2	Lugar	El pallet ha sido cargado
P3	Lugar	El AGV realiza operación de transporte 1
P4	Lugar	Lugar de elección de producto a fabricar
P5	Lugar	Producto A en la máquina M1
P6	Lugar	Producto A descargado de M1
P7	Lugar	El AGV realiza operación de transporte 2
P8	Lugar	Producto A espera a ser maquinado por M3 o M2
P9	Lugar	Producto B espera a ser maquinado por M2 o M3
P10	Lugar	Producto A cargado en máquina M3
P11	Lugar	Producto A cargado en máquina M2
P12	Lugar	Máquina M2 Libre
P13	Lugar	Producto B cargado en máquina M2
P14	Lugar	Producto B cargado en máquina M3
P15	Lugar	Máquina M3 Libre
P16	Lugar	Salida Productos de máquinas M2 y M3
P17	Lugar	El AGV realiza operación de transporte 3

5.3.3. Implementación del modelo RdP en ARENA

Una vez modelado el sistema bajo el formalismo en RdP, se deben identificar los componentes que asemejan los elementos de la RdP con los módulos y bloques del entorno ARENA, el diseñador debe seguir el método planteado en el capítulo 3 para el modelado de RdP en ARENA.

La RdP cuenta con tres lugares asociados a recursos P0, P12 y P15. Los cuales serán representados por bloques *Resource* como se muestra en la figura 5.19. Además se agrega un recurso “máquina 1” el cual no está incluido en el modelo de RdP propuesto en [35], pero se considera necesario para realizar un correcto modelado en ARENA.

Resource - Basic Process									
	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1	AGV	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
2	maquina3	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
3	maquina2	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>
4	maquina1	Fixed Capacity	1	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>

Figura 5.19 Lugares asociados a recursos para el caso 3.

Se cuenta con tres lugares con una entrada y múltiples salidas P4, P8 y P9; y un lugar con múltiples entradas y una salida P16. Estos serán modelados con bloques *Process* y *Decide*. El resto de lugares se modelan a través de bloques *Process* y *Holds* conectados en serie.

Tabla 5.22 Configuración de lugares de múltiples entradas y múltiples salidas caso 3.

Lugar Sup	Módulos que lo modelan	Configuración sobre el módulo	Descripción
P4	Process P4	No necesaria	--
	Decide P4_1	2-way by Chance - 50% y 50%	Se puede disparar T4 o T3
	Decide P4_2	STATE(máquina1) ==IDLE_RES	Verifica que la máquina 1 este libre para disparar T3
P8	Process P8	No necesaria	--
	Decide P8_1	2-way by Chance - 50% y 50%	Se puede disparar T8 o T10
	Decide P8_2	STATE(máquina3) ==IDLE_RES	Verifica que la máquina 3 este libre para disparar T8
	Decide P8_2	STATE(máquina2) == IDLE_RES	Verifica que la máquina 2 este libre para disparar T10
P9	Process P9	No necesaria	--
	Decide P9_1	2-way by Chance - 50% y 50%	Se puede disparar T9 o T11
	Decide P9_2	STATE(máquina3) ==IDLE_RES	Verifica que la máquina 3 este libre para disparar T9
	Decide P9_2	STATE(máquina2) == IDLE_RES	Verifica que la máquina 2 este libre para disparar T11

Tabla 5.23 Configuración de lugares de una entrada y una salida caso 3.

Lugar Sup	Módulos que lo modelan	Configuración sobre el módulo	Descripción
P1	Create P1	Max Arrivals = 1, Entities per Arrival = 5	Crea 5 marcas iniciales en P1
	Process P1	No necesaria	--
P2	Process P2	No necesaria	--
	Hold P2	STATE(AGV) == IDLE_RES	Espera a que el AGV este libre para disparar T1
P3	Process P3	No necesaria	--
P5	Process P5	No necesaria	--
P6	Process P6	No necesaria	--
	Hold P6	STATE(AGV) == IDLE_RES	Espera a que el AGV este libre para disparar T6
P7	Process P7	No necesaria	--
P10	Process P10	No necesaria	--
P11	Process P11	No necesaria	--
P13	Process P13	No necesaria	--
P14	Process P14	No necesaria	--
P16	Process P16	No necesaria	--
	Hold P16	STATE(AGV) == IDLE_RES	Espera a que el AGV este libre para disparar T16
P17	Process P17	No necesaria	--

En este caso el modelo RdP incluye el uso de transiciones temporizadas T2, T5, T7, T12, T13, T14, T15, T17 Y T18. Las cuales son modeladas dependiendo de las entradas y salidas y deben incluir bloques *Delay* para configurar el retardo que estas generan. Casi todas estas transiciones liberan un recurso por tanto deben incluir bloques *Release*.

Tabla 5.24 Modelo y configuración de las transiciones temporizadas del caso 3.

Transición	Módulos que lo modelan	Configuración sobre el módulo	Descripción
T2	Delay T2	Delay Time: 20 Units: hours	Retardo de 20 horas
	Release T2	Resource, AGV,1	Libera el AGV
T5	Delay T5	Delay Time: 20 Units: hours	Retardo de 20 horas
	Release T5	Resource, máquina2,1	Libera la máquina 2
T7	Delay T7	Delay Time: 20 Units: hours	Retardo de 20 horas
	Release T7	Resource, AGV,1	Libera el AGV
T12	Delay T12	Delay Time: 10 Units: hours	Retardo de 10 horas
	Release T12	Resource, máquina3,1	Libera la máquina 3
T13	Delay T13	Delay Time: 30 Units: hours	Retardo de 30 horas
	Release T13	Resource, máquina3,1	Libera la máquina 3

T14	Delay T14	Delay Time: 15 Units: hours	Retardo de 15 horas
	Release T14	Resource, máquina2,1	Libera la máquina 2
T15	Delay T15	Delay Time: 20 Units: hours	Retardo de 20 horas
	Release T15	Resource, máquina2,1	Libera la máquina 2
T17	Delay T15	Delay Time: 1 Units: hours	Retardo de 1 horas
	Release T15	Resource, AGV,1	Libera el AGV
T18	Delay T18	Delay Time: 4 Units: hours	Retardo de 4 horas

Las transiciones restantes T1, T3, T4, T6, T8, T9, T10, T11 Y T16 tienen como función capturar los recursos (máquinas o AGV) por tanto son modeladas por bloques *Seize* y deben ser configurados dependiendo de lo que se desee capturar al pasar una marca por esa transición.

Tabla 5.25 Modelo y configuración de las transiciones del caso 3.

Transición	Módulos que lo modelan	Configuración sobre el módulo	Descripción
T1	Seize T1	Resource, AGV,1	Captura el AGV
T3	Seize T3	Resource, maquina1,1	Capturar la máquina 1
T4	Hold T4	1 (no realiza ninguna acción)	Sin restricción
T6	Seize T6	Resource, AGV,1	Captura el AGV
T8	Seize T8	Resource, maquina3,1	Capturar la máquina 3
T9	Seize T9	Resource, maquina3,1	Capturar la máquina 3
T10	Seize T8	Resource, maquina2,1	Capturar la máquina 2
T11	Seize T8	Resource, maquina2,1	Capturar la máquina 2
T16	Seize T16	Resource, AGV,1	Captura el AGV

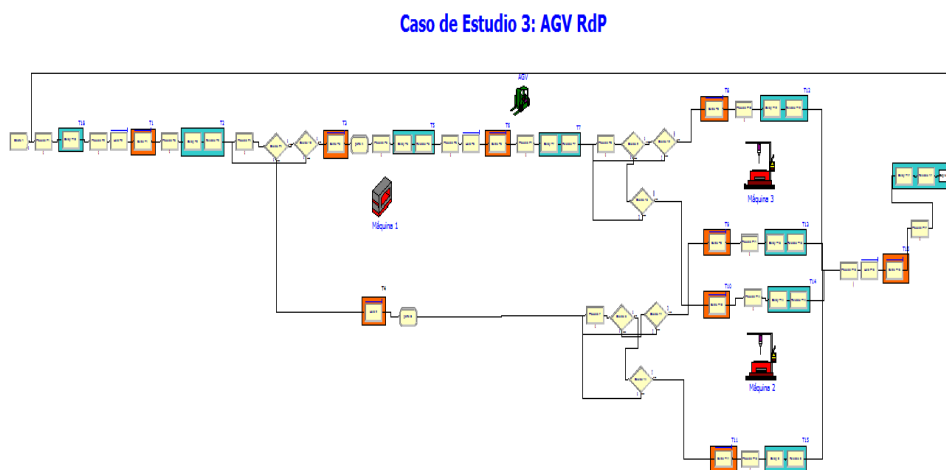


Figura 5.20 Modelo RdP en ARENA del caso de estudio 3.

Para observar el estado de los recursos en tiempo de simulación se agregan indicadores para el AGV y las Máquinas. Para este caso se realiza una simulación de una réplica de 500 horas.

5.3.4. Supervisor basado en RdP para el caso de estudio 3

Siguiendo la metodología descrita en capítulo 4. Para el diseño de supervisores, se inicia con:

LA MATRIZ DE INCIDENCIA

	T1	T10	T11	T12	T13	T14	T15	T16	T2	T17	T18	T5	T3	T4	T6	T7	T8	T9
P0_AGV	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
P1_emptyP	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P10_AinM3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P11_BinM3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
P12_M2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P13_BinM2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P14_AinM2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
P15_M3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
P16_outM2M3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
P17_AGV3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
P2_loadedP	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3_AGV1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
P5_AinM1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P6_AoutM1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P7_AGV2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P9	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Combined incidence matrix /

Figura 5.21 Matriz de Incidencia del caso de estudio 3.

La herramienta Pipe ordena el archivo de lugares y transiciones que usa para este modelo en RdP del caso de estudio 1 de la siguiente manera:

$$PCaso3 = [P_0 P_1 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16} P_{17} P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9]$$

$$TCaso3 = [T_1 T_{10} T_{11} T_{12} T_{13} T_{14} T_{15} T_{16} T_2 T_{17} T_{18} T_5 T_3 T_4 T_6 T_7 T_8 T_9]$$

RESTRICCIÓN DE CONTROL

Tabla 5.26 Descripción de lugares y transiciones relacionadas con el supervisor caso 3.

Nombre	Tipo	Descripción
P18	Lugar de Control	Cuando la máquina M3 este libre debe procurar ocuparse con el producto A y evitar ocuparse con B
P19	Lugar de Control	Cuando la máquina M2 este libre debe ocuparse con el producto B y evitar ocuparse con A

3.4.1. Implementación del Supervisor en RdP en ARENA para el caso de estudio 3

Con el propósito de hacer cumplir las restricciones descritas, se agrega el supervisor a la Rdp, para esto, iniciamos añadiendo los modificadores de transición PRE, necesarios antes de las transiciones T9, T10, T12 y T13, posteriormente se agrega los modificadores de transición POST después de T8, T11, T13 y T14 y se realizan las debidas conexiones. A continuación se incluyen los lugares de supervisión, en este caso se tienen dos de múltiples entradas y múltiples salidas con marcado inicial 0. Las configuraciones necesarias en los bloques *Decide* se deben al estado de los recursos, para que el supervisor no entre en conflicto con el estado de un recurso.

Tabla 5.27 Configuración de las restricciones de avance para incluir el supervisor caso 3.

Lugar Sup	Módulos que lo representan	Configuración necesaria	Descripción
P18	Process P18	No necesaria	--
	Decide P18	STATE(maquina3) ==BUSY_RES && NQ(Match 1.Queue1) == 0 && STATE(maquina2) == IDLE_RES	Verifica las condiciones para que se pueda disparar la transición T10
P19	Process P18	No necesaria	--
	Decide P18	STATE(maquina2) ==BUSY_RES && NQ(Match 4.Queue1) == 0 && STATE(maquina3) == IDLE_RES	Verifica las condiciones para que se pueda disparar la transición T19

Finalmente, se puede simular y observar como el supervisor fuerza al modelo a cumplir las restricciones impuestas, el tiempo de simulación para este caso es de una réplica de 1000 horas.

Caso de Estudio 3: Supervisor RdP AGV

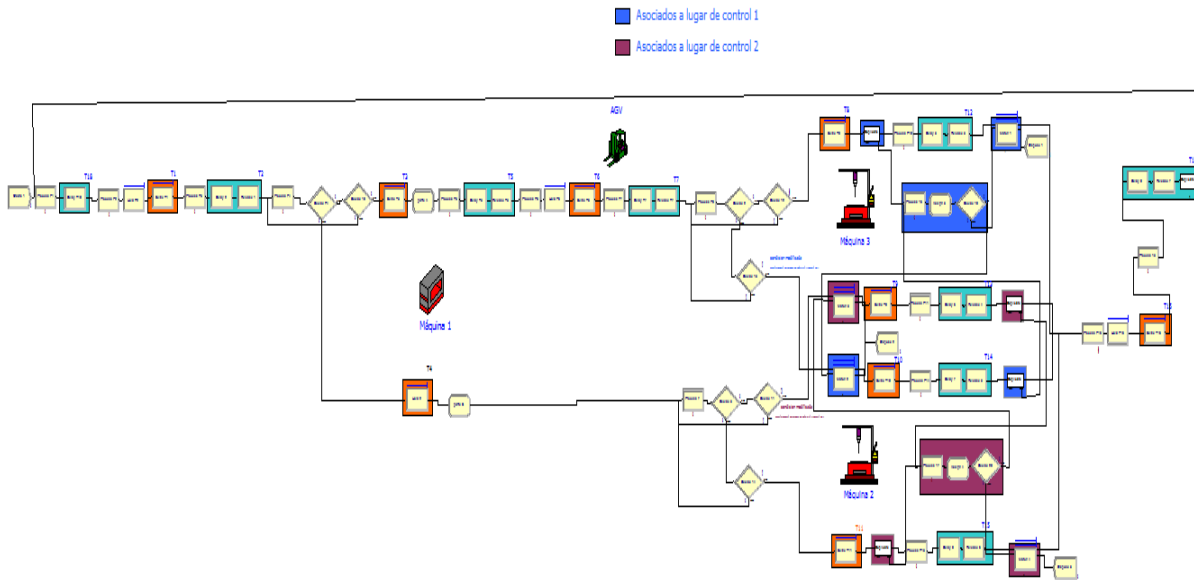


Figura 5.23 Modelo de RdP Supervisada para el caso de estudio 3.

6. VERIFICACIÓN Y VALIDACIÓN

En este capítulo se explican los procedimientos para la verificación de los modelos en RdP implementados en ARENA, a través de una comparación entre el flujo de las entidades en este software y el recorrido de las marcas en PIPE. Las pruebas que se han generado en este trabajo para la verificación, están realizadas con el propósito de constatar que la interpretación de la RdP en ARENA responde de la misma manera que lo hace en PIPE una herramienta propia para el formalismo RdP. Más adelante, se explica la validación del método de implementación del supervisor basado en RdP en ARENA, la cual, se realiza mediante una revisión paralela entre los marcados alcanzados por el modelo de la RdP supervisada en ARENA y el cumplimiento de las restricciones de control, a través de un código realizado en Matlab. Esto es explicado a través de pruebas de verificación y validación sobre los casos de estudio descritos en el Capítulo 5.

Una vez terminado el método para modelar RdP en el software ARENA, el siguiente paso a realizar es verificarla. Este proceso tiene como objetivo comprobar que se comporta tal cual fue concebida. Más allá de asegurarse que el modelo refleje el sistema del mundo real o hipotético, el proceso de verificación tiene que ver con la comparación del modelo, con su especificación realizada a partir del estudio del sistema o Modelo en RdP inicial. A simple vista, esto parece una tarea simple, pero en modelos complejos donde se producen varios procesos de manera simultánea, generalmente existen interacciones y resultados que están fuera de toda previsión y no siempre son fáciles de visualizar [17]. Es por eso que la animación de la simulación es el primer paso para poder ver el sistema y comenzar con una verificación conceptual del modelo a simple vista. A partir de esto, se recomienda generar pruebas sobre el modelo para ver sus resultados y corroborar que estos son coherentes respecto a lo esperado. Estas pruebas permiten determinar si se requiere correcciones en el método propuesto, o si es necesario eliminar algunos módulos, o condiciones que lleven a un comportamiento no deseado de la red.

En resumen la verificación es la tarea de asegurar que el modelo se comporta como se planeó. Verificar un modelo es convencerse de que el modelo funciona en la forma en que se pensó. Una vez lograda la verificación se procede a realizar la validación.

6.1. VERIFICACIÓN DEL MÉTODO DE IMPLEMENTACIÓN DE RDP EN ARENA

Como ya fue propuesto en el capítulo 3, el flujo de marcas en una RdP fue asemejado a un flujo de entidades en ARENA. La herramienta PIPE permite realizar la simulación de la evolución de las marcas de manera aleatoria o seleccionando la transición que se desea disparar, mientras que, El método implementado de RdP en ARENA, no permite que las transiciones puedan ser disparadas por selección, estas se dispararán de manera

aleatoria si cumplen con las condiciones necesarias para dicha acción (cantidad de marcas previas y recursos necesarios). Es por eso, que la verificación de los modelos en RdP se realiza a través, de la comparación del flujo de las marcas o entidades entre los bloques de ARENA y entre los lugares de la red de PIPE. Para esto se ejecutan varias simulaciones en ARENA teniendo como referencia el orden de los disparos de las transiciones, luego, se pasa a simular el mismo orden de disparos en el software PIPE y se revisa el marcado final obtenido en ambos modelos.

Para obtener el orden de los disparos de las transiciones en ARENA, es necesario, hacer uso del bloque *Record*, este módulo permite recoger estadísticas o datos en el modelo de simulación; en este caso los datos entregarán el orden en que se disparan las transiciones, para lograrlo, se agrega un bloque *Record* después del primer bloque de cada transición.

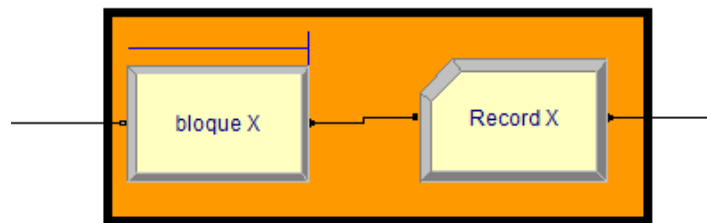


Figura 6.1 Transición modificada para almacenar orden disparo.

El bloque se debe configurar para que cada vez que pase una entidad grabe el número de la transición en un conjunto de datos (*Tally set*), la configuración de los bloques *Record* se realiza como se muestra en la figura 6.2. Del tipo expresión y en *Value* se debe configurar el número de la transición; en este caso se escribe un 7 equivalente la transición T7.

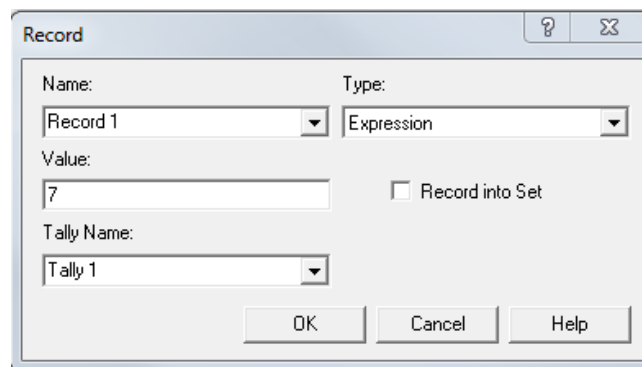


Figura 6.2 Configuración del bloque *Record* para almacenar el orden de disparos.

Para almacenar los datos en un archivo de texto, se debe crear un módulo *Statistic* ubicado en el *Panel Advance Process* en los *módulos de datos*; del tipo *Tally* y se debe configurar el nombre del archivo en el cual se desea guardar.

Statistic - Advanced Process					
	Name	Type	Tally Name	Tally Output File	Categories
1	Statistic 1	Tally	Tally 1	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos orden de marcado\orden_de_marcado1	0 rows

Figura 6.3 Definición del módulo *Statistic*.

Finalmente el archivo de texto será exportado a un archivo de extensión *.dax* haciendo uso del subprograma de ARENA *Output Analyzer* para poder ser leídos como se explica en el Anexo B, de esta manera se obtendrá el orden del disparo de las transiciones.

6.1.1. Comparación entre flujos de marcas y de entidades para el caso de estudio 1

Los datos del orden de los disparos se pueden obtener usando los archivos creados en ARENA que incluyen los bloques *Record*, para el caso de estudio 1 se le ha llamado a este archivo *Model2*, el cual se encuentra en la carpeta *sims arena\mesa_robot\datos orden de marcado*. Al ejecutar una simulación de una réplica de 11.5 horas, este envía el orden de los disparos y los almacena en un archivo de texto llamado *orden de marcado 1*, luego este será exportado a un archivo *texto_orden.dax* a través del *Output Analyzer*.

En el archivo de texto se encuentran almacenados 37 datos correspondientes al orden del disparo de las transiciones, con este orden, se procede a simular en PIPE el mismo orden de disparos obteniendo la siguiente RdP:

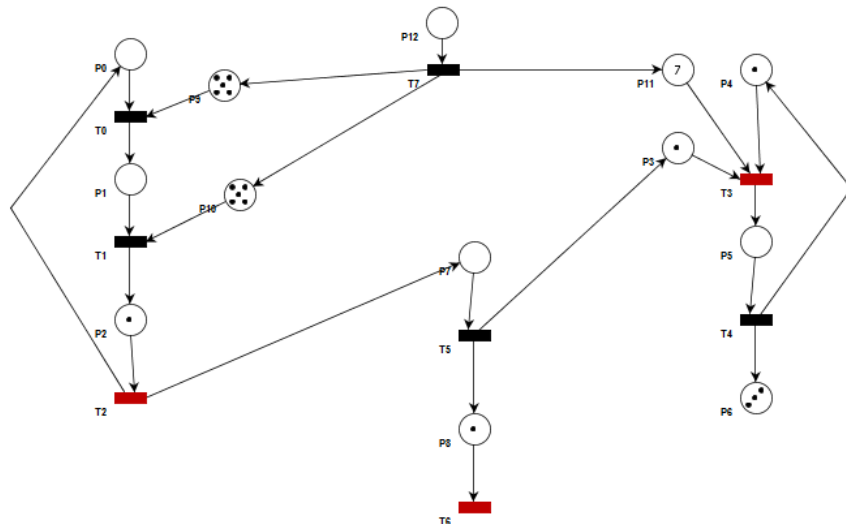


Figura 6.4 RdP luego del disparo de las transiciones en el orden correspondiente.

Por otra parte se observa el marcado final del modelo en ARENA, teniendo en cuenta las variables que representan el marcado de cada lugar, que al final de la simulación se pueden visualizar en la cantidad de entidades en los bloques, encontrando lo siguiente:

El estado de los robots es:

- robot 1 ocupado $\rightarrow P0 = 0$
- robot 2 libre $\rightarrow P4=1$

El marcado final obtenido en ARENA es:

$lugar \rightarrow$ $p0$ $p1$ $p10$ $p11$ $p12$ $p2$ $p3$ $p4$ $p5$ $p6$ $p7$ $p8$ $p9$
 $marcado \rightarrow$ 0 0 5 7 0 1 1 1 0 3 0 1 5

El marcado final obtenido en PIPE es:

		Marking												
		P0	P1	P10	P11	P12	P2	P3	P4	P5	P6	P7	P8	P9
Initial		1	0	0	0	10	0	0	1	0	0	0	0	0
Current		0	0	5	7	0	1	1	1	0	3	0	1	5

Figura 6.5 Marcado final obtenido en PIPE para el caso 1.

Esta prueba nos permite observar que el marcado final obtenido es el mismo en ambos software de simulación, el ensayo puede ser realizado las veces que se desee con diferentes longitudes en las replicas, condición que se configura en el Setup de la simulación (*Replication Length*), encontrando que los marcados finales son los mismos.

6.1.2. Comparación entre flujos de marcas y de entidades para el caso de estudio 2

El documento con el orden de disparos se pueden obtener usando el archivo *Model2* en la carpeta *sims arena\FMS\datos orden de marcado*. Al ejecutar una simulación de una réplica de 70 horas, este enviará el orden de los disparos y los almacenará en un archivo de texto llamado *orden de marcado 2*; luego éste se exporta a un archivo *texto_orden2.dax* usando el *Output Analyzer*.

En el archivo de texto se encuentran almacenados 35 datos correspondientes al orden del disparo de las transiciones, con este orden se procede a simular en PIPE la misma secuencia obteniendo la siguiente RdP:

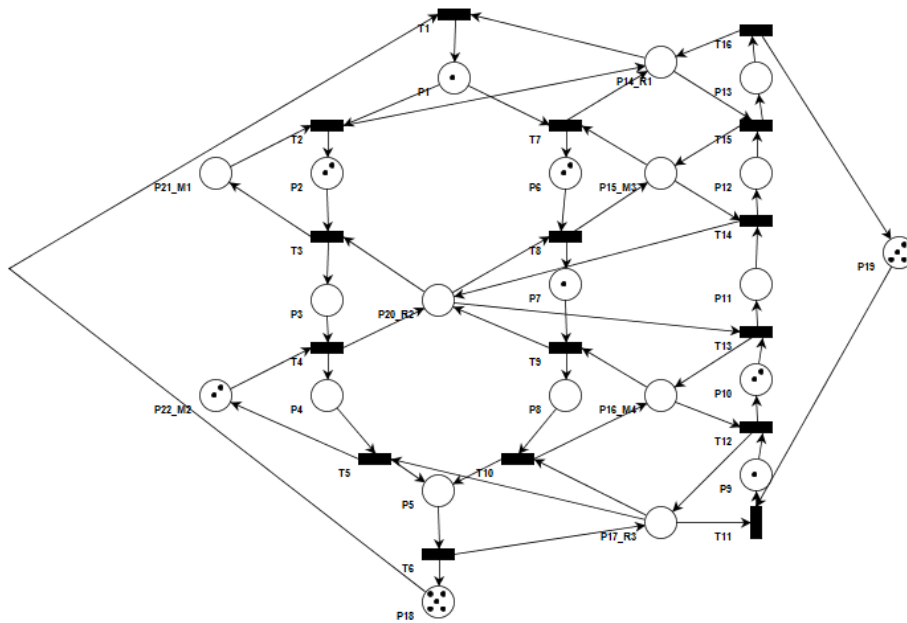


Figura 6.6 RdP luego del disparo del orden dado.

En ARENA se observa el marcado final, a través de las variables que representan el marcado de cada lugar, para el caso de estudio 2 se tiene:

El estado de los robots es:

- robot 1 ocupado $\rightarrow P14 = 0$
- robot 2 ocupado $\rightarrow P17 = 0$
- robot 3 ocupado $\rightarrow P20 = 0$

El estado de las máquinas es:

- máquina 1: 2 en uso $\rightarrow P21 = 0$
- máquina 2: 0 en uso $\rightarrow P22 = 2$
- máquina 3: 2 en uso $\rightarrow P15 = 0$
- máquina 4: 2 en uso $\rightarrow P16 = 0$

El marcado final obtenido en ARENA es:

$p1$	$p10$	$p11$	$p12$	$p13$	$p14$	$p15$	$p16$	$p17$	$p18$	$p19$	$p2$	$p20$	$p21$	$p22$	$p3$	$p4$	$p5$	$p6$	$p7$	$p8$	$p9$
1	2	0	0	0	0	0	0	0	5	4	2	0	0	2	0	0	0	2	1	0	1

El marcado final obtenido en PIPE es:

	P1	P10	P11	P12	P13	P14_R1	P15_M3	P16_M4	P17_R3	P18	P19	P2	P20_R2	P21_M1	P22_M2	P3	P4	P5	P6	P7	P8	P9
Initial	0	0	0	0	0	1	2	2	1	11	7	0	1	2	2	0	0	0	0	0	0	0
Current	1	2	0	0	0	0	0	0	0	5	4	2	0	0	2	0	0	0	2	1	0	1

Figura 6.7 Marcado final obtenido en Pipe para el caso 2.

Así verificamos que el marcado final obtenido en PIPE como en ARENA es el mismo, la prueba puede ser realizada las veces que se requiera con diferentes longitudes en las replicas, obteniendo los mismos resultados. Es importante mencionar que este caso de estudio presenta un bloqueo, el cual se puede corroborar en ambos software, en ARENA es posible identificar un bloqueo, por que se produce un error en la simulación, se detiene el tiempo de simulación y las entidades dejan de avanzar, lo que indica una vez más la efectividad de la verificación.

6.1.3. Comparación entre flujos de marcas y de entidades para el caso de estudio 3

En el archivo Model3 ubicado en la carpeta *sims arena\AGVs_tiempo\datos orden de marcado*, se encuentra el orden de disparos de las transiciones dadas para este ejemplo. La simulación se ejecuta con una réplica de 200 horas, la cual envía el orden de los disparos de las transiciones y los almacena en el archivo de texto llamado *orden de marcado 3*, finalmente este se exporta a un archivo *texto_orden3.dax* para ser visualizado a través del *Output Analyzer*.

En el archivo de texto se encuentran almacenados 17 datos correspondientes al orden del disparo de las transiciones, se procede simular en PIPE el mismo orden, obteniendo la siguiente RdP:

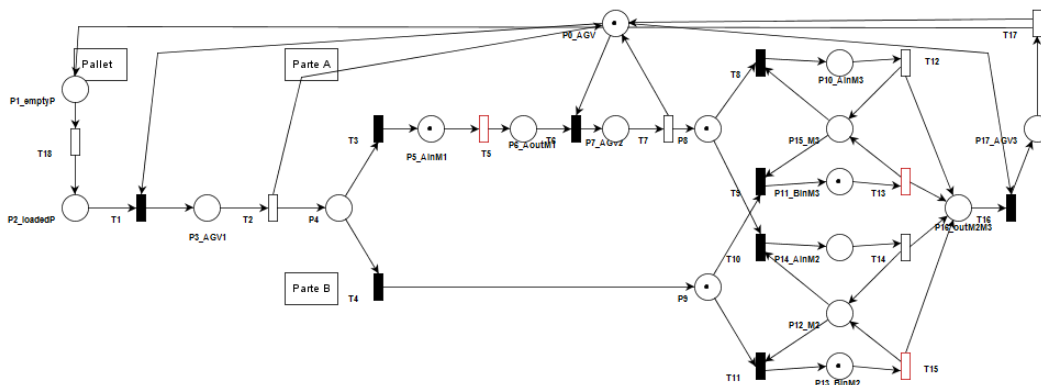


Figura 6.8 RdP luego del disparo del orden dado para caso 3.

El estado del AGV es:

- AGV libre $\rightarrow P0 = 1$

El estado de las máquinas es

- máquina 2 ocupada → P15 = 0
- máquina 3 ocupada → P12 = 0

El marcado final obtenido en ARENA es:

```

p0 p1 p10 p11 p12 p13 p14 p15 p16 p17 p2 p3 p4 p5 p6 p7 p8 p9
1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1

```

El marcado final obtenido en PIPE es:

	Marking																	
	P0_AGV	P1_emptyP	P10_AinM3	P11_BinM3	P12_M2	P13_BinM2	P14_AinM2	P15_M3	P16_outM2M3	P17_AGV3	P2_loadedP	P3_AGV1	P4	P5_AinM1	P6_AoutM1	P7_AGV2	P8	P9
Initial	1	5	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
Current	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	1

Figura 6.9 Marcado final obtenido en Pipe para el caso 3.

Se verifica por este procedimiento que el marcado final obtenido en PIPE como en ARENA es el mismo, lo que garantiza que el método utilizado para modelar y simular RdP en ARENA satisface las reglas de funcionamiento de un software diseñado para este oficio en este tipo de formalismo.

6.2. VALIDACIÓN DEL DESEMPEÑO DEL SUPERVISOR EN ARENA

Es necesario partir conociendo que validación es la tarea de asegurar que el modelo se comporta de la misma forma que el sistema, de acuerdo a [16], en este caso nuestro sistema es el método usado para implementar un supervisor en RdP en ARENA. Como se explico en el capítulo 4 el supervisor deben impedir que ciertos marcados no deseados sean alcanzados por la red, razón por la cual, es necesario validar que el método propuesto cumpla con las restricciones impuestas para cada caso de estudio.

Así, que se ha diseñado un experimento para obtener los resultados de la simulación, que mediante las herramientas estadísticas que posee ARENA, permita analizar los resultados y responder al objetivo inicial, es decir, que garantice la premisa del supervisor.

Para esto se debe conocer el marcado de cada lugar después del disparo de cualquier transición, lo cual se logra creando un ciclo de bloques *Record* que revisa el marcado de la red en un determinado tiempo, siendo este ciclo dos veces más rápido que cualquier otro bloque del modelo; este tomará suficientes muestras para revelar el marcado de la RdP modelada en ARENA. De esta manera se verifica que no se dé el marcado no deseado y se valida el método.

De manera gráfica el ciclo de bloques *Record* se muestra en la siguiente figura:

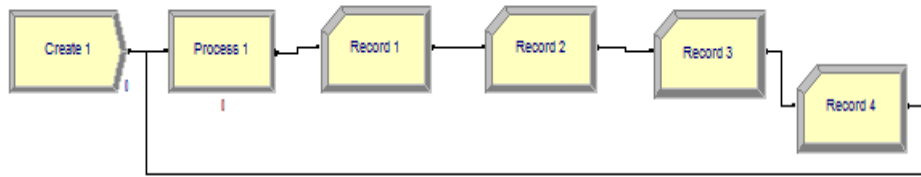


Figura 6.10 Ciclo de bloques para obtener el marcado de la RdP.

Este ciclo consiste de un bloque *Create*, el cual crea una sola entidad que circulará a través de él, además debe incluir un bloque *Process* al inicio, el cual se debe configurar con un tiempo constante, equivalente a la mitad del menor tiempo que exista en el modelo de simulación. Los módulos *Record* deben ser configurados para almacenar un dato correspondiente al marcado de cada lugar, es decir, que serán necesarios tantos bloques *Record* como lugares tenga la RdP y deben ser configurados del tipo expresión, donde se incluye la variable del marcado asociada a cada lugar, por otra parte, se deben seleccionar diferentes índices en la casilla *Set Index*, para que cada bloque *Record* ordene los marcados de cada lugar en diferentes filas.

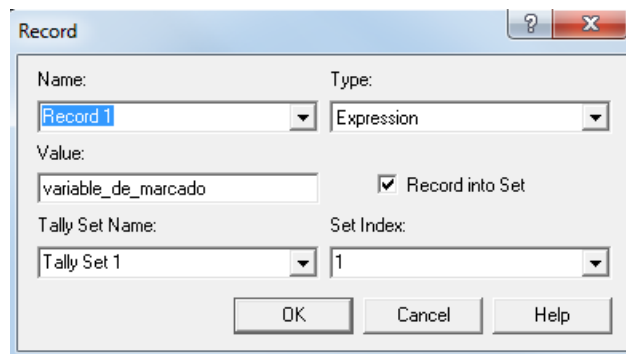


Figura 6.11 Configuración del bloque *Record* para obtener el marcado de un lugar.

Además es necesario crear un módulo de datos *Set* del tipo *Tally*, con una cantidad de filas igual la cantidad de lugares; sobre este *Set* serán escritos los marcados de los diferentes lugares, para grabarlos en un archivo de texto, es necesario definir módulos *Statistic* del tipo *Tally*; a cada uno, se debe asociar una de las filas del *Tally Set* definido anteriormente y se debe configurar el archivo sobre el que se desea escribir la información.

Statistic - Advanced Process				
	Name	Type	Tally Name	Tally Output File
1	Statistic 1	Tally	Tally 1	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\P12_datos
2	Statistic 2	Tally	Tally 2	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\P9_datos
3	Statistic 3	Tally	Tally 3	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\P10_datos

Se deben configurar cada una de las filas del Set definido
Se deben configurar los nombres de los archivos de texto donde se guardarán los datos

Figura 6.12 Configuración de los módulos *Statistic*.

Luego de ejecutar la simulación del modelo, los archivos configurados serán llenados con los marcados de cada lugar, pero no será posible leerlos, para esto se usa la herramienta *Output Analyzer* que exporta los archivos de texto a una extensión .dax la cual permite dar lectura, por cualquier editor de texto.

Finalmente es necesario verificar que las restricciones diseñadas para cada caso de estudio se cumplan, para esto se ha creado un código en Matlab, el cual, toma los datos de los diferentes archivos de simulación de los ejemplos y revisa que las restricciones se cumplen en los archivos que tienen el supervisor incorporado, indicando por medio de sentencias “si cumple o viola la restricción”.

6.2.1. Validación del desempeño del supervisor en RdP para el caso de estudio 1.

Para esta validación es necesario crear 13 bloques *Record*, cada uno por un lugar de la RdP, los cuales se configurarán para escribir una expresión equivalente a la variable de marcado de cada lugar sobre una fila de un *Tally Set*.

Record - Basic Process						
	Name	Type	Value	Record into Set	Tally Set Name	Set Index
1	Record 3	Expression	Process P12.WIP+NQ(Hold P12.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	1
2	Record 4	Expression	Process P9.WIP+NQ(Hold P9.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	2
3	Record 5	Expression	Process P10.WIP+NQ(Hold P10.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	3
4	Record 6	Expression	Process P11.WIP+NQ(Hold P11.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	4
5	Record 7	Expression	Process P1.WIP+NQ(Hold P1.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	5
6	Record 8	Expression	Process P2.WIP+NQ(Hold P2.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	6
7	Record 9	Expression	Process P7.WIP+NQ(Hold P7.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	7
8	Record 10	Expression	Process P3.WIP+NQ(Hold P3.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	8
9	Record 11	Expression	Process P8.WIP+NQ(Hold P8.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	9
10	Record 13	Expression	Process P5.WIP+NQ(Hold P5.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	10
11	Record 14	Expression	Dispose P6.NumberOut	<input checked="" type="checkbox"/>	Tally Set 1	11
12	Record 15	Expression	1-NR(robot1)	<input checked="" type="checkbox"/>	Tally Set 1	12
13	Record 16	Expression	1-NR(robot2)	<input checked="" type="checkbox"/>	Tally Set 1	13

Figura 6.13 Configuración de los bloques *Record* para el caso 1.

Adicionalmente, se debe agregar un bloque *Create* con un solo arribo y un bloque *Process*, el cual contiene un tiempo de proceso de 30 minutos equivalentes a la mitad del menor tiempo configurado en la Rdp. A continuación es necesario crear los 13 módulos *Statistic* y configurar sus archivos correspondientes, sobre los cuales se va a escribir el marcado de cada lugar.

Statistic - Advanced Process				
	Name	Type	Tally Name	Tally Output File
1	Statistic 1	Tally	Tally 1	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP12_datos
2	Statistic 2	Tally	Tally 2	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP9_datos
3	Statistic 3	Tally	Tally 3	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP10_datos
4	Statistic 4	Tally	Tally 4	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP11_datos
5	Statistic 5	Tally	Tally 5	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP1_datos
6	Statistic 6	Tally	Tally 6	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP2_datos
7	Statistic 7	Tally	Tally 7	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP7_datos
8	Statistic 8	Tally	Tally 8	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP3_datos
9	Statistic 9	Tally	Tally 9	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP8_datos
10	Statistic 10	Tally	Tally 10	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP5_datos
11	Statistic 11	Tally	Tally 11	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP6_datos
12	Statistic 12	Tally	Tally 12	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP0_datos
13	Statistic 13	Tally	Tally 13	C:\Users\Victor\Desktop\sims arena\mesa_robot\datos tallys_no_sup\IP4_datos

Figura 6.14 Configuración de los módulos *Statistic* para el caso 1.

Después de realizado esto, se ejecuta una simulación de 26 horas, que permite escribir el orden de los disparos de las transiciones sobre los archivos, posteriormente usando el *Output Analyzer* se exportan los archivos a una extensión *.dax* la cual puede ser leída en Matlab y analizada para ver si se cumplen las restricciones.

Este procedimiento se realiza en los modelos con supervisor y sin él, para poder validar que al incluir el supervisor, los marcados no deseados definidos por las restricciones son no alcanzables. Finalmente se ejecutan los archivos de Matlab: *analisis_mesa_no_sup.m* y *analisis_mesa_sup.m* para observar como los marcados prohibidos son alcanzables cuando no se cuenta con supervisor y como son no alcanzables cuando se analizan los datos obtenidos del ejemplo con supervisor, tal cual como se esperaba.

```

restriccion de control
a marcado Prohibido Encontrado En Fila 3
  marcado Prohibido Encontrado En Fila 4
  marcado Prohibido Encontrado En Fila 5
  marcado Prohibido Encontrado En Fila 6
  marcado Prohibido Encontrado En Fila 7
  marcado Prohibido Encontrado En Fila 8
  marcado Prohibido Encontrado En Fila 9
  marcado Prohibido Encontrado En Fila 10
b restriccion de control
  restriccion de disparo

```

Figura 6.15 Resultado luego de ejecutar (a) *analisis_mesa_no_sup.m* (b) *analisis_mesa_sup.m*.

Los resultados obtenidos indican que las muestras del marcado tomadas en el modelo en ARENA sin supervisor presentan violaciones al comportamiento deseado, indicando en qué fila de la matriz de los datos recolectados se encuentra la violación, mientras los datos obtenidos con el modelo supervisado no muestran ninguna violación.

6.2.2. Validación del desempeño del supervisor en RdP para el caso de estudio 2.

Para esta validación es necesario crear 22 bloques *Record*, cada uno por lugar de la RdP, pero debido a la limitación en número de módulos que se pueden usar en un modelo para la versión estudiantil del software ARENA, sólo se crearán 6 bloques *Record*, ya que con los bloques usados para el modelo de la RdP se ha alcanzado el tope.

Para obtener los marcados de los lugares incluidos en la definición de las restricciones del supervisor, se realiza entonces, la configuración para escribir una expresión equivalente a la variable del marcado de cada lugar sobre una fila de un *Tally Set*.

Record - Basic Process						
	Name	Type	Value	Record into Set	Tally Set Name	Set Index
1	Record 7	Expression	Process P6.WIP + NQ(Hold P6.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	1
2	Record 8	Expression	Process P7.WIP + NQ(Hold P7.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	2
3	Record 9	Expression	Process P9.WIP + NQ(Hold P9.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	3
4	Record 10	Expression	Process P10.WIP + NQ(Hold P10.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	4
5	Record 11	Expression	Process P11.WIP + NQ(Hold P11.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	5
6	Record 12	Expression	Process P12.WIP + NQ(Hold P12.Queue)	<input checked="" type="checkbox"/>	Tally Set 1	6

Figura 6.16 Configuración de los bloques *Record* para el caso de estudio 2.

Se adiciona el bloque *Create* con un solo arribo y el bloque *Process* el cual tendrá un tiempo de proceso de 30 minutos equivalentes a la mitad del menor tiempo configurado en la RdP. Luego es necesario crear los 6 módulos *Statistic* y configurar en el espacio de (Tally Output File) el nombre de los archivos sobre los cuales se va a almacenar el marcado de cada lugar.

Statistic - Advanced Process				
	Name	Type	Tally Name	Tally Output File
1	Statistic 1	Tally	Tally 1	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P6
2	Statistic 2	Tally	Tally 2	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P7
3	Statistic 3	Tally	Tally 3	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P9
4	Statistic 4	Tally	Tally 4	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P10
5	Statistic 5	Tally	Tally 5	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P11
6	Statistic 6	Tally	Tally 6	C:\Users\Victor\Desktop\sims arena\FMS\datos tallys_no_sup\datos_P12

Figura 6.17 Configuración de los módulos *Statistic* para el caso 2.

Después de realizado esto, se simula en un periodo de 70 horas, para obtener los datos de los disparos en los archivos, posteriormente usando el *Output Analyzer* se deben exportar los archivos a una extensión .dax, la cual puede ser leída y analizada en Matlab para ver si se cumplen las restricciones.

Finalmente se pueden ejecutar los archivos de Matlab *analisis_FMS_no_sup.m* y *analisis_FMS_sup.m* y observar que los marcados prohibidos son alcanzables cuando no se cuenta con supervisor de RdP y que como se esperaba, son no alcanzables cuando se chequean los datos obtenidos con supervisor es decir, el bloqueo se eliminó y el supervisor ha logrado su propósito.

```

restriccion 3
marcado Prohibido Encontrado En Fila 61
marcado Prohibido Encontrado En Fila 62
marcado Prohibido Encontrado En Fila 63
marcado Prohibido Encontrado En Fila 64
restriccion 2
marcado Prohibido Encontrado En Fila 5
marcado Prohibido Encontrado En Fila 6
marcado Prohibido Encontrado En Fila 7
marcado Prohibido Encontrado En Fila 9
restriccion 1
marcado Prohibido Encontrado En Fila 8
marcado Prohibido Encontrado En Fila 9
marcado Prohibido Encontrado En Fila 10
a marcado Prohibido Encontrado En Fila 11
b
restriccion 1
restriccion 2
restriccion 3
>>

```

Figura 6.18 Resultado luego de ejecutar (a) *analisis_FMS_no_sup.m* (b) *analisis_FMS_sup.m*.

6.2.3. Validación del desempeño del supervisor en RdP para el caso de estudio 3

Para esta validación es necesario crear 15 bloques *Record* para cada lugar de la RdP, pero como se explico anteriormente, la versión estudiantil de ARENA no permite introducir tantos bloques y elementos Siman en el modelo, así que solo se crearán 4, para obtener los marcados de los lugares incluidos en la definición de las restricciones y se configurarán de tipo expresión para almacenar la variable de marcado de cada lugar sobre una fila de un *Tally Set*.

Record - Basic Process						
	Name	Type	Value	Record into Set	Tally Set Name	Set Index
1	Record 1	Expression	Process P11.WIP	<input checked="" type="checkbox"/>	Tally Set 1	1
2	Record 2	Expression	1-NR(maquina2)	<input checked="" type="checkbox"/>	Tally Set 1	2
3	Record 3	Expression	Process P14.WIP	<input checked="" type="checkbox"/>	Tally Set 1	3
4	Record 4	Expression	1-NR(maquina3)	<input checked="" type="checkbox"/>	Tally Set 1	4

Figura 6.19 Configuración de los bloques *Record* para el caso de estudio 3.

Adicionalmente se agrega el bloque *Create* con un solo arribo y un bloque *Process* con un tiempo de proceso de 30 minutos equivalentes a la mitad del menor tiempo configurado en la RdP. Luego es necesario crear los 4 módulos *Statistic* y configurarlos con el mismo procedimiento de las pruebas anteriores.

Statistic - Advanced Process					
	Name	Type	Tally Name	Tally Output File	
1	Statistic 1	Tally	Tally 1	C:\Users\Victor\Desktop\sims arena\AGVs_tiempo\datos\tallys_no_sup\P11	
2	Statistic 2	Tally	Tally 2	C:\Users\Victor\Desktop\sims arena\AGVs_tiempo\datos\tallys_no_sup\P12	
3	Statistic 3	Tally	Tally 3	C:\Users\Victor\Desktop\sims arena\AGVs_tiempo\datos\tallys_no_sup\P14	
4	Statistic 4	Tally	Tally 4	C:\Users\Victor\Desktop\sims arena\AGVs_tiempo\datos\tallys_no_sup\P15	

Figura 6.20 Configuración de los módulos *Statistic* para el caso 3.

Después de realizado esto, se simula por un tiempo de 70 horas, para obtener el orden de los disparos y con el *Output Analyzer* se exportan los archivos para ser analizados por Matlab y revisar si se cumplen o no las restricciones. Se ejecutan los archivos *analisis_AGV_time_no_sup.m* y *analisis_AGV_time_sup.m* y se observa como los marcados prohibidos son alcanzables cuando no se cuenta con supervisor y como son no alcanzables cuando se analizan los datos obtenidos con el supervisor en RdP.

```

restriccion 1
restriccion 2
a marcado Prohibido Encontrado En Fila 32
a marcado Prohibido Encontrado En Fila 33
b restriccion 1
b restriccion 2

```

Figura 6.21 Resultado luego de ejecutar los archivos (a) *analisis_AGV_time_no_sup.m* (b) *analisis_AGV_time_sup.m*.

Finalmente se ha realizado la validación de los casos de estudio, y se cumplió con el objetivo de que el comportamiento del método de implementación de supervisores basados en RdP en ARENA, presenta los resultados esperados para cada ejemplo.

7. CONCLUSIONES, TRABAJOS FUTUROS Y LIMITACIONES

7.1. CONCLUSIONES

Para modelar un sistema sobre un software de simulación como lo es ARENA, se debe tener en cuenta que si se usa un formalismo de modelado de SEDs particularmente las redes de Petri, se logra estudiar el comportamiento del modelo de forma lógica, ya que las RdP recogen de forma detallada las relaciones dinámicas entre los diferentes elementos que hacen parte del proceso en estudio, ofreciendo una representación explícita del modelo, además de permitir alcanzar conclusiones rápidas sobre la dinámica del sistema y su evolución. Adicionalmente este formalismo es adecuado para modelar sistemas de manufactura permitiéndonos conocer la evolución de un producto a través de la red.

El método desarrollado para el diseño de RdP en ARENA, constituye un aporte significativo para la simulación en dicho software, ya que traduce la estructura de una RdP a un conjunto de módulos y bloques, que a través de un diagrama de flujo logran modelar una RdP y comportarse como tal; haciendo uso de unos bloques estándar configurados adecuadamente que fácilmente le permitirán a un modelador aun sin ser experto en ARENA, diseñar una RdP en esta herramienta, sin tener que probar una y otra vez con diferentes elementos que al final de cuenta le generan pérdida de tiempo y posibles problemas en la marcha.

Por otro lado, se logró implementar un supervisor basado en RdP que garantiza que el comportamiento de la red es el adecuado, generando acciones de control para habilitar y deshabilitar eventos controlables, que cumplen con las restricciones del sistema y no viola ningún marcado indebido de la RdP. Este supervisor para RdP se implemento en ARENA, a través de un método claro y sencillo, que satisface la trayectoria del conjunto de eventos deseados y evita los estados prohibidos.

Los métodos desarrollados fueron implementadas con éxito en tres casos de estudio, los cuales, evidencian las ventajas de modelar en RdP, incluir este formalismo en ARENA para simulación de sistemas y adicionalmente la importancia de agregar un supervisor para garantizar el comportamiento requerido del sistema, tanto en RdP como en el software ARENA.

Para cada caso de estudio se pudo verificar su comportamiento, a través, de una comparación entre dos herramientas de simulación, una de tipo académico y diseñada especialmente para la simulación en RdP como lo es PIPE y un paquete comercial que no contiene la traducción directa para RdP como es ARENA, los resultados obtenidos para la simulación en dichos software es el mismo, lo cual, nos permite dar cumplimiento al objetivo específico dos.

Finalmente se pudo evaluar el correcto desempeño del método de implementación de supervisor basado en RdP sobre ARENA, a través de una validación realizada con pruebas de simulación en dicho software y análisis de resultados a través de Matlab, que incluían los disparos de la transiciones para corroborar el control sobre los eventos controlables, con este procedimiento se dió cumplimiento al objetivo específico tres.

7.3. TRABAJOS FUTUROS

Ampliar la simulación de los modelos en ARENA con animaciones completas de los procesos usando el panel *Advanced Transfers* que permiten a los modeladores generar comportamientos más específicos para actividades “menos estándar”.

Agregar al método de implementación de RdP en ARENA, la posibilidad de trabajar con redes de petri coloreadas (RdPC) que permitan manejar modelos más complejos con un menor tamaño, y permita trabajar la evolución de la red a través de uso de variables. Así como también adicionar la posibilidad de trabajar con RdP jerárquicas.

Diseñar un método para la implementación de arcos inhibidores en la RdP simulado en ARENA, para extender el uso de supervisores que incluyen este tipo de arcos.

Dado que el software ARENA es de uso comercial incluye diferentes funciones para la empresa, razón por la cual se plantea abordar el panel de *Reports* que permite tener una experiencia más detallada con los datos, enfocándose desde el punto de vista estadístico para el análisis de las entidades y recursos; datos importantes para conocer el comportamiento cuantitativo de un proceso.

7.4. LIMITACIONES

La mayor limitante de este trabajo respecto a los modelos en RdP elegidos está dada por la versión de Arena que se ha utilizado. Ésta está destinada a uso académico y por tanto no permite la generación de modelos de tamaño importante. Es por eso que limita la cantidad de entidades que pueden estar en el modelo de manera concurrente a 150. Este número no permite manejar modelos muy complejos, ni generar animaciones adicionales.

Con respecto al diseño del supervisor, si se tiene un caso de estudio donde se haya realizado una restricción que incluya el vector de disparo, la cual de cómo resultado arcos conectados a transiciones no observables o no controlables, no es posible realizarle una transformación de restricción como la descrita en el literal 4.1.3.

REFERENCIAS BIBLIOGRÁFICAS

- [1] J. Acevedo and G. Mejía “*Programación Reactiva y Robusta de la Producción en un Ambiente Sistema de Manufactura Flexible: Llegada de Nuevas Órdenes y Cambios en la Prioridad de las Órdenes de Trabajo*,” Bogotá, Colombia. 2006.
- [2] A.L. Castro and G. Mejía “*Combinación De Métodos Heurísticos Con Redes De Petri Para La Programación De Sistemas De Manufactura Flexible*,” Bogotá, Colombia. 2006.
- [3] “Manufactura Flexible” [online]. Available: http://www.virtual.unal.edu.co/cursos/ingenieria/mecatronica/docs_curso/Anexos/TUTORIALcnc/DOCUMENTOS/TEORIA/MANUFACTURA%20FLEXIBLE.pdf [Accessed Dic. 15, 2011].
- [4] C. Cassandras and S. Laforune, *Introduction To Discrete Event Systems*, New York, USA: Springer, 1999, pp. 225-230.
- [5] “Sistemas de Eventos Discretos Redes de Petri” [online]. Available: http://www.eis.uva.es/~jossan/doct/sed/fich/SED_RdP.pdf. [Accessed Feb. 09, 2012].
- [6] G. Zapata “*Diseño De Automatismos Secuenciales Para Controladores Lógicos Programables*”. Medellín, Colombia. 2007.
- [7] J.E. Castellanos and L. Solaque, “*Modelado con redes de petri e implementación con grafcet de un sistema de manufactura flexible con procesos concurrentes y recursos compartidos*” Bogotá, Colombia. 2010.
- [8] L. Murillo. “*Redes de Petri: Modelado e implementación de algoritmos para autómatas programables*”. Tecnología en Marcha, Vol. 21, N.º 4, pp. 102-125, Octubre-Diciembre 2008.
- [9] M. Granada. “*Redes de Petri: Definición, Formalización y Ejecución*”. Cantabria, España. 2010.
- [10] M.L. Llorens “*Redes Reconfigurables. Modelización y Verificación*,” Valencia, España. 2003.
- [11] C. Salvati, L. Cofre and F. Suárez “*Teoría de Redes de Petri. Base teórica del proyecto final de la carrera de Lic. en Sistemas*”. [online]. Available: <http://freedownload.is/doc/apunte-de-redes-de-petri-7353670.html>. [Accessed Dic. 15, 2011].
- [12] M.E. Perdomo “*Modelado y simulación de eventos discretos mediante Redes de petri usando SIMPY*,” Venezuela. 2007.
- [13] M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers, 1993.
- [14] G. Zapata, J. Cardillo and E. Chacón, “*Aportes Metodológicos para el Diseño de Sistemas de Supervisión de Procesos continuos*” Medellín, Colombia. 2010.
- [15] P.J. Ramadge and W.M. Wonham, “*Supervisory Control of a Class of Discrete-Event Processes*,” *SIAM Jour. Control and Optimization*, Vol. 25, No. 1, pp. 206–230, January, 1987.
- [16] W. Kelton, R. Sadowski and D. Sadowski, *Simulación Con Software Arena*, Edition Mc Graw Hill, Vol 4. 2002.
- [17] M. Eurnekian “*Simulación en Arena de un problema de colas en un aeropuerto*”, Buenos Aires, Argentina. 2009.

- [18] X. Basogain and M. Olabe “Modelado y Simulación de Sistemas de Eventos Discretos”, June 2009. [Online]. Available: eduCommons Web site: <http://ocw.ehu.es/enseñanzas-tecnicas/modelado-y-simulacion-de-sistemas-de-eventos-discretos>. [Accessed Mar. 16, 2011].
- [19] C. Parra “*Evaluación del Desempeño de procesos industriales utilizando Redes de Petri Simuladas Bajo el Formalismo DEVS*”, Pamplona, Colombia. 2008.
- [20] “Complete Overview of Petri Nets Tools Database,” May 2002. [Online]. Available: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/complete_db.html. [Accessed Mar. 5, 2012].
- [21] P. Bonet, C. Lladó and R. Puigjaner “*PIPE v2.5: a Petri Net Tool for Performance Modeling*”, Palma de Mallorca, España. 2007.
- [22] S. Casillas “Tutorial de Matlab” [Online]. Available: <http://proton.ucting.udg.mx/~cheko/pdf/tutorialmatlab.pdf>. [Accessed Abr. 16, 2012]
- [23] Aditec Ingenieros S.A “Arena” [Online]. Available: <http://www.aditec-ingenieros.com/arena3.html>. [Accessed Mar 04, 2012].
- [24] D.A. Takus and D.M. Profozich “Arena® Software Tutorial” Presented at Winter Simulation Conference 1997.
- [25] Mísisis, “Arena Software” [Online]. Available: <http://www.mimesis-soluciones.com/gpage.html>. [Accessed Mar 04, 2012].
- [26] G. Mejía, D. Martínez and F. Torres “*Modeling and development of an ARENA® interface for petri nets. A case study in a Colombian cosmetics company*,” Bogotá, Colombia. 2008.
- [27] A. Guasch, M. Piera, J. Casanovas and J. Figueras. “*Modelado y Simulación, Aplicación a procesos logísticos de fabricación y servicios*,” Barcelona, España. 2003.
- [28] J.O. Moody and P.J. Antsaklis “*Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions*,” Notre Dame. 1999.
- [29] G. Cansever, Y. Li, L. Beklan and A. Jones, “*Supervisor Design, Which Is Based On Petri nets For Sequential Control of The Manufacturing System By Using Programmable Logic Controller*,” England, UK. 2001.
- [30] M. Ramírez, “*modelización y programación de una célula de control flexible mediante el empleo de redes de petri*,” Catalunya, España. 2011
- [31] J.F. Sánchez “*Control Supervisor de Sistemas de Eventos Discretos con Retroalimentación de Eventos y Estados*,” Guadalajara, México. 2007.
- [32] K. Yamalidou, J. Moody and P. Antsaklis, “*Feedback Control Of Petri Nets Based On Place Invariants*,” Great Britain, 1995
- [33] P. Dietrich, R. Malik, W.M. Wonham and B. A. Brandin, “*Implementation Considerations in Supervisory Control*,” *Synthesis and Control of Discrete Event Systems*, pp. 185-201, Kluwer Academic Publishers, 2002.
- [34] Z. Li and H. Hu “*On systematic methods to remove redundant monitors from liveness-enforcing net Supervisors*,” China. 2008.
- [35] C. Hellfritsch, “TimeNet - Examples of Extended Deterministic and Stochastic Petri Nets”, 2009