Universidad de Castilla-La Mancha

Escuela Superior de Ingeniería Informática de Albacete

Departamento de Sistemas Informáticos

Programa Oficial de Postgrado en Tecnologías Informáticas Avanzadas

Tesis de Máster

AUTOMATIZACIÓN DEL ANÁLISIS DE PRESTACIONES EN SISTEMAS CONCURRENTES

Julio de 2010

Alumno : **Diego Pérez Leándrez**
Directores : **Dra. Dª. M. Carmen Ruiz Delgado**
Dr. D. Diego Cazorla López

Agradecimientos

A los directores por su apoyo y al grupo ReTiCS por su colaboración

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Curriculum Vitae

En este capítulo se presenta un Curriculum Vitae actualizado y resumido del autor.

1.1. Información Personal

Nombre	Diego Pérez Leándrez
Nacionalidad	Española
Lugar de Nacimiento	Villarrobledo (Albacete)
Fecha de Nacimiento	
Ciudad de Residencia	
Teléfono	
Dirección	
Email	Diego.Perez@uclm.es
Título	Ingeniero en Informática

1.2. Educación

2001-2005	Universidad de Castilla-La Mancha Ingeniería Técnica en Informática de Gestión
2005-2008	Universidad de Castilla-La Mancha Ingeniería en Informática
2008-2009	Universidad de Castilla-La Mancha Master en Tecnologías Informáticas Avanzadas

1.3. Experiencia Laboral

2008	Instituto de Investigación en Informática de Albacete (I^3A) 10 meses trabajando en prácticas en el grupo de investigación ReTiCS
2009	Instituto de Investigación en Informática de Albacete (I^3A) 4 meses trabajando contratado a tiempo completo en el grupo de investigación ReTiCS

1.4. Congresos y Publicaciones

2009	*Automatic Performance Evaluation* [51] XVII Jornadas de Concurrencia y Sistemas Distribuidos (JCSD 2009). Sagunto (Valencia)
2010	*BAL Tool in Flexible Manufacturing Systems* [52] Thirteenth International Conference on Algebraic Methodology And Software Technology (AMAST2010). LNCS. Quebec (Canada)
2010	*BAL Tool: what else?* [56] 12th Italian Conference on Theoretical Computrer Science(ICTCS 2010). Camerino (Italia)

Capítulo 2

Resumen de Asignaturas

En este capítulo se presentan los resúmenes de las asignaturas cursadas en el máster, divididos por módulos. Para cada curso se describen los objetivos, los contenidos y un breve resumen del trabajo final realizado para superarlas.

2.1. Generación de Documentos Científicos en Informática

2.1.1. Objetivos

- Realizar búsquedas de información científica mediante consultas a las fuentes de información y documentación más populares (bases de datos electrónicas, repositorios, revistas, ...).
- Generar documentos y presentaciones científicas de calidad utilizando el entorno de composición LaTeX.
- Presentar técnicas que nos permitan aseverar cuando nuestra propuesta es superior a las existentes, planteándose como diseñar los experimentos en función del objetivo y como contrastar estadísticamente los resultados.

2.1.2. Contenidos

Los contenidos de este curso se dividen en tres partes:

- **"Metodología de investigación"** donde se presentan los diferentes medios de divulgación científica, las herramientas de búsqueda de información, la correcta organización de documentos científicos y la forma de evaluar el impacto de las publicaciones.

- **"Edición de documentos con LaTeX"** donde se describen los diferentes elementos para componer documentos en el entorno LaTeX y los paquetes disponibles para la creación de presentaciones mediante este entorno.

- **"¿Cómo contrastar mi algoritmo/método?"** donde se explica cómo se deben diseñar los experimentos y los contrastes de hipótesis estadísticos que se pueden aplicar sobre estos experimentos.

2.1.3. Tarea

La tarea principal realizada para superar este curso fue realizar un trabajo en el cual se presentaba el contraste de hipótesis realizado sobre unos problemas propuestos por el alumno. El contraste de hipótesis se debía de realizar aplicando un test paramétrico y uno no paramétrico, ambos test asignados por el profesor correspondiente. El trabajo debía ser realizado bajo el entorno de composición LaTeX.

TEST PARAMÉTRICO

Para esta parte del trabajo, el test asignado fue el ANOVA de dos factores. El problema elegido para aplicar este tipo de test fueron los resultados obtenidos al realizar el análisis de prestaciones sobre un sistema de producción flexible (FMS). Inicialmente, se realizo una descripción del test a utilizar, así como del problema a analizar. Posteriormente se comprobó que los datos elegidos cumplían las condiciones del test del ANOVA de dos factores. Comprobado se cumplían las restricciones del test, se inició el estudio estadístico sobre las muestras.

TEST NO PARAMÉTRICO

Para la segunda parte del trabajo, el test no paramétrico asignado fue el test de Wilcoxon. Al igual que con el test del ANOVA, se realizó una descripción del test, así como de la fuente de donde provenían las muestras elegidas. El análisis estadístico consistía en realizar una comparativa entre dos algoritmos de "búsqueda y poda" implementados en la herramienta BAL. Tras esta descripción se comprobó que las muestras cumplían las restricciones del test de Wilcoxon y se inició el estudio estadístico.

2.2. Introducción a la Programación de Arquitecturas de Altas Prestaciones

2.2.1. Objetivos

- Presentar las ideas básicas de la programación secuencial orientada a bloques.

- Presentar los modelos y las ideas básicas de la computación paralela.

- Adquirir una metodología de diseño y evaluación de los algoritmos paralelos.

- Introducir diversas librerías de programación de arquitecturas de altas prestaciones orientadas a la resolución de problemas de álgebra lineal numérica.
- Realización de diferentes sesiones prácticas que consoliden los conceptos introducidos en teoría.

2.2.2. Contenidos

Los contenidos de este curso se dividen en seis partes:

- **"Introducción a la programación de arquitecturas de altas prestaciones"** donde se presentan las nociones básicas de la programación paralela.
- **"Programación orientada a bloques"** donde se aplican los principios de la programación secuencial orientada a bloques en la implementación de diversos ejemplos de optimización.
- **"Introducción a las arquitecturas paralelas"** donde se presentan las posibles clasificaciones que se pueden hacer de las arquitecturas paralelas.
- **"Paradigmas de computación paralela"** donde se describen diversas metodologías y paradigmas para la programación paralela.
- **"Diseño de programas en arquitecturas paralelas"** donde se aplican las técnicas de descomposición y balanceo de carga para la implementación de varios ejemplos de optimización.
- **"Software de programación de arquitecturas de altas prestaciones"** donde se utilizan librerías de álgebra lineal numérica para la implementación de diversos ejemplos de optimización.

También se realizan tres sesiones prácticas para afianzar los conceptos explicados: Optimizaciones secuenciales, programación paralela con MPI y BLAS y programación paralela con BLACS y PBLAS.

2.2.3. Tarea

Para la evaluación del curso se consideró la realización de las prácticas propuestas y la exposición pública del trabajo realizado para la asignatura. El objetivo principal de la tarea fue presentar un estado del arte de cómo paralelizar la ejecución de la herramienta BAL. Para ello se presentaron varias opciones de cómo realizar en paralelo una búsqueda exhaustiva y sistemática en el espacio de soluciones, con el objetivo de reducir el tiempo de ejecución, el cual hasta el momento era de orden de complejidad factorial o exponencial. El trabajo se componía de una descripción previa del problema afrontar, así como, de un conjunto de algoritmos candidatos para la resolución del problema, en los cuales se presentaban los

diferentes modos de balancear la carga (balanceo estático y dinámico), asignación de procesos a procesadores, sincronización entre procesos y un análisis de como establecer la comunicación entre los procesos. Finalmente se realizó una comparativa entre los diferentes algoritmos y sus vertientes, en la cual se presentaban las principales ventajas e inconvenientes de cada uno de ellos.

2.3. Sistemas Inteligentes Aplicados a Internet

2.3.1. Objetivos

El objetivo principal de este curso es la difusión de los fundamentos de distintos formalismos relacionados con los sistemas inteligentes (redes bayesianas, algoritmos genéticos, ...) y como pueden aplicarse estos formalismos para resolver determinados problemas relacionados con Internet.

2.3.2. Contenidos

Los contenidos de este curso se dividen tres unidades temáticas:

- **"Modelado e inferencia en redes bayesianas"** donde se realiza una introducción a las redes bayesianas y se presentan las técnicas más utilizadas para modelado e inferencia de las mismas.
- **"Aprendizaje de redes bayesianas"** donde se describen diversas técnicas para la estimación de parámetros de la red, para la selección de modelos y para tratar con datos perdidos.
- **"Recuperación de información. Web Mining"** donde se realiza una introducción a los sistemas de recuperación de información (SRI) y se presentan diferentes modelos para la representación de estos sistemas, como una presentación de las técnicas utilizadas para resolver problemas de optimización, prestando especial atención a las técnicas basadas en algoritmos evolutivos.

2.3.3. Tarea

Para la evaluación de este curso se realizaron una serie de ejercicios relacionados con los distintos bloques temáticos que componen la asignatura:

- **"Modelado e independencias:"** se tenía que crear la red bayesiana que modelara la situación planteada en el ejercicio, así como, indicar las dependencias e independencias existentes en esta red.

- **"Inferencia y cálculo de probabilidades:"** se tenían que calcular las probabilidades que modelaran las redes bayesianas descritas en el ejercicio, así como, realizar la inferencia necesaria para obtener diversos resultados requeridos.
- **"Aprendizaje en redes bayesianas:"** se tenía que decidir cuál era la red bayesiana que mejor modelaba los datos proporcionados en el enunciado del ejercicio, optimizando la métrica BIC o la métrica K2.
- **"Lógica Difusa:"** se debía calcular y comparar los resultados obtenidos tras aplicar el proceso de inferencia sobre un conjunto difuso, así como la aplicación del algoritmo Wang and Mendel para generar un conjunto de reglas difusas lingüísticas, y obtener el espacio de búsqueda para el método wCOR.
- **"Modelado de un algoritmo genético:"** se tenían que describir los diferentes componentes del algoritmo genético que modelara el problema elegido entre un conjunto de ellos. El problema que se escogió para modelar fue el de la mochila 0-1.
- **"Implementación con LiO:"** se implementó utilizando la librería Lio el problema de las *N reinas*. Para esta tarea se tubó que elegir la representación más adecuada del problema de entre todos los que proporciona la herramienta. Además, se comprobó el rendimiento del algoritmo bajo distintas configuraciones de parámetros, así como la mejora que producían los operadores específicos creados.

2.4. Modelos para el Análisis y Diseño de Sistemas Concurrentes

2.4.1. Objetivos

- Mostrar los aspectos principales de los sistemas concurrentes, que han de ser capturados por las diferentes técnicas de modelado.
- Estudiar los diferentes modelos: algebraicos, redes de Petri y autómatas temporizados.
- Estudiar las extensiones probabilísticas y temporizadas de estos modelos.
- Estudiar las principales propiedades de interés sobre los sistemas concurrentes, y las técnicas de análisis de las mismas.
- Mostrar la aplicación de estas técnicas sobre sistemas de diverso tipo.

2.4.2. Contenidos

Los contenidos de este curso se dividen en tres bloques:

- **"Redes de Petri"** donde se estudian, modelan y analizan diversos sistemas aplicando esta técnica, presentándose además sus extensiones probabilística y temporizada.

- **"Álgebras de procesos"** donde se especifican los sistemas y sus propiedades mediante lenguajes formales basados en lógicas temporales.
- **"Autómatas temporizados"** donde se presenta la herramienta **UPPAAL** y se realiza el modelado de varias situaciones de ejemplo.

2.4.3. Tarea

Para la evaluación de este curso se tuvo que realizar la especificación del protocolo del bit alternante (Alternating Bit Protocol, **ABP**) mediante uno de los formalismos estudiados, el cual fue *Redes de Petri*.

Además, mediante la herramienta **UPPAAL** se presentó el uso del model checking en el análisis de sistemas flexibles de producción (FMS- Flexible Manufacturing Systems) libres de interbloqueos. Esta técnica nos permite realizar una búsqueda exhaustiva de todos los posibles casos y escenarios. La técnica de Model checking nos permite analizar sistemas industriales y comprobar mediante el grafo de transición de estados si el modelo del sistema satisface ciertas propiedades las cuales son expresadas mediante formulas.

2.5. Modelado y Evaluación de Sistemas

2.5.1. Objetivos

Los objetivos de este curso fueron los siguientes: modelar un sistema siguiendo las metodologías del proceso de simulación; evaluar las prestaciones de un sistema informático, utilizando los conceptos fundamentales de la teoría de colas; crear un modelo de red de interconexión, a través de simuladores informáticos.

2.5.2. Contenidos

Los contenidos de este curso se desarrollaron en tres partes:

- **Introducción al modelado y simulación de sistemas**: donde se presentaron los conceptos fundamentales del modelado y las partes del proceso de simulación.
- **Evaluación de prestaciones de sistemas informáticos** donde se presentaron los conceptos fundamentales de la teoría de colas, aplicados a la evaluación de sistemas informáticos.
- **Simuladores de redes de interconexión**, donde se presentaron tres ambientes de simulación.

2.5.3. Tarea

El objetivo principal fue presentar un estado del arte sobre la herramienta BAL. En él se especificó el poder de BAL y su capacidad para automatizar el análisis de prestaciones sobre ciertos sistemas. En la primera parte del trabajo se presentó una breve introducción del objetivo por el cual fue desarrollado BAL, así como los beneficios que aporta su utilización. A continuación se realizó una descripción de la herramienta BAL, en la cual se presentó las características, componentes y funcionalidad de la herramienta. Por ultimo se presentaron varios casos de uso sobre los que se utilizo la herramienta y en los que en líneas futuras sería utilizada.

2.6. Grid Computing

2.6.1. Objetivos

El objetivo principal de este curso es presentar el estado del arte en el diseño de sistemas distribuidos complejos mediante el uso de tecnología **Grid**.

Este objetivo principal incluye:

- Estudiar los principales problemas a resolver relacionados con la arquitectura Grid.
- Definir los requisitos de los sistemas Grid.
- Presentar el estado del arte de tecnologías relacionadas con los sistemas Grid como Grid economy, Grid portals, ...

2.6.2. Contenidos

Los contenidos de este curso se pueden dividir en dos partes:

- **"Review"** donde se realiza un repaso de los conceptos más importantes relacionados con sistemas distribuidos, arquitecturas orientadas a servicios y servicios web.
- **"Grid computing"** donde se presenta el estado del arte de la computación Grid, incluyendo la descripción de los elementos de una arquitectura Grid, la presentación de proyectos existentes que sirven como framework para la computación Grid (centrándose en **Globus**) y el planteamiento de futuros retos para la tecnología Grid.

2.6.3. Tarea

El objetivo principal de la tarea fue presentar un estado del arte sobre Grid Economy, el cual se afronto desde un punto de vista económico, considerando que el futuro de Internet

será ofrecer servicios por los cuales se pagará por su utilización, es decir, los recursos podrán ser comprados y vendidos según las necesidades de los usuarios.

La tarea se realizó dividiendo su contenido en tres partes diferenciadas:

- Una introducción a Grid Economy donde se presentaba el futuro de Grid Computing, y el porqué de Grid Economy.
- Una descripción de los principales Grid Schedulers
- Por último se presentó un número de casos de uso relacionados con Grid Economy. En esta parte se realizo un análisis de los diferentes casos de uso para Grid Computing, donde los recursos pueden ser comprados o vendidos, y de que modo se realiza.

En conclusión, se presento Grid Economy como una alternativa de comercio relacionada con el problema de la localización de recursos.

Capítulo 3

Estado del Arte

En este capítulo se presenta el estado del arte para la automatización del análisis de *Sistemas Concurrentes*. En la primera sección se realiza una breve introducción a los métodos formales para, posteriormente en la segunda sección, centrar la atención en el álgebra de procesos temporizada BTC, donde se presentarán sus principales característica, sintaxis y semántica. En la tercera sección se presenta el análisis de prestaciones que se llevará a cabo en los sistemas especificados mediante BTC. A continuación se presentan las principales herramientas desarrolladas sobre teorías basadas en técnicas de métodos formales. Posteriormente se describirá los *Sistemas de Producción Flexibles*, sobre los cuales se realizará un análisis de prestaciones haciendo uso de la herramienta BAL. En la sexta sección se presenta Métodos Formales en FMS. Para concluir, en la última sección se realiza un análisis de las ventajas de utilizar los métodos formales en el campo de WSN, así como de las características que hacen relevante el estudio de WSN bajo técnicas formales.

3.1. Métodos Formales

Los métodos formales se basan en el empleo de técnicas matemáticas y la lógica formal para especificar, analizar y verificar las propiedades de determinados sistemas de manera sistemática. Estos métodos incrementan la seguridad y eliminan posibles errores existentes durante las primeras etapas del desarrollo de nuevos sistemas.

Hace una década los métodos formales fueron considerados principalmente como una técnica para la verificación de sistemas y solo eran apropiados para el análisis de pequeños problemas académicos. El uso de métodos formales es aún una técnica joven cuya popularidad ha ido aumentando sobre todo en los procesos modernos de desarrollo industrial donde importantes empresas de desarrollo de sistemas han reconocido su potencial debido a su corrección y rigor matemático. Este auge de popularidad es debido principalmente a que la técnica de simulación no puede ser usada para comprobar todas las posibles alternativas en el análisis de un sistema. En el área de algoritmos probabilísticos esta característica permite realizar una distinción entre el uso de métodos formales y técnicas de simulación y testeo.

El principal objetivo de los lenguajes formales de especificación es el de servir de apoyo para el estudio de otros sistemas computacionales (implementados o no). La aplicación de los lenguajes de especificación para el estudio de sistemas se puede dividir en dos pasos.

- El primero consiste en la representación de las características *relevantes* del sistema a analizar, utilizando la sintaxis del lenguaje de especificación correspondiente. Cabe destacar que los lenguajes de especificación son, en general, de más alto nivel que los lenguajes en los que los sistemas a analizar están/serán implementados. Por ello, algunos de los aspectos concretos de la implementación serán usualmente ignorados. Tal abstracción de las características relevantes será la razón de la potencia de análisis que nos proveerán estos lenguajes, a través de la eliminación de algunos aspectos irrelevantes, el análisis automático o sistemático de las propiedades relevantes del sistema se hará (parcial o totalmente) posible.
- El segundo paso de la aplicación de los lenguajes de especificación consiste en la manipulación de la especificación formal del sistema de forma que se consigan extraer las propiedades relevantes de dicho sistema. Dicha manipulación se apoya, en general, en el amplio conocimiento que se dispone de la semántica del lenguaje de especificación. Ésta permite establecer de antemano aspectos tales como las posibles evoluciones futuras del sistema, el cumplimiento de ciertos invariantes, o la aparición de ciertas propiedades indeseables. De esta forma, el análisis del sistema puede consistir en el estudio de las propiedades de su especificación o bien en una cierta comparación entre la especificación y una implementación dada que, supuestamente, es conforme con la especificación.

Dependiendo de las características consideradas como relevantes en el primer paso, o de las propiedades a analizar en el segundo, el desarrollo de procedimientos automáticos de análisis podrán o no ser posible. No obstante, incluso en el caso de que alguno de los pasos no pueda llevarse a cabo de manera automática, el poder de análisis de dichas técnicas las convierte en herramientas imprescindibles para el estudio formal de sistemas computacionales de diversa índole.

La potencia de análisis de los lenguajes de especificación se basa en su capacidad para extraer las características relevantes de los sistemas a modelar. Por lo tanto, se deberán fijar en primer lugar estas características para el análisis concreto que desee realizarse. El principal objetivo en el que nos centraremos es el de analizar el comportamiento de sistemas *concurrentes o distribuidos*. La característica más relevante de dichos modelos es aquella intrínseca a tal tipo de sistemas: la *comunicación entre procesos*, elemento indispensable para el funcionamiento de cualquier sistema concurrente.

En consecuencia, los lenguajes de especificación centrados en la descripción de sistemas concurrentes tendrán como objetivo el modelado detallado de la comunicación entre los procesos presentes en el sistema. En esta línea se intentarán abstraer, en mayor o menor medida, todos los demás aspectos. Ello implica que el comportamiento interno de cada proceso, es decir, el conjunto de acciones ejecutadas por el proceso que no suponen una comunicación con otros procesos, se abstraerá de forma que no estará representado en los modelos.

Sin embargo, esta limitación afectará a la interpretación de aspectos que si deseamos

representar, es decir, a las comunicaciones entre los procesos. Ello es así porque las acciones desarrolladas internamente por los procesos determinarán qué comunicaciones deben llevarse a cabo con otros procesos. Por lo tanto, inevitablemente, la abstracción de ciertos detalles que no son *directamente* interesantes hará que, *indirectamente*, perdamos cierta información relevante sobre aquellos aspectos que consideramos interesantes.

Veamos por tanto que todo lenguaje de especificación *centrado* en el estudio de las comunicaciones entre procesos en los sistemas concurrentes convertirá, en virtud de la abstracción que realiza, ciertas *certidumbres* acerca del comportamiento que modela en *incertidumbres*. Concretamente, las *elecciones* que se produzcan de manera interna en el sistema relativas a realizar una determinada acción de comunicación u otra dejarán de ser predecibles en el modelo resultante, apareciendo así un cierto *indeterminismo*.

Se pueden encontrar multitud de métodos y técnicas formales, con lo que los criterios de clasificación son bastante variados, pero podemos agruparlos en tres grandes bloques atendiendo principalmente a la especificación de comportamiento:

- Métodos basados en álgebra de procesos: modelan la interacción entre procesos concurrentes. Esto ha potenciado su difusión en la especificación de sistemas de comunicación (protocolos y servicios de telecomunicaciones) y de sistemas distribuidos y concurrentes. Los más conocidos son: CCS,CSP y LOTOS.

- Métodos basados en Redes de Petri: una red de Petri es un formalismo basado en autómatas, es decir, un modelo formal basado en flujos de información. Permiten expresar eventos concurrentes. Los formalismos basados en redes de Petri establecen la noción de estado de un sistema mediante lugares que pueden contener marcas. Un conjunto de transiciones (con pre y post condiciones) describe la evolución del sistema entendida como la producción y consumo de marcas en varios puntos de la red.

- Métodos basados en lógica temporal: se usan para especificar sistemas concurrentes y reactivos. Los sistemas reactivos son aquellos que mantienen una continua interacción con su entorno respondiendo a los estímulos externos y produciendo salidas en respuestas a los mismos, por lo tanto el orden de los eventos en el sistema no es predecible y su ejecución no tiene por qué terminar.

Antes de acabar esta breve introducción a los métodos formales, nos gustaría recordar brevemente los beneficios que aporta el uso de los métodos formales:

- **Medidas de Corrección.** Proporcionan una medida objetiva de la corrección de un determinado sistema, a diferencia de las tradicionales medidas de calidad.

- **Temprana detección de errores.** Las técnicas basadas en métodos formales pueden ser aplicadas en las primeras fases del desarrollo del sistema, para detectar, prevenir y eliminar de una manera más temprana los defectos y errores existentes en el diseño del sistema.

- **Garantía de corrección.** A diferencia del *testing* o la *simulación*, el uso de técnicas como por ejemplo *Model Checking*, consideran todas las posibles alternativas de ejecución del sistema, con lo que si existiera alguna condición indeseada en cualquiera de esas posibles alternativas, esta sería encontrada. Por tanto se obtiene una cobertura total sobre el comportamiento del sistema.
- **Aproximación analítica a la complejidad del sistema.** La naturaleza analítica de los métodos formales es más adecuada que la técnica de *testing*, para verificar el comportamiento de determinados sistemas complejos. Probablemente una correcta abstracción puede ser utilizada para representar el comportamiento espacial de sistemas con comportamiento *determinista* o *no determinista*.

Veamos, en el apartado siguiente, concretamente el álgebra de procesos temporizada BTC con la que trabajaremos y afrontaremos los problemas y desafíos expuestos anteriormente.

3.2. Álgebra de Procesos Temporizada BTC

El álgebra de procesos BTC (*Bounded True Concurrency*) [55] está basada en CSP [35] a la que ha extendido sus sintaxis con el fin de poder considerar la duración de las acciones mediante el operador prefijo temporizado. La semántica operacional también ha necesitado ser extendida para poder tener en cuenta el contexto (número y tipo de recursos) en el que los procesos se ejecutan. El álgebra de procesos BTC tiene en cuenta el número de procesadores que existen en el sistema y la disponibilidad en cada momento. Con ello se consiguen aunar las características de los modelos algebraicos, que permiten modelar el comportamiento real de los procesos, pero sin suponer máximo paralelismo, con lo que se incluyen los razonamientos hechos en la teoría de scheduling.

La consecuencia más importante de este enfoque es que si hay más procesos que procesadores entonces, no se pueden ejecutar simultáneamente todos los procesos. Un proceso deberá esperar a que le sea asignado un procesador para poder continuar su ejecución. Con esto llegamos a la conclusión de que en un sistema, y por lo tanto en la especificación de dicho sistema, encontramos dos tipos de esperas en la ejecución de los procesos; las esperas relativas a la sincronización entre procesos y las relativas a la asignación de recursos (entre ellos el procesador). Las primeras son las que habitualmente encontramos en la teoría de concurrencia, mientras que las segundas sólo aparecen al considerar que el número de procesadores de un sistema es limitado y son las esperas que se consideran en la teoría de scheduling. Por lo que se realizará un tratamiento claramente diferenciado para los recursos expropiativos (aquellos que se pueden arrebatar al proceso que lo tiene sin que haya efectos adversos) y no expropiativos (aquel que no puede quitársele a su poseedor actual sin generar efectos colaterales).

BTC se basa en la noción de concurrencia real la cual distingue entre comportamiento concurrente y comportamiento en interleaving: $(a \mid b) \not\equiv (a.b + b.a)$. Esto significa que está permitido que más de una acción pueda ser ejecutada en el mismo instante de tiempo.

Por tanto, las principales características de BTC son:

- Álgebra de procesos temporizada.
- Toma en consideración el hecho de que los recursos del sistema deben de ser compartidos por los procesos.
- Capaz de manejar tanto recursos homogéneos como heterogéneos.
- Realiza tratamiento claramente diferenciado para recursos expropiables y no expropiables.
- Soporta paralelismo real: $a \parallel b \neq a.b + b.a$.

3.2.1. Sintaxis

En esta sección introduciremos el álgebra de procesos BTC mediante su sintaxis y una interpretación intuitiva de sus operadores.

Para tener lo más claro posible la sintaxis que presentamos, empezaremos diciendo que BTC considera tres tipos de acciones: **acciones temporizadas** *(timed actions)* que consumen tiempo y pueden usar recursos en caso de necesitarlos; **acciones sin tiempo** *(untimed actions)* que se utilizarán para sincronizaciones y que no consumen tiempo ni recursos y las **acciones especiales** *(special actions)* que utilizan recursos (pero no tiempo) y que se utilizarán para trabajar con recursos no expropiativos.

Se define la sintaxis de BTC como sigue:

Sea Act_T un conjunto finito de acciones temporizadas, Act_U un conjunto finito de acciones sin tiempo y Act_S un conjunto finito de acciones especiales:

$$Act_T \cap Act_U = \emptyset,\ Act_T \cap Act_S = \emptyset,\ Act_S \cap Act_U = \emptyset.$$

Así la sintaxis de los procesos del lenguaje queda definida por:

$$P ::= stop \mid a.P \mid \langle b, \alpha \rangle.P \mid P \oplus P \mid P + P \mid P \parallel_A P \mid recX.P$$

donde $A \subseteq Act_U$, $a \in (Act_U \cup Act_S)$, $b \in Act_T$, and $\alpha \in \mathbb{N}$, donde $\mathbb{N}$, representa el conjunto de números naturales. Además, se suponen procesos Id y un conjunto de acciones $Act = Act_U \cup Act_T \cup Act_S$.

En un sistema podemos encontrar $m \in \mathbb{N}$ tipos diferentes de recursos compartidos. Para cada uno de estos tipos de recursos definimos un conjunto Z_i formado por todas las acciones que requieren para su ejecución al menos uno de los recursos compartidos del tipo i. La

cardinalidad de este conjunto, esto es, el número de acciones que necesitan usar ese recurso será $x_i \in \mathbb{N}$. Así, cada conjunto Z_i se definirá de la siguiente manera:

$$Z_i = \{b_1,\ b_2,\ ...,\ b_{x_i}\}$$

Uniendo todos estos conjuntos Z_i se define el conjunto Z del siguiente modo:

$$Z = \{Z_1,\ Z_2,\ ...,\ Z_m\}$$

donde Z está formado por todos los conjuntos Z_i generados para cada diferente tipo de recurso compartido existente en el sistema. Además, se considera

$$\mathcal{N} = \{N_1,\ N_2,\ ...,\ N_m\}$$

donde $N_i \in \mathbb{N}$ representa la cantidad de recursos compartidos del tipo i disponibles en el sistema.

Así, recogiendo también la información necesaria para saber que acciones necesitan de cada recurso y cuántos de ellos hay disponibles, un proceso se denotará como:

$$[\![P]\!]_{Z,\mathcal{N}}$$

3.2.2. Semántica Operacional

Con la definición de la semántica operacional, se proporciona a los operadores del lenguaje un significado e interpretación precisa mediante la descripción de como un proceso se transforma en otro. Esta evolución se presenta mediante un sistema de transiciones.

Una **transición** es una tripleta $((P,\mathcal{N}),(\mathcal{Q},\mathcal{N}),(\mathcal{B},\alpha))$ que se representa como:

$$(P,\mathcal{N}) \xrightarrow{\mathcal{B},\alpha} (\mathcal{Q},\mathcal{N}').$$

Intuitivamente, el significado de la transición anterior es que el proceso P ejecuta las acciones contenidas en la bolsa B y pasa a comportarse como el proceso Q. Todas las acciones de una misma bolsa requieren el mismo tiempo para su ejecución, en este caso α.

Con el fin de poder considerar evoluciones internas de un proceso, se considera un nuevo tipo de acción no incluida en la sintaxis: $\tau \notin Act_T$, la cual representa una acción interna. Esta acción se usa para definir la evolución de una elección interna de dos procesos y tiene una duración de 0 unidades de tiempo.

Veamos que cada proceso lleva asociado el valor de $\mathcal{N}$, el cual recoge en todo momento el número de recursos de cada tipo disponibles en el sistema. Esta información es necesaria para poder trabajar con recursos no expropiativos.

El valor de $\mathcal{N}$ no se modifica a menos que un recurso no expropiativo sea solicitado o liberado. Después de esta afirmación parece lógico preguntarse por qué no se modifica cuando

una acción temporizada está utilizando un recurso (expropiativo) ya que a fin de cuentas el número de recursos disponibles está variando. Esto es así porque cuando una acción necesita un recurso expropiativo para su ejecución, este recurso es utilizado (retenido) por un periodo igual al tiempo que dure la acción, por lo que al finalizar la acción ese recurso ha sido liberado y por lo tanto restablecido en $\mathcal{N}$. Esto se mantiene incluso en el caso de que una acción deba ser ejecutada en distintos momentos (esto es, si la acción es interrumpida) ya que los recursos expropiativos pueden ser expropiados a un proceso y reasignados más tarde (el mismo o cualquier otro recurso del mismo tipo) sin efectos secundarios. En la Tabla 3.1 se presentan las reglas de la semántica operacional de **BTC**.

R1a) $\dfrac{}{(a.P,\mathcal{N}) \xrightarrow{\{a\},0} (P,\mathcal{N})}$

R1b) $\dfrac{}{(<b,\alpha>.P,\mathcal{N}) \xrightarrow{\{b\},\alpha} (P,\mathcal{N})}$

R1c) $\dfrac{0 < \alpha' < \alpha}{(<b,\alpha>.P,\mathcal{N}) \xrightarrow{\{b\},\alpha'} (<b,\alpha-\alpha'>.P,\mathcal{N})}$

R1d) $\dfrac{N_i > 0 \quad N_i \in \mathcal{N}}{(c_i.P,\mathcal{N}) \xrightarrow{\{c_i\},0} (P,\mathcal{N}')}$ *Donde* $\mathcal{N}' = \mathcal{N}$ *con* $N_i = N_i - 1$

R1e) $\dfrac{}{(\hat{c_i}.P,\mathcal{N}) \xrightarrow{\{\hat{c_i}\},0} (P,\mathcal{N}')}$ *Donde* $\mathcal{N}' = \mathcal{N}$ *con* $N_i = N_i + 1$

R2a) $\dfrac{}{(P,\mathcal{N}) \oplus (Q,\mathcal{N}) \xrightarrow{\{\tau\},0} (P,\mathcal{N})}$

R2b) $\dfrac{}{(P,\mathcal{N}) \oplus (Q,\mathcal{N}) \xrightarrow{\{\tau\},0} (Q,\mathcal{N})}$

R3a) $\dfrac{(P,\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N})}{(P,\mathcal{N}) + (Q,\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N})}$

R3b) $\dfrac{(Q,\mathcal{N}) \xrightarrow{B,\alpha} (Q',\mathcal{N})}{(P,\mathcal{N}) + (Q,\mathcal{N}) \xrightarrow{B,\alpha} (Q',\mathcal{N})}$

R4) $\dfrac{(P,\mathcal{N}) \xrightarrow{B',0} (P',\mathcal{N}) \wedge (Q,\mathcal{N}) \xrightarrow{B',0} (Q',\mathcal{N}) \wedge B' = B' \cap A \neq \emptyset}{(P,\mathcal{N}) \parallel_A (Q,\mathcal{N}) \xrightarrow{B',0} (P',\mathcal{N}) \parallel_A (Q',\mathcal{N})}$

R5a) $\dfrac{(P,\mathcal{N}) \xrightarrow{B_1,\alpha} (P',\mathcal{N}) \wedge (Q,\mathcal{N}) \xrightarrow{B_2,\alpha} (Q',\mathcal{N}) \wedge B_i \cap A = \emptyset \wedge \forall i\ |B_1|_{Z_i} + |B_2|_{Z_i} \leq N_i}{(P,\mathcal{N}) \parallel_A (Q,\mathcal{N}) \xrightarrow{B_1 \cup B_2,\alpha} (P',\mathcal{N}) \parallel_A (Q',\mathcal{N})}$

R5b) $\dfrac{(P,\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N}) \wedge B \cap A = \emptyset}{(P,\mathcal{N}) \parallel_A (Q,\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N}) \parallel_A (Q,\mathcal{N})}$

R5c) $\dfrac{(Q,\mathcal{N}) \xrightarrow{B,\alpha} (Q',\mathcal{N}) \wedge B \cap A = \emptyset}{(P,\mathcal{N}) \parallel_A (Q,\mathcal{N}) \xrightarrow{B,\alpha} (P,\mathcal{N}) \parallel_A (Q',\mathcal{N})}$

R6) $\dfrac{(P\{recX.P/X\},\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N})}{(recX.P,\mathcal{N}) \xrightarrow{B,\alpha} (P',\mathcal{N})}$

Tabla 3.1: Semántica Operacional

Vale la pena explicar algunas aspectos que son bastantes diferentes de otras álgebras. Empecemos por la regla $R1$ la cual muestra el funcionamiento del operador prefijo, pero que ha necesitado ser dividida para poder considerar los distintos tipos de acciones con las que BTC puede trabajar. La regla $R1a$ es la clásica para el operador prefijo cuando la acción no utiliza tiempo ni recursos, en BTC para las *acciones sin tiempo*, pero aquí se necesita incluir información adicional sobre la disponibilidad de los recursos en el sistema ($\mathcal{N}$). La regla $R1b$ establece que, dado un proceso P y una *acción temporizada* b con un tiempo de ejecución de

α unidades, primero se ejecuta dicha acción b durante esas α unidades de tiempo y después el proceso se comporta como P.

En la regla $R1c$ se permite la ejecución parcial de una *acción temporizada*. La acción b se ejecuta durante α' unidades de tiempo dejando el resto $(\alpha - \alpha')$ para más tarde. Recordar que $\alpha \in \mathbb{N}$ por lo que esta regla no genera infinita transiciones al trabajar con un dominio discreto.

Finalmente, las reglas del operador prefijo para las *acciones especiales* son $R1d$ y $R1e$. La primera es la utilizada cuando se solicita un recurso (no expropiativo) y la segunda para liberarlo. En la regla $R1d$, una acción especial que necesite un recurso de tipo i (c_i) para su ejecución sólo podrá ser llevada a cabo si en ese momento hay algún recurso de ese tipo disponible en el sistema $(N_i > 0)$. La acción representa realmente el hecho de asignar un recurso por lo que después de ser ejecutada, el conjunto $\mathcal{N}$ debe ser modificado y reflejar el hecho de que hay un recurso menos disponible de ese tipo en concreto. Cuando el recurso es liberado, la regla $R1e$ vuelve a modificar el valor de N_i pero ahora añadiéndole la unidad que se libera.

Como se puede ver en la Tabla, la regla $R2$ recoge el comportamiento del operador elección interna. La regla $R4$ recoge el mecanismo de sincronización donde sólo se tienen en consideración aquellas acciones que pueden ser ejecutadas tanto por el proceso P como por Q. Dado que todas las acciones de sincronización son *acciones sin tiempo*, el proceso completo evoluciona en 0 unidades de tiempo a $P' \parallel_A Q'$. La regla $R5a$ captura la ejecución simultanea de acciones que no son de sincronización con la restricción de que para cada tipo de recurso, se pueden estar ejecutando al mismo tiempo un máximo de N_i acciones y, finalmente, la regla $R6$ captura la semántica de la recursión.

3.2.3. Evaluación de Prestaciones

Con el modelo formal definido se capturan las características temporales del sistema a analizar. El trabajo realizado que aquí se presenta se basará en este formalismo, pero se centrará en la evaluación de prestaciones por lo que pasamos a definir el algoritmo inicial con el que se partirá para realizar dicha evaluación. Dicho algoritmo permite calcular el tiempo necesario para evolucionar entre estados. Para ello, a partir de la semántica operacional se puede construir para cada proceso un grafo de transiciones que ayudará en la evaluación formal de prestaciones. En este grafo se abstrae la información sobre las acciones y sólo se tiene en consideración información sobre el tiempo (duración de acciones). El grafo que se obtiene es un grafo dirigido con pesos, donde los pesos son siempre números positivos.

Una vez obtenido el grafo lo que se pretende resolver es un problema de maximizar el rendimiento relativo a minimizar el tiempo necesario para alcanzar el estado final (a partir del inicial). Para ello es necesario encontrar el camino más corto (con menos peso) entre estos dos estados lo cual se hace utilizando una modificación del algoritmo de Dijkstra.

Se denota con T_{Si} al tiempo necesario para evolucionar del *estado inicial* S_0 al estado S_i y $In(S_i)$ al conjunto de nodos predecesores del nodo S_i. Para cada estado $S_j \in In(S_i)$ existe

un arco etiquetado con α_{ji} que representa el tiempo necesario para que el sistema evolucione del estado S_j al estado S_i. Una representación gráfica se muestra en la Figura 3.1.

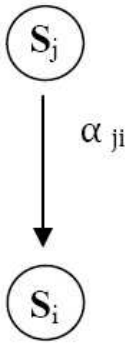

Figura 3.1: Grafo de Transiciones

A partir de aquí, T_{S_i} se puede obtener de forma recursiva usando la siguiente ecuación:

$$T_{S_i} = \begin{cases} 0 & \text{if } S_i = S_0 \\ \min\{T_{S_j} + \alpha_{ji} \mid S_j \in In(S_i)\} & \text{c.c.} \end{cases} \tag{3.1}$$

Con el objetivo de calcular (3.1), los n nodos del grafo son numerados desde 0 a $n-1$ y se supone que dicho grafo G está representado mediante su matriz de adyacencias donde $cost[i][j]$ es el peso del arco (i, j) y que los pesos de los arcos que no pertenecen al grafo están inicializados a ∞. Esto es:

$$cost[i][j] = \begin{cases} \alpha_{ij} & \text{if } S_i \in In(S_j) \\ \infty & \text{c.c.} \end{cases}$$

Sea S el conjunto de nodos (incluyendo S_0) para los que ya se ha calculado el mínimo tiempo necesario para llegar a ellos desde el nodo inicial. Con ello, el algoritmo de evaluación de prestaciones quedará como se muestra en la Figura 3.2.

3.3. Herramientas Basadas en Métodos Formales

Las herramientas que se presentan en esta sección implementan parcial o completamente diferentes teorías sobre métodos formales. En la literatura se pueden encontrar interesantes herramientas basadas en métodos formales pero todas ellas, por una o varias razones, se alejan de las características y objetivos de BAL. Existen varias herramientas basadas en CSP. FDR [44] fue la primera herramienta comercial para CSP, la cual ha jugado un papel importante en la evolución de la notación de CSP de un lenguaje de pizarra a un lenguaje concreto. Permite una amplia gama de condiciones de corrección para ser monitorizadas,

```
TiempoMin(int v, int cost[][SIZE], int n, int dist[]) {
// v es el nodo inicial
// cost es la matriz de adyacencias con pesos
// n es el número de nodos en el grafo
int u, minima; bool S[SIZE];

for (int i=0; i<=n-1; i++) {
    S[i] = false; dist[i] = cost[v][i];
} S[v] = true; dist[v] = 0; for (int num=1; num<=n-1; num++) {
// Elegimos u de entre los nodos que no están todavía en S
// el que tenga dist[u] mínima.
    minima = numero_grande;
    for (int i=1; i<=n-1; i++)
      if ((S[i]==false) && (dist[i]<minima))
            minima=dist[i]; u=i;
    S[u]=true;
    for (int w=1; w<=n; w++) // Actualizamos las distancias
        if ((S[w]==false) && (dist[w]>dist[u]+cost[u][w]))
            dist[w] = dist[u] + cost[u][w];
    }
}
```

Figura 3.2: Algoritmo de Evaluación de Prestaciones

incluyendo interbloqueos y vivacidad. ProBE [3] es otra herramienta basada en CSP, pero a diferencia de FDR, ProBE considera la necesidad del usuario de ganar comprensión del sistema interactuando recíprocamente con él, de manera, que el usuario controla la resolución de situaciones de no-determinismo y de la elección de acciones, visualizando la evolución del sistema. En conclusión, se puede decir que FDR está dirigido a verificar propiedades y ProBE es un simulador de CSP, permitiendo al usuario explorar interactivamente el comportamiento de los modelos. Por último hacer mención a CSPsim [17], cuya principal característica es que posee el potencial de explorar modelos con infinitos componentes. CSPsim simula sistemas en CSP.

Existen varias herramientas de *model checking* basadas en métodos formales, que merecen la pena ser mencionadas; SPIN [5], PRISM [34] y mCRL2 [33]. SPIN es un eficiente sistema de verificación para modelos software de sistemas distribuidos. Los sistemas para verificar son descritos en Promela (Process Meta Language) [58], el cual soporta modelado de algoritmos distribuidos asíncronos como un autómata no determinista. PRISM es un model checker probabilístico para el modelado y análisis de sistemas con un comportamiento aleatorio o probabilístico. Soporta tres tipos de modelos probabilísticos: *Discrete Time Markov Chains (DTMCs), Continuous Time Markov Chains (CTMCs) y Markov Decisión Processes (MDPs)*. mCRL2 es un lenguaje formal de especificación que tiene asociado un gran set de herramientas que pueden ser usadas para modelar, validar y verificar sistemas concurrentes.

Estas herramientas están destinadas principalmente a detectar y prevenir problemas en el software. Las herramientas nos permiten transformar especificaciones, generar y visualizar el espacio de estados y verificar propiedades.

La grán mayoría de herramientas que utilizan métodos formales están basadas en Redes de Petri. Entre ellas podemos destacar GreatSPN [11] o PIPE2 [23], basadas en Redes de Petri estocásticas. Otras herramientas basadas en Redes de Petri son PEP [12] o TINA [6] dedicadas a editar, simular y verificar Redes de Petri.

Kronos [16] y UPPAAL [13] son las principales herramientas basadas en autómatas temporizados. Kronos tiene como principal objetivo verificar sistemas en tiempo real donde los componentes están modelados mediante autómatas temporizados y las exigencias de corrección son expresadas mediante la lógica temporal TCTL. UPPAAL nos permite modelar, validar y verificar sistemas de tiempo real modelados mediante redes de autómatas temporizados, y extendido con datos. También podemos encontrar otras versiones de UPPAAL como pueden ser UPPAAL Tiga [9] , UPPAAL Tron [10] o UPPAAL Cora [8]. Esta última versión extiende el clásico autómata temporizado añadiéndole información de coste a los estados que representa las unidades de tiempo que se debe permanecer en esa posición. De este modo toda ejecución de un autómata temporizado posee un coste de tiempo global.

3.4. Sistemas de Producción Flexibles (FMS)

En los últimos años las industrias han tenido que adaptarse a las demandas cambiantes del mercado. Ante este hecho los sistemas de producción deben ser lo suficientemente flexibles como para poder dar satisfacción a la demanda de nuevos productos manteniendo calidad y competitividad en los costos. La necesidad de contar con sistemas de producción que se adapten rápidamente a las demandas, ha obligado el desarrollo de sistemas de organización y planificación que permitan adaptarse, sin pérdida de tiempo, a los diferentes requerimientos de producción.

Los Sistemas de Producción Flexibles consisten en un tipo de producción controlado por un ordenador central, que conecta varios centros o estaciones de trabajo informatizados con un sistema automático de manipulación de materiales. Su funcionamiento es, básicamente, el siguiente: los operarios llevan las materias primas de una familia de artículos hacia las estaciones de carga y descarga de materiales, donde el FMS comienza su actividad; bajo las instrucciones de un ordenador central, los elementos de transporte comienzan a mover los materiales hacia los diferentes centros de trabajo; en cada uno de ellos, los artículos son desplazados de acuerdo con su particular secuencia de operaciones, estando marcada la ruta a seguir por el ordenador central.

El objetivo perseguido es la sincronización de las actividades, de forma que se maximice la utilización del sistema. Como las máquinas automáticas pueden ser utilizadas para la ejecución de diversas tareas, es posible cambiar rápidamente sus herramientas, con lo que los tiempos de lanzamiento son muy cortos. Esta flexibilidad posibilita, además, que una operación pueda ser realizada por más de una máquina, dando lugar a la aparición de células virtuales. Gracias a

ello, la producción puede continuar aunque algunas máquinas estén paradas por cuestiones de mantenimiento. Cambiando y combinando las rutas a seguir se evitan los embotellamientos.

Sus principales características son:

- Máquinas polivalentes, que sirven para realizar diferentes tareas.
- Series de producción cortas y con gran variedad de producto.
- La secuencia de operaciones es variable, ya que el movimiento de material no es uniforme.
- Hay un mayor problema de asignación de recursos.

Figura 3.3: Sistema de Producción Flexible

Los FMS tienen un objetivo diferente al de la producción rígida. Mientras que en la producción rígida se trata de producir series largas de un número muy reducido de productos, en la producción flexible se trata de producir series cortas de un número elevado de productos. Por tanto, el objetivo es encontrar un conjunto de productos con necesidades de fabricación similares y minimizar el cambio de máquinas o los lanzamientos, debiendo para ello organizar las máquinas y herramientas necesarias para desarrollar los procesos básicos en áreas separadas llamadas células, pretendiéndose agrupar aquellas que permitan crear una pequeña línea de fabricación o de ensamblaje. De esta forma, las máquinas requieren tan solo ajustes menores para adaptarse a las necesidades de los diferentes lotes. Al simplificar los cambios de máquinas y las rutas de fabricación se reduce el tiempo que cada lote de pedido pasa por el taller, las colas de los artículos esperando a ser procesados son disminuidas considerablemente y en algunos casos eliminadas.

Analizando estos objetivos, se puede observar que el álgebra de procesos BTC es método adecuado para analizar este tipo de sistemas de producción, ya que BTC considera los

recursos del sistema como el número de recursos existentes, además del tiempo de duración de las acciones, pudiendo con ello conocer el número de recursos del sistema que optimice el funcionamiento del sistema de producción, así como el tiempo que emplearía el sistema en obtener un cierto número de productos a partir de un determinado número de recursos.

3.5. Métodos Formales en Sistemas de Producción Flexibles

El diseño de sistemas flexible de producción y el análisis de su comportamiento es uno de los problemas a los cuales se le ha prestado más importancia por parte de los investigadores. Relacionado con este aspecto se han desarrollado diversas herramientas computacionales que permiten el modelado y simulación de un determinado sistema con la finalidad de analizar un conjunto de características. Comúnmente estas herramientas proporcionan la facilidad de realizar simulaciones guiadas por el usuario, permitiendo elegir los eventos que se desea que tengan lugar, dirigiéndose de este modo el proceso de simulación, o por el contrario permitiendo realizar simulaciones de manera automática. Estas herramientas son de gran utilidad para el análisis del rendimiento de un sistema bajo ciertas condiciones. Pero para poder ser utilizadas como herramienta de planificación, en la representación del sistema además de representar la arquitectura del mismo se debería representar también las infinitas maneras en que éste se puede comportar.

Existen campos de aplicación en los que se requiere explorar todas las posibles combinaciones con el fin de poder determinar cuál es el mejor caso de organización y planificación, que nos permita mejorar el rendimiento y utilización de recursos para un determinado sistema, pero esta característica es inalcanzable por el tipo de herramientas de simulación anteriormente mencionadas. Además, actualmente las metodologías de desarrollo en la mayoría de sus análisis utilizan grandes sistemas de prueba que, aunque eliminan gran cantidad de errores, pueden resultar insuficientes para las áreas críticas del sistema desarrollado, por ello se hace imprescindible la utilización de métodos formales en el campo de sistemas de producción flexibles, los cuales aportan una serie de ventajas como:

- Aumentan la confianza en la validación de los requerimientos.
- Permiten definir que actividades implican mayor o menor esfuerzo, así como cuáles de éstos aportarán mayores beneficios.
- Permiten visualizar qué métodos y herramientas se usarán por cada tarea.

Por tanto, este nuevo tipo de sistemas debe ser estudiado desde un marco formal. En relación con el análisis del comportamiento, varios modelos son usados para establecer una relación dinámica entre la ocurrencia de eventos, operaciones y el estado del sistema. Estas técnicas de modelado incluyen la clásica Máquina de Estados Finitos [31], Redes de Petri [21, 25] y modelos de reglas [45].

La gran mayoría de las técnicas formales utilizadas en este área, principalmente Redes de Petri, incluyen estrategias de recorrido tanto *top-down* como *bottom-up*. Las técnicas basadas

en *bottom-up* comienzan con pequeños modelos que serán integrados para conformar el modelo en su totalidad, pero con el inconveniente que estos son más difíciles de validar. Las técnicas basadas en *top-down* por el contrario comienzan con un modelo global del sistema, el cual progresivamente en las sucesivas etapas es refinado. Las metodologías hibridas combinan las características de ambas técnicas.

Trabajo Relacionado

En la literatura se pueden encontrar varios estudios realizados en el campo de *Sistemas de Producción Flexibles* mediante el uso de técnicas formales. Por ello, a continuación se presentará un breve resumen de algunos de estos estudios.

- Se han propuesto y realizado estudios basados en Redes de Petri, uno de ellos es el propuesto en [59], el cual basa el modelado en Redes de Petri coloreadas con tiempo estocástico, basandose especialmente en el estudio de sistemas de producción. Una de las principales ideas es la de separar el modelado de los sistemas de fabricación de la rutas de producción para, posteriormente agruparlos para completar el modelo global y comenzar el análisis de prestaciones mediante una simulación o análisis numérico.

- En [32] se presenta un acercamiento a cómo realizar especificaciones de sistemas dinámicos mediante la extensión de un álgebra de procesos estocástica (PEPA). Concretamente se presentan dos especificaciones realizadas mediante la herramienta TIPP, para el problema de llenado de pallets, en el cual un robot tendrá que seleccionar de manera repetitiva una serie de bloques de una cinta transportadora, para posteriormente transportarlos y agruparlos sobre un pallet, el cual será retirado cuando éste se complete.

- El lenguaje *Chi* [38, 14] también ha sido utilizado para modelar sistemas de producción flexibles. Este lenguaje se basa en el lenguaje llamado *discrete Chi*, se trata de un simulador de eventos discretos con construcción probabilística. El objetivo principal para el cual se creó fue el de modelar sistemas de fabricación y alcanzar resultados que fueran útiles para este campo.

- En [30] se presenta el modelado de una celda de producción flexible utilizando los métodos formales orientados al modelo tal como Redes de Petri (RdP), y los métodos formales axiomáticos como RAISE (Rigurous Approach to Industrial Software Engineering). Para completar el estudio realizado en este trabajo, se presenta un caso de estudio en el que se modela utilizando Redes de Petri y RAISE.

- El trabajo realizado con PEPA (Performace Evaluation Process Algebra)[18] es similar al trabajo realizado con BTC. PEPA es una extensión estocástica de la clásica álgebra de procesos donde la duración de las acciones es representada por una distribución exponencial que varía aleatoriamente. PEPA al igual que BTC trabaja con interleaving, pero las características que las diferencian son que BTC utiliza tiempos discretos para la ejecución de sus acciones, y en el modo de trabajar con los recursos.

- En [26] se presenta una herramienta de simulación basada en Redes de Petri Coloreadas que permite explotar las características de representación y análisis de esta técnica de modelado de sistemas en la generación automática de esquemas de planificación de la producción.
- En [37] se presenta un formalismo el cual se ha diseñado específicamente para modelar Sistemas Flexibles de Producción. Con este método se intenta combinar las características de las Redes de Petri tradicionales como las Redes de Petri coloreadas. Además, en el estudio se presenta como este nuevo método desarrollado puede ser utilizado y es de utilidad en el estudio de Sistemas Flexibles de Producción.

3.6. Redes de Sensores Inalámbricos

Las Redes de Sensores Inalámbricos (Wireless Sensor Networks, WSN) en los últimos años han ido ganando gran atención mundial, particularmente con la proliferación en *Micro-Electro-Mechanical Systems (MEMS)* lo cual ha facilitado disminuir cada vez más el tamaño de los sensores.

Las redes de sensores inalámbricos están formadas por pequeños elementos hardware con capacidades sensitivas y de comunicación inalámbrica, también conocidos como motes (ver Figura 3.4), haciendo alusión al diminuto tamaño que se pretende que tenga su evolución hacia el polvo inteligente o "smart dust".

Los sensores son pequeños dispositivos (de bajo coste y con limitaciones a lo que se refiere a procesamiento y computación) que se componen de:

- Un microprocesador, el cerebro del sensor que hace que pueda recoger información, procesarla y comunicar sus propias medidas a la red.
- Una interfaz de transmisión/recepción, típicamente vía radio, aunque también existen otras opciones como puede ser una interfaz óptica o basada en ultrasonidos.
- Una fuente de alimentación, comúnmente baterías tipo AA.
- Las placas de sensores o los sensores externos propiamente dichos, que son los que tienen los mecanismos de medida (termómetros, medidores de presión, detectores de luz, etc.).

La batería es la principal fuente de energía en los sensores y frecuentemente el principal problema en el desarrollo e implantación de estas redes. En ocasiones, dependiendo de las condiciones ambientales donde se despliegue la WSN sea desplegada, los sensores puede obtener un aporte adicional de energía mediante pequeños paneles solares.

Los sensores tienen la capacidad de conformar una red ad-hoc, esto es, una red sin infraestructura física preestablecida ni necesidad de control central que coordine su actividad. Una característica de este tipo de redes es su capacidad de autoconfiguración, de modo que los

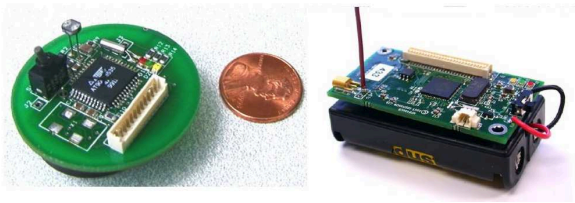

Figura 3.4: Sensor

sensores pueden trabajar como emisores o receptores y pueden establecer caminos de comunicación entre nodos sin visibilidad directa y modificar estos caminos si alguno de los nodos que participo en el encaminamiento falla. Además, las redes de sensores en su configuración adhoc pueden implementar protocolos de búsqueda que les permitirán conocer la posición de los diferentes nodos (y por ende, la topología de la red) de forma no centralizada, transmitiéndose la información salto a salto. Esto facilita notablemente el despliegue y mantenimiento de la red, que es resistente a caídas y fallos.

Los nodos de la red pueden medir parámetros del ambiente como la humedad y la temperatura, monitorizar parámetros relacionados con la salud de los usuarios como la medida de la saturación del oxígeno en sangre o pueden detectar presencia, por ejemplo. Gracias a las capacidades de enrutamiento de este tipo de redes, cualquier información recogida en algún sensor perteneciente a la red puede ser comunicada a los sensores que lo requieran.

Las WSN son empleadas en varios escenarios. Algunos ejemplos de su aplicación se pueden observar en la Figura 3.5:

- Aplicaciones militares: monitorización de fuerzas y equipos, vigilancia del campo de batalla, reconocimiento del terreno, detección de ataques biológicos, químicos o nucleares, etc.
- Aplicaciones medioambientales: seguimiento de animales, monitorización de las condiciones ambientales en cultivos, riego, agricultura de precisión, detección de incendios forestales, detección de inundaciones, estudios de contaminación, prevención de desastres, monitorización de áreas afectadas por desastres, etc.
- Aplicaciones médicas: telemonitorización de datos fisiológicos en pacientes, diagnóstico, administración de medicamentos, seguimiento de médicos y pacientes en hospitales, etc.
- Aplicaciones en el hogar/edificios: uso económico de calefacciones y aires acondicionados, ayuda a la evacuación de personas en caso de incendios, reconocimiento del estado de estructuras para controlar su deterioro natural o por efectos derivados de terremotos, por ejemplo, así como control de electrodomésticos, entornos inteligentes, control ambiental, etc.
- Aplicaciones en agricultura: cuyo objetivo es incorporar los últimos avances en agricultura de precisión, gestión de fincas y trazabilidad integral en la agricultura. En la Figura

3.6 se muestra una red de sensores, constituida por cientos de sensores desplegados sobre un viñedo, donde cada uno de los sensores obtienen la temperatura, nivel de luz y humedad del suelo para posteriormente comunicar los datos recolectados sobre una red multi-hop hacia un punto central donde posteriormente serán analizados.

- Aplicaciones industriales: seguimiento de vehículos, control de flota, control de inventarios, etc.

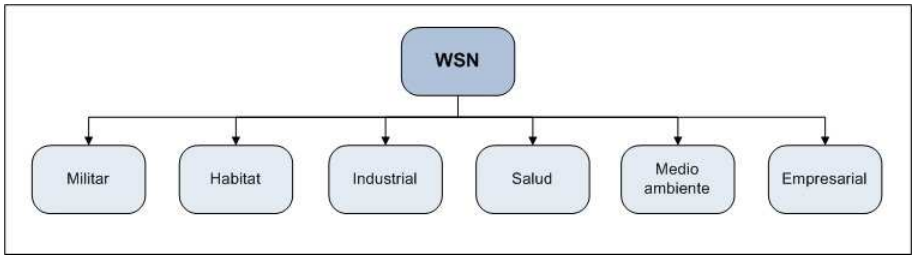

Figura 3.5: Aplicaciones de WSN

A diferencia de las redes tradicionales, las WSN tienen su propio diseño y recursos. Respecto a los recursos, los sensores poseen energía limitada (con lo que su tiempo de vida es limitado) y un pequeño rango de comunicación, un pequeño ancho de banda, así como limitada capacidad de procesamiento y almacenamiento. Las principales diferencias entre las WSN y las redes tradicionales, son:

- Número de nodos que constituyen la red.
- Los sensores son densamente desplegados.
- Los sensores tienen tendencia a los fallos.
- Cambios frecuentes en la topología de la red.
- Comunicación mediante *broadcast*.
- Los sensores poseen una fuente de energía limitada.
- Los sensores no poseen un identificador dentro de la red.
- Características *hardware* de los sensores.

3.6.1. Desafíos

A modo de resumen veamos cuales son los principales desafíos, en cuanto a investigación, que plantean hoy en día las redes inalámbricas de sensores. Dichos desafíos se pueden centrar en tres áreas principales.

Figura 3.6: WSN en Agricultura

- El consumo de energía: generalmente los nodos sensor están alimentados mediante una batería con una determinada cantidad de energía. El problema surge cuando, una vez desplegada una red de sensores, no es factible el cambio de baterías cuando éstas se agoten ya que la dispersión y el número de nodos que la suelen formar lo impide. Por lo tanto, se plantea la necesidad de diseñar los nodos y sus sistemas tanto hardware como software de forma que el consumo de potencia de los nodos sea lo más bajo posible de forma que se maximice el tiempo de vida de las baterías, garantizando un funcionamiento lo más autónomo posible de toda la red.

- Detección e interacción con el mundo físico: se trabaja en algoritmos de colaboración para el procesamiento de los datos captados para poder disponer de distintas vistas del ambiente monitorizado. También, provocado por la restricción del consumo de energía de los nodos y el consumo de energía que conllevan las comunicaciones, se hace necesario desarrollar algoritmos capaces de discernir si un dato capturado es o no importante antes de ser enviado al nodo central.

- Diseño de la red de comunicaciones: la cantidad de nodos, desde pocas decenas hasta miles, y las restricciones energéticas hacen que los protocolos y sistemas de comunicación que se utilicen sean cruciales para garantizar un funcionamiento correcto de la red de sensores. También en este punto, es importante tener en cuenta la latencia en la comunicación aunque no es un factor muy restrictivo ya que la sensorización de variables generalmente no es crítica, hay que mantener la latencia dentro de unos márgenes adecuados sobre todo si se introducen actuadores o tareas de control con restricciones temporales.

Cualquier sistema que este limitado por un cierto número de recursos, como puede ser el consumo energía, debe ser considerado para un perfecto estudio, desarrollo y funcionamiento del sistema. En el ámbito de las redes de sensores, el recurso más difícil de conocer es el consumo de energía. El tiempo de vida de la red quizás sea la métrica de evaluación más importante en las redes de sensores, ya que la red solamente podrá desempeñar su tarea mientras ésta esté viva. Por lo tanto, el tiempo de vida de la red indica la máxima utilidad que la red puede proveer. Si esta métrica es analizada previamente a su despliegue, el tiempo de vida estimado puede contribuir a la justificación del coste del despliegue de la red. Además, el tiempo de vida también es considerado como un parámetro fundamental en el contexto de disponibilidad y seguridad en redes [36], y depende del tiempo de vida de cada uno de los sensores que constituyen la red. Si el consumo de energía de cada uno de los sensores no se predice con exactitud, es posible que el tiempo de vida para la red predicho se desvié de manera incontrolada. Así, el modelo que representa a cada uno de los nodos (sensores) juega un papel importante en la predicción de dicha métrica, campo donde puede ser de gran utilidad el estudio de este campo o característica bajo los métodos formales.

La finalidad y el medio en el cual serán empleadas las WSN influirá en el desarrollo y despliegue de los sensores. Para reducir drásticamente el coste de la instalación de la red, las WSN tienen la habilidad de adaptarse dinámicamente a los cambios que se produzcan en el medio en el cual han sido desplegada. Los mecanismos de adaptación pueden responder a cambios de la topología de red o a cambios entre modos de operación drásticamente diferentes. Por ejemplo, la misma red que supervisa el escape de gas en una fábrica química podría ser reconfigurada para ser utilizada a su vez para rastrear la difusión de gases venenosos, con lo que la red podría dirigir a los trabajadores al camino más seguro durante su evacuación.

Existen varios factores que determinan el tamaño de la red, como el medio sobre el cual se despliega la red, el esquema de despliegue, y la topología de red. El tamaño de la red variará respecto al medio en el cual se despliegue. Por ejemplo, en espacios interiores se necesita una menor cantidad de sensores que en espacios exteriores. En espacios exteriores, pueden aparecer obstáculos que limiten la capacidad de comunicación entre sensores, con lo que esto afectará a la topología y por tanto en el modo de despliegue de los sensores. El posicionamiento de los sensores dentro de la red no tiene porque ser diseñado a priori, sino que se puede realizar de forma arbitraria, como puede ser en el caso de terrenos inaccesibles o de aplicaciones destinadas a desastres naturales. Por lo tanto, los protocolos y algoritmos diseñados para las redes de sensores deben de poseer la capacidad de auto-organización ante posibles cambios en la red.

3.6.2. Métricas de evaluación de WSN

El objetivo de esta sección es presentar cada una de las métricas de evaluación más importantes para las redes de sensores inalámbricas y la relación existente entre cada una de ellas. Las principales métricas de evaluación de WSN son: lifetime, coverage, cost and ease of deployment, response time, temporal accuracy, security, y effective simple rate.

La principal característica de estas métricas es que están interrelacionadas entre ellas. A

menudo puede ser necesario disminuir el rendimiento de una métrica, por ejemplo la velocidad de la red (rate), para aumentar el rendimiento de otra, como puede ser el tiempo de vida de la red (lifetime). Conjuntamente, todas estas métricas representan un espacio de medida multidimensional de la capacidad y rendimiento de las redes de sensores inalámbricas.

LifeTime

Una de las características más importantes y crítica en las redes inalámbricas de sensores es el tiempo de vida de la red, ya que no es una tarea fácil reemplazar los nodos de una red o recargar las baterías de estos, debido a su elevado coste operacional. Por lo que el objetivo principal de toda red es mantener el conjunto de sensores que conforman la red en funcionamiento durante el mayor espacio de tiempo posible.

El principal factor que limita el tiempo de vida de las redes de sensores inalámbricas es el suministro y consumo de energía por parte de los sensores, de manera que una posible solución es que cada uno de los sensores que integran la red administre de manera eficiente su consumo de energía para así maximizar el tiempo de vida de toda la red. En ocasiones, el principal objetivo no es estudiar cual es el tiempo máximo durante el cual la red está activa, sino analizar cual es el tiempo mínimo de vida de cada uno de los sensores, ya que por ejemplo, en redes de sensores orientadas a la seguridad, la inactividad de un sensor convierte la red en una red vulnerable. Esta es la razón por la que actualmente la mayoría de las investigaciones se centran en el diseño de protocolos y algoritmos con bajo consumo de energía, ya que el consumo de energía influye directamente en el tiempo de vida de la red.

Algunos artículos hablan del tiempo de vida de la red como un parámetro a mejorar, pero ninguno de ellos define exactamente el término Network lifetime [22].

Coverage

Conjuntamente al tiempo de vida de la red, la cobertura es una de las principales métricas de evaluación de las redes inalámbricas de sensores. Uno de los principales objetivos que se ha perseguido en las redes inalámbricas de sensores es desplegar una red sobre la mayor superficie posible.

Principalmente hay que diferencias los términos cobertura (coverage) de la red del término ámbito (range) de la red (ver Figuras 3.7 y 3.8).

Utilizando técnicas de comunicación Multi-hop, se puede ampliar la cobertura de la red mas allá del ámbito de cobertura de la tecnología utilizada. Por tanto, utilizando esta técnica, el ámbito de cobertura de la red se puede ampliar indefinidamente. Por el contrario, al aumentar el rango de transmisión de la red vuelve a surgir el problema del consumo de energía, ya que a mayor rango de cobertura, mayor consumo de energía, limitando esto el tiempo de vida de la red.

La escalabilidad es una de las principales características de las redes de sensores inalámbri-

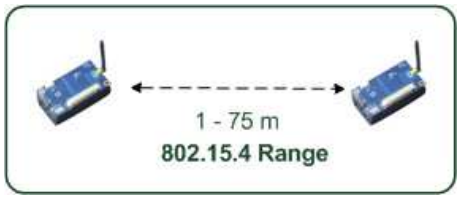

Figura 3.7: Ámbito de WSN

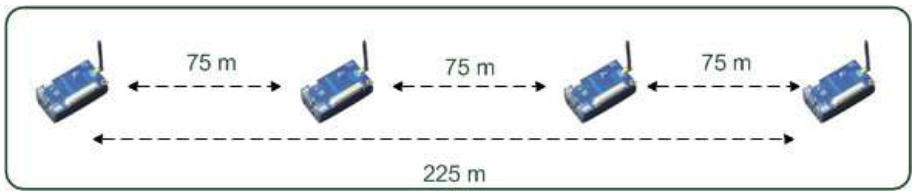

Figura 3.8: Cobertura de WSN

cas. Inicialmente, la red puede estar conformada de un pequeño número de sensores, pero progresivamente se pueden añadir más sensores a la red para cubrir de este modo las necesidades del usuario. Un inconveniente que surge con la escalabilidad es que al aumentar el número de sensores, aumenta proporcionalmente la cantidad de información a ser transmitida por la red, lo cual incrementa el consumo de energía por parte de los sensores disminuyendo de este modo el tiempo de vida de la red.

Cost and ease of deployment

Una de las características que diferencia las redes inalámbricas de sensores del resto de redes tradicionales es su facilidad de despliegue e instalación. Ligada a esta característica se encuentra la característica de *autoconfiguración*. Idealmente, la red debe ser capaz de autoconfigurarse ante cualquier cambio en la estructura de la red, como puede ser un cambio de posición de sensores, eliminación de sensores o la aparición de nuevos sensores en la red. Sin embargo, los sistemas reales poseen un cierto número de restricciones las cuales no se pueden violar. Por tanto la red debería ser capaz de volver a la última situación permitida cuando alguna de estas restricciones sea incumplida.

En conclusión, la red debe ser capaz de adaptarse ante posibles cambios en su estructura o cambios en el medio físico sobre el cual fue desplegada, auto-configurándose de una manera confiable y segura. Pero, al igual que las otras métricas, al disminuir el coste y aumentar la facilidad de despliegue, disminuye el tiempo de vida de la red.

Response Time

La capacidad de disminuir el tiempo de respuesta en la red entra en conflicto con el tiempo de vida de la red. En algunas aplicaciones de seguridad, por ejemplo las de detección de intrusos mediante sensores, el tiempo de respuesta conjuntamente con el tiempo de vida de la red son unas de las características mas deseadas en el sistema. En este tipo de sistemas, la alarma debe de activarse inmediatamente tras la detección de un intruso. Otro área donde el tiempo de respuesta es una característica deseada es en aplicaciones destinadas al control de sistemas de producción y equipamiento.

Pero la disminución del tiempo de respuesta, influye directamente en el consumo de energía y por tanto en el tiempo de vida de la red. Una solución a este problema es tener sensores en los cuales su radio solo esté activa durante ciertos periodos de tiempo, pero ésta no sería una opción deseada para ciertas aplicaciones de seguridad.

Security

Las redes de sensores inalámbricas deben de ser capaces de mantener en privado la información recogida del medio. Para aplicaciones destinadas a la seguridad, la confidencialidad de los datos es una característica significante y deseada. Por tanto, el sistema no solo debe mantener la privacidad de la información, sino que la red debe de ser capaz de autentificar la comunicación de los datos. Por ejemplo, no es deseable la posibilidad de introducir falsos mensajes de alarma dentro de una red destinada a seguridad.

Para ello se pueden utilizar técnicas de encriptación y criptografía, aumentando directamente el ancho de banda necesario y el tiempo de procesamiento por parte de los sensores en encriptación y desencriptación de la información y, por tanto disminuyendo el tiempo de vida de la red al aumentar el consumo de energía de cada uno de los sensores que intervienen en esta operación.

Effective Sample Rate

Se define el término *Effective Sample Rate* o también llamado *Tiempo de Comunicación*, como el tiempo trasncurrido desde que el sensor capta los datos del medio, hasta que llegan a un punto de colección de información, como puede ser la estación base.

En los arboles de enrutamiento, todo sensor debe manejar la información de todos los sensores descendientes a él. Por ejemplo, si cada sensor transmite una sola vez la información captada y cada sensor tiene 60 sensores descendientes, tendrá que enviar al resto de nodos 60 mensajes diferentes siendo capaz de recibir todos los mensajes en un mismo periodo de muestreo. Este aumento tiene efectos adversos sobre el rendimiento del sistema y aumenta la posibilidad de la transmisión de mensajes redundantes.

Para mejorar el tiempo de comunicación se pueden utilizar varias técnicas de compresión espacial y temporal, reduciéndose así el ancho de banda requerido mientras se mantiene el

mismo ratio de muestreo. Adicionalmente, se puede almacenar localmente en cada uno de los sensores la información recolectada para así evitar la transmisión y recepción de información redundante.

3.6.3. Simuladores

Según el diccionario de la Real Academia Española (R.A.E) [4] simular es «*Representar algo, fingiendo o imitando lo que no es* ». Por tanto, podríamos definir un simulador como una herramienta cuya finalidad es la representación de un sistema mediante un modelo simplificado, que nos permita analizar, verificar, validar o predecir el comportamiento de este sistema.

En el área de las redes inalámbricas de sensores, los simuladores son utilizados a menudo para analizar y evaluar nuevos protocolos antes de ser implementados, o realizar una comparación con otros protocolos o aplicaciones ya desarrolladas. Frecuentemente, los simuladores son utilizados durante la fase de diseño antes de iniciar la fase de implementación. Estos nos permiten analizar y evaluar el rendimiento del sistema, y comparar varios modelos y alternativas de configuración de manera controlada, para posteriormente decidir cual se adapta mejor a nuestras necesidades. Esto supone un ahorro de coste y tiempo durante el proceso de desarrollo, ya que la comprobación en el ambiente real, supone una tarea costosa (respecto a tiempo y dinero), difícil y compleja.

Existen varios simuladores de redes disponibles con características diferentes, por lo cual cada uno de ellos será más apropiado y eficaz en un ámbito, según las características del sistema a estudiar. A continuación se presentara algunos de los simuladores mas importantes que se han desarrollado hasta el momento.

Ns-2

Ns-2 [1] es el simulador mas popular para redes de sensores, ampliamente extendido en el ámbito académico. Ns-2 permite simular redes cableadas e inalámbricas (802.11 y 802.15.4) y simula un amplio rango de tecnologías de redes: protocolos TCP y UDP, implementa multicasting y algunas de las capas de enlace MAC para simular LANs, e incorpora funcionalidad para modelar el consumo de energía.

Se trata de un simulador de libre distribución dirigido por eventos. Su construcción modular lo hace realmente extensible y es quizás lo que lo ha hecho tan popular. Su facilidad de ser extendido y su arquitectura orientada a objetos permite el desarrollo de nuevos protocolos, así como facilidad de mantenimiento, reusabilidad, etc. Por el contrario, no posee una buena escalabilidad en el campo de redes de sensores. En casos con un gran numero de nodos, el tiempo de simulación es de orden exponencial debido a que cada nodo de la red puede interactuar con cualquier otro nodo, lo cual conlleva a un excesivo numero de dependencias a controlar.

OPNET

OPNET es una plataforma comercial para modelado y simulación de redes de comunicaciones y sistemas distribuidos, la cual puede ser utilizada tanto como herramienta de investigación, como una herramienta para el diseño y análisis de redes.

OPNET no es tan popular como otros simuladores como son Ns-2 o GloMoSim ya que no dispone de gran número de protocolos para simular. Por el contrario, OPNET es capaz de modelar de manera fiel diferentes características como son una representación de la transmisión de radio, las características de los transmisores, antenas a nivel de físico, ciertos obstáculos que puedan afectar en la comunicación, así como, desarrollar diferentes paquetes acorde a las necesidades del usuario. Al igual que ocurre con Ns-2, OPNET tiene problemas de escalabilidad con las redes de sensores.

TOSSIM

TOSSIM [7] es un simulador de eventos para las redes de sensores TinyOS. El principal objetivo de TOSSIM es realizar una simulación fidedigna de las aplicaciones diseñadas para TinyOS. TOSSIM se podría decir que es mas bien un emulador que un simulador. Por tanto, TOSSIM transporta directamente el código de la aplicación a simular al código de la plataforma donde será ejecutado, pero éste no será ejecutado de igual modo en el sensor que en una simulación, debido a las abstracciones que se consideran para simplificar el modelo.

Existen varias extensiones de TOSSIM, entre ellas las mas importantes son TinyViz, una herramienta de visualización y PowerTOSSIM [53] encargado de modelar el consumo de energía.

GloMoSim

GloMoSim inicialmente fue desarrollado para simular redes inalámbricas de sensores. Está implementado bajo el lenguaje Parsec, el cual es una variante de C para programación paralela, lo cual lo distingue del resto de simuladores, al poder ser utilizado en paralelo.

Entre las principales características de GloMoSim cabe destacar que es un buen simulador de redes IP móviles, tiene varias opciones para la propagación de radio, protocolos CSMA MAC, protocolos de enrutamiento e implementaciones de UPD y TCP. Por el contrario, tiene ciertas limitaciones para simular cualquier otro tipo de redes, fenómenos físicos externos a los sensores, condiciones medioambientales, soporte para sensores,etc. GloMoSim ha sido utilizado para evaluar redes de sensores inalámbrica, pero la fidelidad de sus resultados es un aspecto cuestionable.

Sidh

Sidh es un simulador por eventos desarrollado específicamente para redes inalámbricas de sensores. Sidh esta implementado bajo el lenguaje de programación Java, lo cual le aporta una mayor portabilidad.

Sidh se compone de un conjunto de módulos, de los cuales una parte se puede acceder mediante la llamada a diferentes métodos (estos módulos son definidos mediante una interfaz), y el acceso al resto de módulos se realiza mediante eventos. La utilización de la comunicación mediante eventos asegura la coordinación entre iteraciones.

Sidh tiene la propiedad de escalar de manera eficiente, siendo capaz de simular, redes con miles de nodos, de manera más rápida que la ejecución en tiempo real.

3.7. Métodos Formales en Redes de Sensores Inalámbricas

La aplicación de los métodos formales en el campo de las redes de sensores inalámbricas-es relativamente escasa. Esto es debido a la escasa existencia de herramientas destinadas a la verificación de este tipo de redes. Actualmente en esta área, los métodos formales deben de competir con los tradicionales simuladores destinados al estudio de redes. Es común, que aplicaciones de simulación como NS2 o TOSSIM acompañen a plataformas extensamente aceptadas como es el caso de TinyOS, por lo que reduce la existencia de herramientas destinadas al análisis, validación y verificación de WSN. Pero no todas las áreas de interés de las redes inalámbricas de sensores son cubiertas por las herramientas de simulación. El hecho que favorece a los métodos formales frente a las típicas herramientas de simulación, es que la técnica de simulación permite verificar ciertas propiedades pero no son capaces de comprobar todos los casos posibles que se pueden presentar. En el campo de algoritmos probabilísticos ésta es una característica que diferencia el uso de técnicas formales de las técnicas de simulación, además de poder crear una especificación exacta del modelo a analizar debido al rigo y corrección de las técnicas basadas en métodos formales.

Un argumento que motiva el uso de métodos formales en este campo, es que las redes de sensores se componen de una colección de pequeños dispositivo que solo se diferencian en su ID individual o dirección de red. Teniendo en cuenta este aspecto, las redes de sensores son unos excelentes candidatos para la utilización de métodos formales, ya que una vez que se haya diseñado el modelo que represente a un sensor, éste puede ser copiado y adaptado para representar al resto de nodos, debiendo únicamente de cambiar el identificador de dicho nodo. Además, las redes de sensores podrían ser modelados como sistemas concurrentes, teniendo el potencial adicional de que las reducciones de simetría pueden ser aplicadas de manera muy eficiente.

Los algoritmos de redes de sensores inalámbricas (WSN) presentan un conjunto de desafíos, apropiados para ser abordados con técnicas y herramientas basadas en los métodos formales, entre los cuales se encuentran:

- Modelado y razonamiento sobre el comportamiento dependiente del tiempo. Por ejemplo, la transmisión de mensajes está sujeto a los conocidos *delays*, así como que habitualmente los algoritmos suelen utilizar relojes, etc.
- Muchos algoritmos dependen de determinadas condiciones geométricas como localización, distancia, etc.
- Modelado de diferentes modos de comunicación (broadcast, punto a punto,..).
- Simulación y análisis de sistemas con gran cantidad de nodos distribuidos aleatoriamente.
- Tanto la corrección como el funcionamiento apropiado del algoritmo son aspectos críticos que deben ser analizados en profundidad.

Por tanto, creemos que los métodos formales pueden ser de gran beneficio y ayuda en el desarrollo de nuevos sistemas y protocolos en el ámbito de WSN, proporcionando nuevos modelos y técnicas de diseño, capturando a un alto nivel el comportamiento del sistema.

Una de las áreas más destacadas donde se ha utilizando los métodos formales es en el análisis de protocolos de seguridad, pero en los últimos años otros campos están ganando popularidad como son el análisis de algoritmos probabilísticos, la predicción de consumo de energía para pronosticar la duración de la batería de los sensores y por tanto el tiempo de vida de la red. Algunos ejemplos de aplicación son:

- *Estimación del consumo de energía:* Se han realizado investigaciones sobre cómo podría ser utilizada la técnica de model checking para el análisis de estimación del consumo de energía en WSN. En este aspecto se han desarrollado modelos matemáticos basados en la teoría de autómatas temporizados para determinar el empleo de energía de un sensor con un ajuste predefinido. En el contexto de optimización de energía existe un artículo [42] que presenta la investigación realizada sobre sistemas embebidos de tiempo real usando voltaje dinámico. En [29] se presenta un estudio realizado con autómatas probabilísticos sobre redes de sensores con baja señal de radio.
- *Corrección en el diseño de protocolos:* En este campo se han diseñado modelos probabilísticos acordes a protocolos de transporte, así como de protocolos diseñados para optimizar el periodo de establecimiento de la red. En este campo se han realizado estudios con autómatas para verificar la corrección de aplicaciones diseñadas para el sistema Operativo TinyOS bajo la herramienta llamada *FSMGen* [39].

Trabajo Relacionado

Existen diferentes modos de modelar el comportamiento de las redes inalámbricas de sensores, como el funcionamiento de los sensores que la conforman. Para conocer las ventajas del modelado y análisis de este tipo de redes mediante la utilización de los métodos formales es necesario conocer las alternativas que existen. En la literatura se pueden encontrar

varias propuestas (técnicas, lenguajes de especificación, herramientas) basadas en los métodos formales. Por ello a continuación se presentará un breve resumen de algunas de ellas.

- *CaVi* [15] es una interfaz que integra el estado del arte del simulador *Castalia* [54] y la herramienta de model checking *PRISM* [2]. *CaVi* provee una interfaz basada en el estado del arte de los métodos de simulación y los métodos formales de verificación para redes de sensores inalámbricas. Mediante esta interfaz podemos llevar a cabo la simulación y verificación del comportamiento en gran detalle de las redes de sensores inalámbricas.

  La funcionalidad de *CaVi* podría ser dividida en:

  - **Edición y Creación de las redes:** inicialmente *CaVi* genera automáticamente los modelos basados en PRISM (síncronos o asíncronos), utilizando las plantillas genéricas de los protocolos *flooding y gossiping*. Estos modelos son guardados como ficheros XML, pudiendo ser exportados para *Castalia y PRISM*.
  - **Simulación y Visualización de las trazas:** *CaVi* genera automáticamente el modelo *Castalia*, simulando y visualizando las trazas, las cuales pueden ser exportadas. La simulación consiste en mostrar que nodo envía paquetes a que otro nodo y con qué probabilidad sucede dicha transmisión.
  - **Monte-Carlo Simulation y Model Checking:** *CaVi* soporta simulación Monte-Carlo para el análisis de la sensibilidad de parámetros, así como, la capacidad de verificar ciertas propiedades.

- PAWSN [48] es un álgebra de procesos para WSN. Esta combina las características del álgebra de procesos CCS (paralelismo, composición secuencia, ...) con la aportación de nuevas características como son tiempo, probabilidad y comportamiento de consumo de energía. Para facilitar el diseño y análisis de los modelos especificados mediante PAWSN, se ha implementado la herramienta TEPAWSN, la cual permite realizar un análisis cuantitativo y cualitativo de los modelos especificados mediante PAWSN, traduciendo y adaptando estos a otros leguajes para que el análisis pueda ser realizado por otras herramientas (PowerTOSSIM, VMNet, PRISM, MRMC, UPPAAL, ...) acorde a los propósitos establecidos para el análisis (simulación, verificación y análisis de energía).

- En [27] se presenta un modelo formal probabilístico para modelar protocolos ***flooding*** y ***gossiping***, y analizar como influye la elección de diferentes modelos en los resultados del análisis de prestaciones. Para ello combinan el autómata probabilístico *PRISM* como herramienta para el model checking y el método tradicional de simulación *Monte-Carlo*.

- Luo y Tsai desarrollaron un nuevo modelo formal llamado *Space Time Petri Nets (STPNs)* para modelar WSN [46]. *STPNs* es un lenguaje formal que se extiende de *Time Petri Nets (TPN)* y *Colored Petri Nets (CPN)*. La nueva aportación de este lenguaje es la inclusión de información espacial a los lugares de las redes originales TPN y CPN para modelar las redes de sensores. Basada en esta idea, un conjunto de nuevos conceptos son propuestos:

- Información espacial: en el modelo que representa la red de sensores, los lugares representarán a los nodos de la red y las transiciones las acciones de transmisión de datos. Además, los tokens serán utilizados para representar el nivel de energía de cada uno de los nodos.
- Topología dinámica: este nuevo concepto es introducido debido a que los nodos que constituyen la red pueden tener una posición fija y definida o una posición aleatoria en la red. Para ello se programa un script que especifique de qué modo se establecen los nodos de la red.
- Transmisión broadcast: la transmisión broadcast es modelada mediante una transición especial la cual es llamada *broadcast transition*, que solo tiene lugares de entrada pero no de salida.

Para evaluar la capacidad del nuevo lenguaje STPN, implementa un entorno de simulación gráfico para construir los modelos y simularlos. En conclusión, se puede decir que la finalidad de este nuevo lenguaje es el modelado y análisis de protocolos y algoritmos diseñados para redes de sensores.

- En [50] los autores utilizan *Lamport's temporal logic of actions* [43] para modelar y simular protocolos de difusión con la finalidad de descubrir los arboles de enrutamiento.

- En el artículo [47] proponen el uso de *Real-Time Maude* como modelo formal para modelar este tipo de redes, además de simular y analizar algoritmos destinados a WSN. En el se realiza como ejemplo el modelado del sofisticado algoritmo OGDC.

- En [24], los autores presentan un intento de aplicar los métodos formales en la especificación y verificación de redes de sensores, así como la utilzación de *Active Sensor Processes* ***ASP*** como notación de especificación formal.

- $\mu - Calculus$ [40] es una generalización del estándar $\mu - Calculus$ de Kozen [41] para sistemas probabilísticos, el cual puede ser utilizado como un lenguaje de especificación y análisis en esquemas de redes inalámbricas respecto al consumo de energía de éstas. Por tanto este nuevo lenguaje de especificación y análisis contribuye a describir de manera formal un modelo de los protocolos de comunicación inalámbricos, incorporando capacidad para manejar el consumo de energía utilizando *acciones probabilísticas etiquetadas*. También contribuye a la creación de una nueva especificación cuantitativa de $\mu - Calculus$ ($qM\mu$) para el análisis del tiempo óptimo esperado en el modo de bajo consumo. Utilizando esta nueva especificación se es capaz de calcular el tiempo óptimo en el cual los nodos deben permanecer dormidos.

- El artículo [49] analiza los beneficios de un acercamiento formal al análisis de las redes inalámbricas; en particular se presenta la investigación realizada sobre cómo el model checking puede ser utilizado para validación de algunos diseños de decisión, y para proveer un perfil de comportamientos cuantitativos. En conclusión se puede decir que la finalidad del articulo es presentar como el uso de técnicas formales pueden ser de gran utilidad en el análisis de prestaciones y corrección de los protocolos diseñados para redes inalámbricas. Especialmente se presenta:

  - La facilidad de como los contra-ejemplos de los *model checkers* pueden ser utilizados para demostrar de manera concisa las limitaciones de los protocolos.
  - La facilidad de búsqueda exhaustiva de las técnicas probabilísticas de *model checking* pueden ser utilizadas para calcular un amplio rango de comportamientos probabilísticos.
  - El uso de modelos que especifican el funcionamiento de pequeños ejemplos pueden ser aplicados igualmente a sistemas de gran escala.

- En [28], el protocolo de control de acceso para redes de sensores LMAC ha sido modelado y verificado mediante la herramienta UPPAAL, para detectar y resolver colisiones que ocurren en el envío de información por parte de los nodos.

- Por último, en el artículo [19] el autor utiliza la herramienta HyTech para automatizar el análisis de sistema embebidos. HyTech es capaz de reconocer la causa o el factor bajo la cual un sistema satisface una condición temporal.

Capítulo 4

Trabajo de Investigación Realizado

En este capítulo se presenta el trabajo de investigación realizado en el que se ha participado. El trabajo realizado, esta principalmente enfocado en tres líneas de investigación:

- Diseño y elaboración de la herramienta de evaluación de prestaciones basada en parámetros temporales (BAL).
  - Diseño y elaboración del analizador sintáctico.
  - Diseño y elaboración del analizador de prestaciones temporales.
  - Diseño y elaboración del entorno gráfico.
  - Diseño y elaboración del editor de texto.
- Mejora de la eficiencia de la herramienta BAL.
  - Estudio, adaptación e implementación de técnicas de poda.
  - Paralelización.
- Aplicación de la herramienta BAL en *Análisis de prestaciones de Sistemas Flexibles de Producción (Flexible Manufacturing Systems FMS).*

4.1. Diseño y elaboración de la herramienta BAL

En esta sección se presentará la herramienta BAL, herramienta capaz de realizar de modo automático el análisis de prestaciones en sistemas especificados mediante el álgebra de procesos BTC. El análisis realizado por la herramienta BAL puede ser utilizado con dos finalidades diferentes, las cuales son:

1. Si se dispone de un sistema con un número fijo de recursos, con la utilización de BAL se puede averiguar cuál es el tiempo necesario para llegar del estado inicial al estado

final, con lo que podemos comprobar diferentes configuraciones para un sistema antes de su implantación y elegir entre éstas cual nos beneficia más, ahorrando con ello una gran cantidad de tiempo y dinero.

2. Partiendo de una especificación de un sistema, podemos hallar el número óptimo de recursos necesarios para cumplir un número determinado de restricciones.

Principalmente, BAL se compone de las siguientes partes:

- Interfaz gráfica.
- Asistente para la especificación de sistemas.
- Analizador sintáctico.
- Generador del grafo de transiciones.
- Analizador de prestaciones.

Un esquema de la estructura general de dicha herramienta puede encontrarse en la Figura 4.1.

4.1.1. Características

La potencia de BAL recae en las características del álgebra de procesos temporizada BTC, la cual nos permite realizar el análisis de prestaciones de sistemas complejos considerando los recursos compartidos que en cada instante están disponibles.

Al tratarse de un álgebra temporizada, los análisis realizados con este álgebra se centran en un análisis temporal. Para ello, a partir de la semántica operacional se construye un grafo de transiciones en el cual se abstrae la información sobre las acciones para centrar la atención en las características temporales.

Usualmente para sistemas reales complejos, el número de estados del grafo de transiciones es enorme, convirtiéndo el análisis de éste en una tarea que conlleva gran cantidad de esfuerzo y tiempo. Por ello, se desarrollo la herramienta BAL, la cual presenta como uno de sus principales objetivos la facilidad de la construcción y análisis de dicho grafo.

Una vez se ha realizado la especificación del sistema el funcionamiento de la herramienta puede ser dividido en tres etapas, (Figura 4.1):

1. **Análisis Sintáctico:** Inicialmente el analizador sintáctico comprobará que la especificación del sistema no contiene errores y se ha realizado acorde con las reglas sintácticas.
2. **Grafo de Transiciones:** Si la especificación del sistema es correcta, la herramienta construirá el correspondiente grafo de transiciones utilizando cada una de las reglas de la semántica operacional.

3. **Análisis de Prestaciones:** Finalmente la herramienta recorrerá el grafo construido en busca del tiempo mínimo necesario para alcanzar el estado final a partir del inicial. Esto es, determinar cual es el tiempo mínimo necesario para que el sistema finalice.

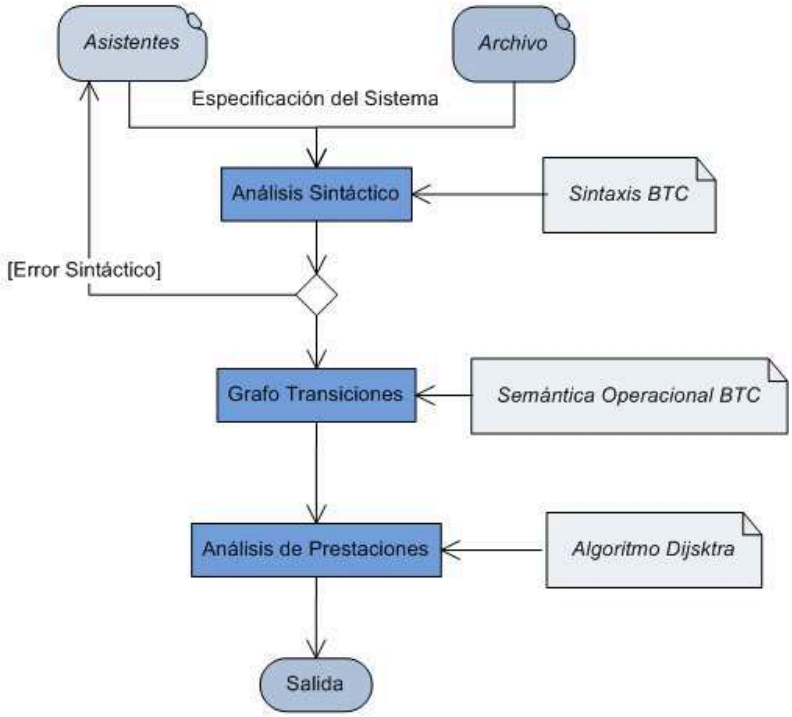

Figura 4.1: Arquitectura de la herramienta BAL

4.1.2. Sintaxis

Para obtener una sintaxis mas amigable e intuitiva se realizaron algunas modificaciones sobre la sintaxis de BTC. Los cambios realizados son únicamente sintácticos, no afectando estos a la semántica operacional de BTC.

En ocasiones, cuando se realiza la especificación de un sistema, es usual que se requiera que una secuencia de acciones se ejecute más de una vez. Por tanto, para obtener especificaciones más simples y claras sin necesidad de repetir varias veces una misma secuencia de acciones se definió el operador de repetición, llamado *Repeat*. Este operador se debe utilizar después de la secuencia de acciones que queremos que se repita. La sintaxis del nuevo operador es:

$$< Repeat, \omega >$$

donde $\omega \in \mathbb{N}$. Por tanto, para la especificación del proceso $P. < Repeat, \omega >$ el operador *Repeat* será sustituido por el proceso $P. < Repeat, \omega - 1 >$ mientras $\omega > 0$. Esto significa que las acciones que componen el proceso P serán ejecutadas $\omega + 1$ veces.

A continuación se presentará un ejemplo del funcionamiento de este operador:

$Process\_1 :=< action\_1, 9 > . < action\_2, 2 > . < action\_1, 9 > .$
$< action\_2, 2 > . < action\_1, 9 > . < action\_2, 2 > .stop;$

utilizando el operador *Repeat*, la especificación anterior pasaría a ser:

$Process\_1 :=< action\_1, 9 > . < action\_2, 2 > . < Repeat, 2 > .stop;$

Otros cambios de menor importancia que se han realizado sobre la sintaxis de BTC son:

- Para poder denotar el comportamiento especial de los recursos no expropiativos, los cuales deben ser solicitados antes de su utilización y liberados al finalizar, se ha modificado la sintaxis de las *acciones especiales*. El nombre de las *acciones especiales* irá acompañado del símbolo ˆ del siguiente modo: $\hat{}action$, para las acciones que solicitan el recurso no expropiativo y $action\hat{}$ para las acciones que lo liberan.

- De igual modo se ha modificado la sintaxis de las *acciones sin tiempo*, acciones utilizadas en la sincronización entre procesos. Esta modificación se ha realizado para impedir que un proceso se sincronice consigo mismo en casos innecesarios. Para que un proceso P esté sincronizado con otro proceso Q mediante la acción a, el proceso P' deberá de contener la acción conjugada (a') de la acción a.
  Por tanto, los procesos P y Q se sincronizarán mediante la acción a del siguiente modo:

$$P = a. < b, 2 > .....stop;$$

$$Q = a'. < d, 2 > .....stop;$$

  La acción a o a' no será ejecutada hasta que no aparezca otra instancia de sincronización de la misma acción, es decir su conjugado.

Con estos cambios, la sintaxis de las acciones para BAL será la siguiente:

- acciones temporizadas $\Rightarrow < action, time >$
- acciones sin tiempo $\Rightarrow action$ y $action'$
- acciones especiales $\Rightarrow \hat{}action$ and $action\hat{}$

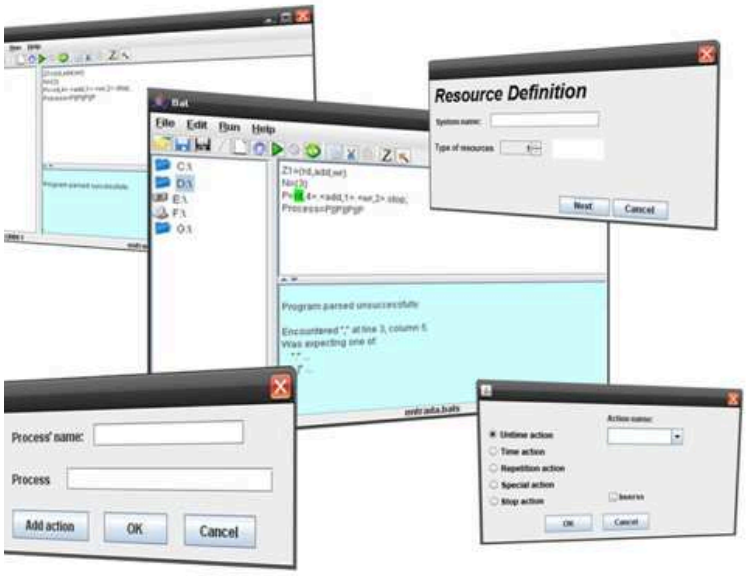

Figura 4.2: Interfaz Gráfica de la herramienta BAL

4.1.3. Interfaz Gráfica y Asistentes

Además de eficaz y eficiente, podemos decir que BAL es una herramienta fácil de utilizar ya que incluye una interfaz gráfica y varios asistentes de especificación que facilitan su utilización. En la Figura 4.2 se puede observar las principales ventanas de las que se compone la interfaz.

La utilización de la interfaz gráfica desarrollada para BAL es el modo mas comodo de realizar el análisis de prestaciones para sistemas concurrentes. Debido a su sencillez, con escasos conocimentos, cualquier usuario podrá crear una especificación de un sistema, analizar sintácticamente dicha especificación y realizar un análisis temporal del sistema. La ventana principal de la herramienta es la que se puede observar en la Figura 4.3, donde se especifica el nombre de las diferentes áreas.

- **Specification Area:** Área destinada a la especificación del sistema a analizar. Este área mostrará la descripción textual del sistema, que el usuario podrá escribir directamente en dicha área o utilizando el asistente desarrollado para esta tarea.
- **Files Tree:** A la izquierda del área de especificación se encuentra un árbol de directorios que permite al usuario explorar los distintos sistemas de almacenamiento de los que se disponga.
- **Notification Area:** Bajo el área de especificación se encuentra el área de notificaciones destinada a mostrar el consecuente de alguna tarea realizada, como puede ser el resultado del análisis sintáctico o la solución correspondiente al camino más corto entre un estado inicial y un estado final.

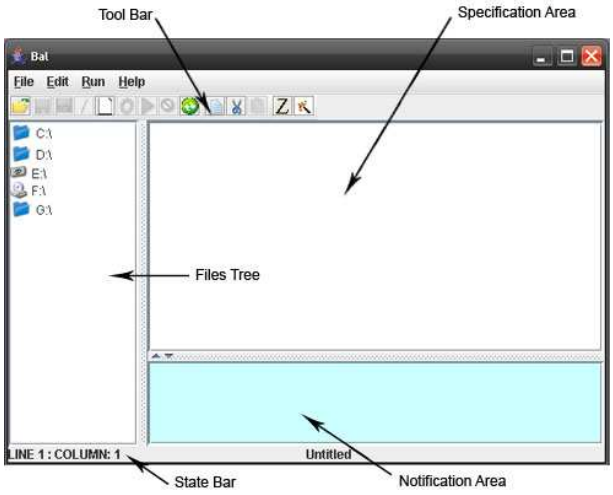

Figura 4.3: Ventana Principal

- **Tool Bar:** En la parte superior del interfaz, podemos diferenciar la barra de menús y la barra de herramientas. La barra de menús contiene todas las operaciones que la aplicación nos permite realizar, agrupadas según su funcionalidad en diferentes menús despegables. La barra de herramientas contiene iconos para ejecutar de manera inmediata algunas de las operaciones más utilizadas incluidas en la barra de menús.

- **State Bar:** En la parte inferior se encuentra la barra de estado que proporciona información adicional sobre la posición del cursor en el área de especificación, el estado de la tarea realizada o en proceso.

Hasta ahora se ha visto la interfaz desde un aspecto estático. Ahora pasaremos a ver cómo podemos utilizar todos estos elementos para realizar el análisis de un sistema concurrente. Previamente, para poder iniciar el análisis de prestaciones se tiene que especificar el sistema a analizar. Para facilitar la tarea de especificación del sistema a analizar e impedir errores sintácticos, la herramienta dispone de un asistente que guiará al usuario durante la especificación del sistema, facilitando que esta tarea se realice de un modo más sencillo e intuitivo. Este asistente consta de dos partes:

1. Resource Assistant $\mathbf{Z}_i$: Mediante este asistente se especificará la cantidad de recursos compartidos de los que se compone el sistema, así como el conjunto de acciones que requieren para su ejecución de dicho recurso (ver Figura 4.4). Dependiendo de que el recurso sea expropiativo o no expropiativo, las acciones que utilizan este recurso serán diferentes. Si el recurso es expropiativo, las acciones en el conjunto $\mathbf{Z}_i$ serán acciones temporizadas, por el contrario si el recurso es no expropiativo éste constará únicamente de acciones especiales.

2. Process Assistant: Asistente encargado de la declaración de los procesos que intervienen

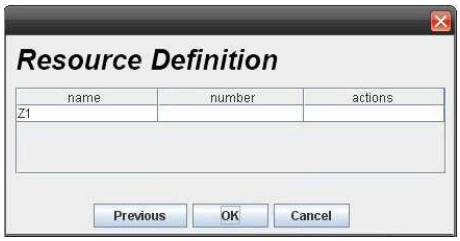

Figura 4.4: Asistente de Especificación de Recursos

en el sistema (ver Figura 4.5).

(a) Procesos (b) Acciones

Figura 4.5: Asistente de Especificación de Acciones y Procesos

Una vez especificado el sistema a analizar, se deberá realizar el análisis sintáctico sobre la especificación del sistema. Esta opción no estará activa si previamente no se ha proporcionado ningún sistema a analizar. El análisis sintáctico se podrá iniciar mediante la opción Syntax Check (compile) en el menú Run o mediante el icono situado en la barra de herramientas. Durante esta fase, el sistema comprobará si lo especificado está declarado en base a la sintaxis de BTC.

Finalizado el análisis sintáctico con éxito, se podrá iniciar el análisis de prestaciones del sistema, opción que permanecerá inactiva hasta no haber realizado y concluido con éxito el análisis sintáctico. El análisis de prestaciones se podrá iniciar mediante la opción Run del menú contextual o mediante el icono Play de la barra de herramientas. Una vez iniciado el análisis de prestaciones se activará la opción Stop para poder ser interrumpido por parte del usuario cuando desee.

Cuando la ejecución ha finalizado, el resultado es mostrado en el área de notificaciones *(notification area)*. La información que muestra dicha pantalla es la que se puede obsrevar en la Figura 4.6:

- El tiempo mínimo en el que puede ser ejecutado ese sistema, es decir, el tiempo mínimo necesario para alcanzar el estado final a partir del inicial.

- El estado de cada uno de los nodos del grafo que componen la rama más corta, indicando qué acciones hay preparadas para ejecutar y cuales de ellas son ejecutadas.

- El tiempo empleado por BAL en realizar el análisis de prestaciones.

Además, el usuario podrá guardar la traza con el resultado del análisis.

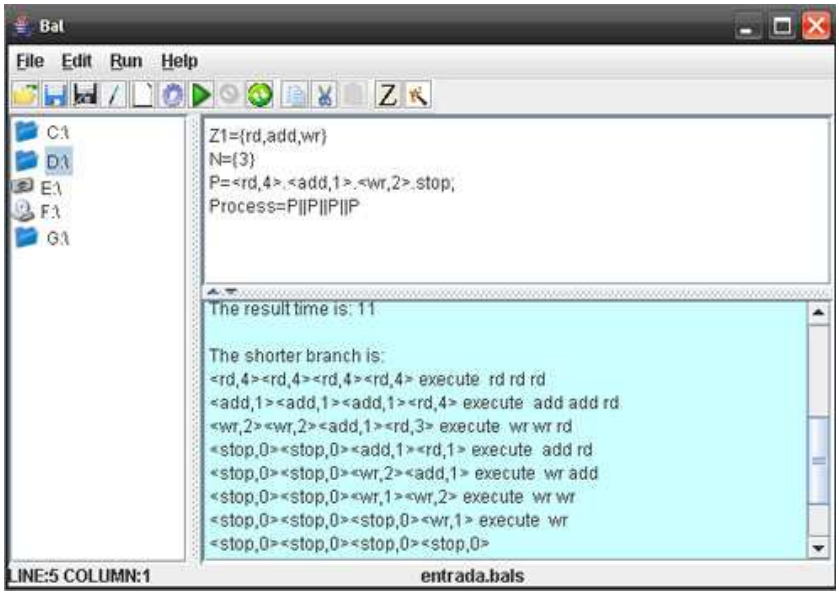

Figura 4.6: Pantalla de Resultados

4.2. Mejora de la eficiencia de la herramienta BAL

En las primeras etapas del desarrollo de la herramienta BAL, la principal limitación que se encontró fue que al incrementar el tamaño del sistema a estudiar, el número de nodos del grafo de transiciones aumentaba considerablemente, desembocando esto en una explosión de estados, debido a la gran cantidad de alternativas que se presentaban en el grafo. Para resolver este problema se han aplicado métodos de reducción del espacio de estados [57] [20], adaptando a nuestras necesidades la funcionalidad de los diferentes métodos estudiados.

Además, se ha incorporado en BAL la capacidad de ser ejecutado en paralelo, con lo que ha disminuido drásticamente el tiempo de ejecución y se han reducido las restricciones de memoria. Para ello, la técnica utilizada consiste en la paralelización de la búsqueda de cada una de las ramas de decisión y la utilización de memoria distribuida donde se ha utilizado MPI *(Message Passing Interface)* o PVM *(Parallel Virtual Machine)* para la comunicación y sincronización entre procesos.

4.2.1. Estudio, adaptación e implementación de técnicas de poda

Inicialmente, para resolver el problema de maximizar el rendimiento relativo a minimizar el tiempo necesario para alcanzar el estado final (a partir del inicial) se había implementado un algoritmo recursivo (una modificación del algoritmo de Dijkstra) que permitía calcular el tiempo necesario para evolucionar entre estados y que conforma la solución que maximiza el rendimiento del sistema. Para encontrar dicha solución se realiza un recorrido en profundidad del grafo (Depth first). Para ello, cada nodo que es explorado genera todos sus sucesores antes de que otro nodo sea explorado. Después de cada expansión, un hijo es seleccionado para ser expandido. Si se llega a un estado a partir del cual no se puede continuar, es decir, el coste alcanzado es mayor a uno de las ramas previamente analizadas, se regresa al punto más cercano de decisión con alternativas no exploradas.

Durante el análisis de los primeros sistemas se pudo comprobar que conforme el tamaño del sistema a estudiar aumentaba, el número de nodos del grafo correspondiente aumentaba considerablemente, con lo que el tiempo necesario para obtener el resultado final y los requerimientos de memoria aumentaban considerablemente. Realizando un estudio detallado sobre los grafos de transiciones generados durante el análisis, se descubrió la posibilidad de utilizar nuevos métodos de poda los cuales mejorarían el rendimiento de la herramienta. Estas nuevas técnicas se basan en los algoritmos ***branch and bounds***, los cuales pueden ser usados para simplificar la búsqueda en el espacio de estados.

Además, existía la posibilidad de explorar nodos que ya se habían analizado en algún otro camino, incrementado con ello la complejidad de la estrategia de búsqueda. Para resolver este problema se optó por no estudiar aquellos estados que ya habían sido analizados previamente en algún otro camino. Esto requiere guardar en memoria todos los estados ya analizados previamente. Para ello se modificó el algoritmo de búsqueda incorporando dos nuevas estructuras de datos: *VIEW Y EXPLORE*, donde se almacenaran aquellos estados ya explorados, obteniéndose el algoritmo que se muestra en 1.

Inicialmente *VIEW Y EXPLORE* estarán vacías y en cada iteración, conforme finalice la búsqueda en el espacio de soluciones de un nodo (se hayan analizado todos los hijos de un nodo), se añadirá a VIEW este nodo conjuntamente con el coste necesario para alcanzar un nodo hoja. *EXPLORE* almacenará los nodos candidatos a ser analizados.

Por tanto, cuando un nodo va ha ser analizado, antes de llamar a la función recursiva para proseguir con la búsqueda en profundidad, se comprobará si dicho nodo se encuentra almacenado en la lista (el nodo ya ha sido expandido en un camino previo), pudiendo ocurrir una de las siguientes alternativas:

- Si el nodo se encuentra almacenado en la lista, se aborta la búsqueda por dicha rama, evitando así el análisis de posibles alternativas estudiadas previamente.
- En caso contrario, se prosigue con el análisis hasta una futura situación de parada (fin de la rama, conste superior al de una rama ya analizada o nodo ya estudiado).

Tras implementar la técnica de poda sobre el algoritmo de generación del grafo de transi-

Algorithm 1 Algoritmo óptimo de planificación

```
VIEW := ∅;
cost := ∝
Function Cost(S, cn)
    Inputs
      S := node to explore
      cn := cost to arrive to node S
 1: EXPLORE := ∅
 2: if (S is in VIEW) and ((cn + getCost(S)) < cost ) then
 3:     cost = cn + getCost(S)
 4: else
 5:
 6:     ** \ add to EXPLORE childrems of S
 7:
 8:     EXPLORE ← setChildrem S;
 9:     if EXPLORE ≠ ∅ then
10:         while EXPLORE ≠ ∅ do
11:             N= getChildrem(EXPLORE)
12:
13:             ** \ obtain the cost of the transition between states
14:
15:             ct = getCostTransition(S,N)
16:             if (cn + ct) < cost then
17:                 cost(N,(cn + ct))
18:             end if
19:         end while
20:     else if cn < cost then
21:         cost = cn
22:         VIEW ← VIEW ∪ S
23:     end if
24: end if
```

ciones, se realizó un conjunto de pruebas para comprobar la corrección de la nueva versión del algoritmo implementado. Con los resultados obtenidos de esta batería de pruebas, se realizó un estudio comparativo entre la versión inicial del algoritmo y la nueva versión implementada.

El sistema que se eligió para realizar el estudio comparativo modela la suma de matrices de tamaño $10x10$. En este sistema solo se considera la existencia de un único recurso, *el procesador*, y contamos con el mismo número de procesos que filas (columnas) tengan las matrices a sumar. La especificación del sistema sería la siguiente:

$$Z_1 = \{rd, add, wr\}$$

$$N = \{n\}$$

$$[\![P]\!] = [\![< rd, 4 > . < add, 1 > . < wr, 2 > .stop]\!]_{Z,\mathcal{N}}$$

$$Process = P||P|| \ldots ||P$$

donde rd_i es la acción correspondiente a leer la fila i, add_i es la acción correspondiente a la suma de las filas i, wr_i es escribir el resultado de la suma y n representará el número de procesadores de los que se dispone.

La Tabla 4.1 muestra los resultados obtenidos en este análisis. Como conclusión de este estudio, observando los resultados se puede decir que se han obtenido una mejora significante respecto a la primera versión, ya que muchos casos que inicialmente no podían ser estudiados debido a problemas de falta de memoria o a un tiempo de ejecución excesivo, ahora pueden ser estudiados en un tiempo razonablemente corto.

Processors	Ver. Inicial	Ver. Poda
2	-	2' 41"
3	170h 21' 2"	8' 42 "
4	51h 12' 4"	6' 17"
5	22h 45' 21"	1' 42"
6	5h 12' 34"	9"
7	1h 23' 5"	2"
8	22' 24"	<1"
9	22"	<1"
10	<1"	<1"

Tabla 4.1: Resultado Suma de Matrices

4.2.2. Paralelización

Para mejorar el tiempo de búsqueda de los algoritmos secuenciales se puede utilizar varios procesadores trabajando en paralelo y colaborando entre ellos. Cada uno de los procesadores analizará un subgrafo del grafo de estados, reduciéndose con ello el tiempo de ejecución. Idealmente el tiempo de búsqueda se reduciría en proporción al número de procesadores que colaboren en la búsqueda, pero también hay que tener en cuenta el factor de sobrecarga introducido debido al intercambio de información entre los procesadores, reduciéndose con ello el Speedup.

La estrategia utilizada para paralelizar la búsqueda en el espacio de soluciones del grafo, divide el grafo inicial en subgrafos de menor tamaño que serán asignados a cada uno de los procesadores para ser analizados. Para realizar esta tarea se optó por utilizar el algoritmo de *Branch and Bounds paralelo*.

Desafortunadamente el espacio de estados no está balanceado y nuestro grafo dispondrá de una estructura desconocida (ver Figura 4.7), por lo que uno de los problemas de este algoritmo es la distribución del espacio de búsqueda sobre cada uno de los procesadores. Por lo tanto, hay que conseguir obtener subgrafos equitativos respecto al trabajo a realizar en cada uno de ellos.

Existen dos alternativas a la hora de distribuir la carga de trabajo entre los procesadores: balanceo estático de la carga o balanceo dinámico de la carga.

Se ha podido apreciar que la partición estática no es conveniente para resolver problemas sobre un grafo no uniforme debido al pobre rendimiento que ofrece. La opción de balanceo estático de la carga es óptima cuando se conoce la estructura del grafo y ésta es uniforme, ya que el particionamiento se realizará siempre de la misma forma, asignando la misma cantidad de trabajo a cada uno de los procesadores que intervienen en el análisis.

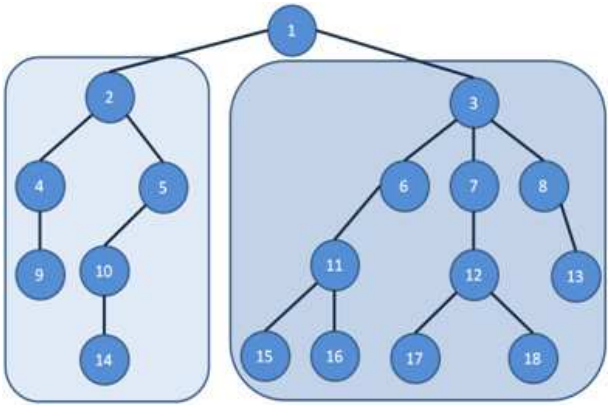

Figura 4.7: Balanceo Estático

Por lo tanto, la opción adecuada en nuestro caso es balancear el espacio de búsqueda entre los procesadores de manera dinámica. Sin embargo, esta opción conlleva la tarea de obtener de manera anticipada la distribución de los espacios de búsqueda. Antes de realizar la asignación de trabajos a los diferentes procesadores, se realizará un recorrido en anchura de los primeros niveles del grafo hasta que el número de nodos en dicho nivel, sea superior al número de procesadores de los que se disponga. Ya que, al ir profundizando en el grafo, se generan un mayor número de nodos, con lo que se puede realizar una división más uniforme entre los procesadores. Además, durante esta etapa se comprobará que no se exploraran aquellos nodos que generen un mismo espacio de estados. De este modo se establecerá una cota superior dinámica, que nos indicará el nivel a partir del cual se paraleliza la búsqueda (ver Figura 4.8).

Un esquema general de cómo trabaja la distribución de trabajo dinámica se encuentra en la Figura 4.9.

Inicialmente el espacio de búsqueda es asignado a un procesador y los otros procesadores esperan a que les sea asignado un trabajo. Cuando un procesador está inactivo, solicita trabajo al nodo maestro. Cada uno de ellos posee una pila donde almacena en orden los nodos que ya

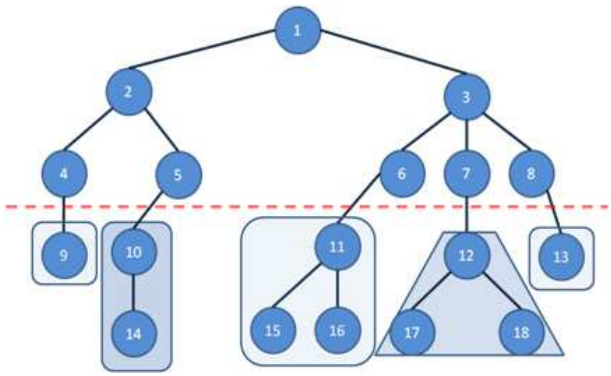

Figura 4.8: Balanceo Dinámico con Cota Superior

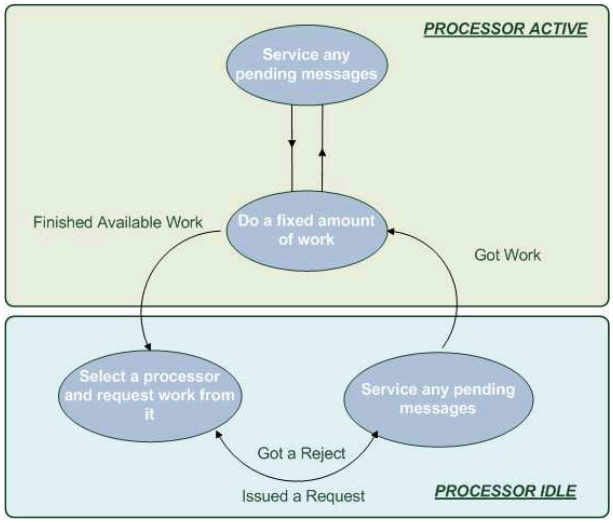

Figura 4.9: Esquema Distribución Dinámica del Trabajo

han sido expandidos y los nodos ya analizados por el resto de procesadores.

Cuando un proceso termina:

- Indica al maestro el valor óptimo que ha encontrado.
- Pide nuevo trabajo al maestro.
- Realiza el backtracking inicializando su valor a uno indicado por el maestro.
- Cuando el maestro asigna un nuevo trabajo a un proceso, le indica el mejor valor encontrado hasta el momento.

La técnica empleada para realizar la división de trabajo en un procesador es *Random Polling (RP)*, elección aleatoria de los procesadores a quienes solicitar más trabajo cuando su

tárea finalice. En este caso, cada procesador tiene la misma probabilidad de ser el donador. Es decir de otorgar carga de trabajo.

4.3. Aplicación de la herramienta BAL en FMS

Una de las áreas de estudio donde se ha empleado la herramienta BAL es en el análisis de prestaciones de FMS [52]. Debido al hecho de que la construcción de cualquier nueva cadena de producción requiere una inversión grande, la fase de diseño adquiere una gran importancia. En este contexto BAL nos ha permitido el estudio de diferentes configuraciones para varios sistemas de producción antes de que estos fueran desarrollados o en la mejora de algunos ya establecidos.

4.3.1. Caso de Estudio FMS

Varios FMS han sido evaluados mediante la herramienta BAL. En esta sección se presentará el estudio de una celda de ensamblaje utilizada en la mayoría de FMS. La celda se compone de tres máquinas y un robot (ver Figura 4.10). La máquina $M1$ produce piezas de tipo A y la máquina $M2$ produce piezas de tipo B. Mas tarde, la máquina $M3$ ensamblará una pieza de cada tipo para obtener la pieza final la cual abandonará la celda de ensamblaje. El movimiento de las piezas dentro de la celda de ensamblaje lo llevará a cabo un robot, el cual trabajará en exclusión mutua.

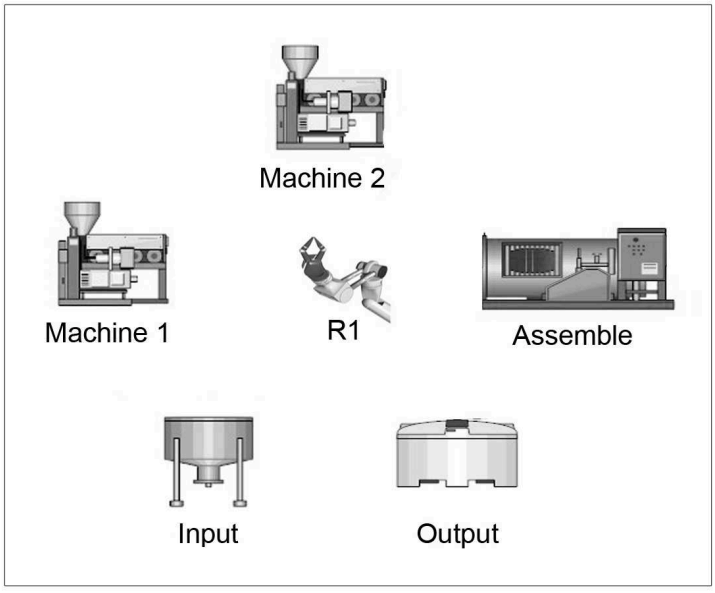

Figura 4.10: Celda de Ensamblaje

En este sistema podemos encontrar 4 tipos de recursos: tres máquinas ($M1$, $M2$, $M3$)

y un robot. El recurso que modela las máquinas se trata de un recurso expropiativo y el recurso robot es un recurso no expropiativo. Para cada uno de ellos definimos un conjunto de acciones las cuales utilizan al menos un recurso de ese tipo para su ejecución. Para los recusos expropiativos ($M1$, $M2$ y $M3$) se definen los conjuntos: Z_1= { *process_m1* }, Z_2= { *process_m2* }, Z_3= { *process_m3* }. Y las acciones que necesitan un recursos no expropiativo se especifica como: Z_4= { *r_robot* }.

Sobre este sistema se estudio cómo influye el número de recursos del tipo *Robot* en el funcionamiento del sistema. Inicialmente supondremos que solo disponemos de un recurso de cada tipo (N={1,1,1,1}), con lo que la especificación del sistema será la siguiente:

$$Z_1 = \{process\_m1\} \quad Z_2 = \{process\_m2\} \quad Z_3 = \{process\_m3\} \quad Z_4 = \{r\_robot\}$$

$$\mathcal{N} = \{1,1,1,1\}$$

$$[\![\text{sys\_assembly\_cell}]\!]_{Z,\,\mathcal{N}} \equiv [\![(GEN\_A \parallel GEN\_B) \parallel ASSEMBLE]\!]_{Z,\,\mathcal{N}}$$

$$GEN\_A \equiv < r\_robot > . < mov\_in\_m1, 2 > .< \widehat{r\_robot} >. (PROC\_A \parallel GEN\_A)$$

$$GEN\_B \equiv < r\_robot > . < mov\_in\_m2, 2 > .< \widehat{r\_robot} >. (PROC\_B \parallel GEN\_B)$$

$$PROC\_A \equiv < process\_m1, 5 > . < r\_robot > . <| mov\_m1\_m3, 2 > . < \widehat{r\_robot} >.syn\_m1$$

$$PROC\_B \equiv < process\_m2, 7 > . < r\_robot > . < mov\_m2\_m3, 2 > . < \widehat{r\_robot} >.syn\_m2$$

$$ASSEMBLE \equiv (< sync\_1 > \| < sync\_2 >). < process\_m3, 4 > . < r\_robot > . < mov\_m3\_out, 2 > .< \widehat{r\_robot} >. ASSEMBLE$$

Figura 4.11: Especificación de Celda de Ensamblaje

Una vez especificado el sistema y estudiado éste bajo la herramienta BAL, se presentan los resultados obtenidos en la Tabla 4.2. Para ello hemos considerado los casos donde el sistema cuenta con uno o dos robots. Se tomó esta decisión porque observando el sistema se podría pensar que el recurso compartido *Robot* es el causante de un cuello de botella en el sistema y que quizás el rendimiento mejoraría si tuviéramos dos recursos de este tipo. Para estudiar esta suposición, en la especificación cambiaríamos el valor del conjunto N aumentado hasta 2 el número de recursos de tipo *Robot* (N={1,1,1,2}).

Según los resultados que se pueden observar en la Tabla 4.2, se pudo deducir que pasar

	1 Robot	**2 Robots**
Primera pieza	19	17
Intervalo entre piezas	13,06	11,06
Throughput (x100)	7,65	9,04

Tabla 4.2: Resultados con robot rápido en la celda de ensamblaje

	1 Robot	**2 Robots**
Primera pieza	49	38
Intervalo entre piezas	45,18	24,97
Throughput (x100)	2,21	4

Tabla 4.3: Resultados con robot lento en la celda de ensamblaje

de un *robot* a dos no conlleva una mejora notable, obteniéndose un rendimiento similar en ambos casos.

En el mercado existen distintos tipos de *robots* que presentan diferencias en las características técnicas. Veamos que sucede si el robot de nuestro sistema es reemplazado por uno más económico pero que invierte en sus desplazamientos 9 unidades de tiempo. La especificación sería la misma pero se deberá modificar los tiempos empleados por las acciones: mov_in_m1, mov_in_m2, mov_m1_m3, mov_m2_m3, mov_m3_out.

Si comparamos los resultados obtenidos con un robot y dos robots, el coste económico de utilizar dos robots debería ser demasiado elevado para no incorporar otro al sistema de producción ya que el número de piezas que pueden ser procesadas por unidad de tiempo con dos robots es casi el doble que con un solo robot.

Nuestro estudio no aclara si vale la pena en invertir en un robot más rápido o no. Según los resultados obtenidos, podemos observar que el sistema es capaz de procesar 0.075 piezas con el robot rápido y solamente 0.0221 con el robot más lento. La decisión parece estar clara, pero un dato importante es el coste de estos robots. Según el estudio que hemos realizado, la variación de los gastos podría ser considerable, con lo que la decisión quedaría en manos del diseñador.

Otro caso de estudio el cual ha sido sometido bajo el análisis de BAL, es un sistema dedicado a la creación de piezas metálicas, las cuales se conforman de dos partes (A y B). Estas partes se crean mediante la inyección de metal en moldes, para posteriormente ser lijadas, pulidas y ensambladas para conformar la pieza final.

En este sistema disponemos de diferentes tipos de recursos: siete máquinas ($M1 - M7$), dos buffers ($Buf1, Buf2$), cuatro cintas trasportadoras (converyor) ($C1 - C2$) y cuatro robots ($R1 - R4$). Los recursos que modelan a las máquinas se trata de recursos expropiativos, así como, los recursos que modelan a los recursos *robot, buffer* and *conveyor* son recursos

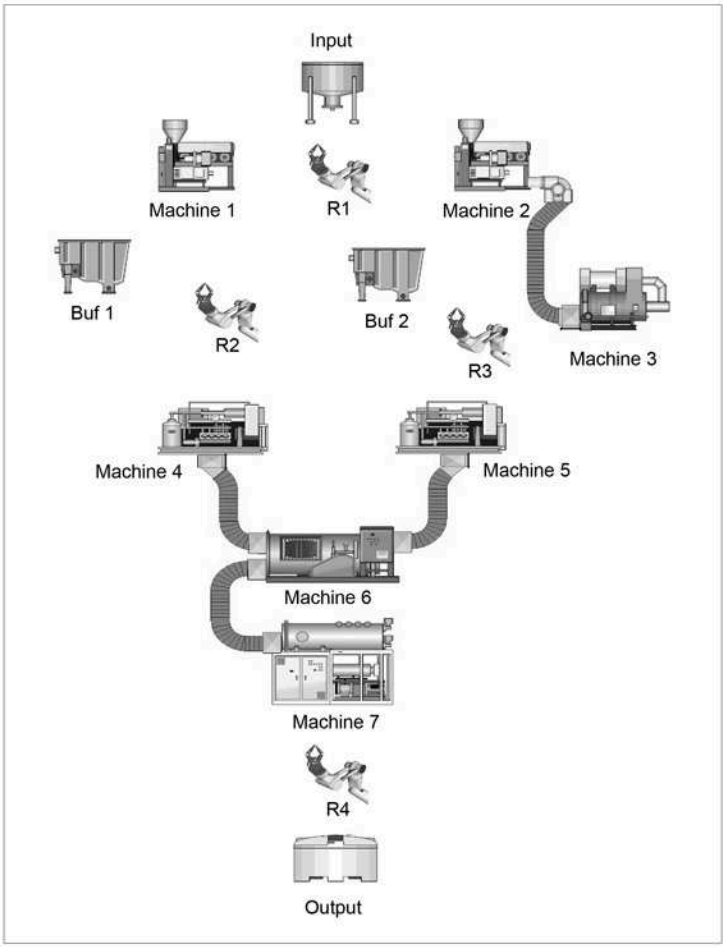

Figura 4.12: Celda de Ensamblaje2

no expropiativos. Para cada uno de ellos se define un conjunto de acciones que necesitan al menos un recurso de este tipo para su ejecución. Para los recursos expropiativos ($M1 - M7$) se definen los siguientes conjuntos:

$$Z_1 = \{\ inject\_A\ \},\ Z_2 = \{\ inject\_B\ \},\ Z_3 = \{\ file\ \},\ Z_4 = \{\ vibrate\ \},\ Z_5 = \{\ polish\ \}$$

$$Z_6 = \{\ assemble\ \},\ Z_7 = \{\ pack\ \}$$

los cuales incluyen las acciones que se ejecutan en cada máquina. Las acciones que utilizan los recursos no expropiativos son:

$$Z_8 = \{\ r\_b1\ \},\ Z_9 = \{\ r\_b2\ \},\ Z_{10} = \{\ r\_c1\ \},\ Z_{11} = \{\ r\_c2\ \},\ Z_{12} = \{\ r\_c3\ \},\ Z_{13} = \{\ r\_c4\ \}$$

$$Z_{14} = \{\ r\_r1\ \},\ Z_{15} = \{\ r\_r2\ \},\ Z_{16} = \{\ r\_r3\ \},\ Z_{17} = \{\ r\_r4\ \}$$

El número de recursos de cada tipo se define como:

$$N=\{2,4,2,2,2,4,2,8,9,12,12,1,1,1,2,2,1\}$$

La especificación BTC para estos sistemas es la que podemos encontrar en la Figura 4.13.

$$Z_1 = \{inject\_A\} \quad Z_5 = \{polish\} \quad Z_8 = \{r\_b1\} \quad Z_{10} = \{r\_c1\} \quad Z_{14} = \{r\_r1\}$$
$$Z_2 = \{inject\_B\} \quad Z_6 = \{assemble\} \quad Z_9 = \{r\_b2\} \quad Z_{11} = \{r\_c2\} \quad Z_{15} = \{r\_r2\}$$
$$Z_3 = \{file\} \quad Z_7 = \{pack\} \quad Z_{12} = \{r\_c3\} \quad Z_{16} = \{r\_r3\}$$
$$Z_4 = \{vibrate\} \quad Z_{13} = \{r\_c4\} \quad Z_{17} = \{r\_r4\}$$

$$\mathcal{N} = \{2, 4, 2, 2, 2, 4, 2, 8, 9, 12, 12, 1, 1, 1, 2, 2, 1\}$$

$$[\![\text{sys\_manu}]\!]_{Z,\ \mathcal{N}} \equiv [\![< new, 2 > .GEN\_A \parallel < new, 2 > .GEN\_B \parallel ASSEMBLE]\!]_{Z,\ \mathcal{N}}$$

$$GEN\_A \equiv < r\_r1 > . < mov\_in\_m1, 2 > .\widehat{< r\_r1 >}.(PROC\_A \parallel < new, 2 > .GEN\_A)$$

$$GEN\_B \equiv < r\_r1 > . < mov\_in\_m2, 2 > .\widehat{< r\_r1 >}.(PROC\_B \parallel < new, 2 > .GEN\_B)$$

$$\begin{aligned} PROC\_A \equiv & < inject\_A, 5 > . < r\_b1 > . < r\_r2 > . < mov\_m1\_b1, 2 > . \\ & \widehat{< r\_r2 >}. < r\_r2 > . < mov\_b1\_m4, 2 > .\widehat{< r\_b1 >}.\widehat{< r\_r2 >}. \\ & < vibrate, 4 > . < r\_c1 > .syn\_A \end{aligned}$$

$$\begin{aligned} PROC\_B \equiv & < inject\_B, 7 > . < r\_c3 > . < file, 3 > .\widehat{< r\_c3 >}. < r\_b2 > . < r\_r3 > . \\ & < mov\_m3\_b2, 2 > .\widehat{< r\_r3 >}. < r\_r3 > . < mov\_b2\_m5, 2 > . \\ & \widehat{< r\_b2 >}.\widehat{< r\_r3 >}. < polish, 8 > . < r\_c2 > .syn\_B \end{aligned}$$

$$\begin{aligned} ASSEMBLE \equiv & (< syn\_A > \| < sync\_B >). < assemble, 3 > . < r\_c4 > . \\ & ASSEMBLE \parallel (\widehat{< r\_c4 >}.(< pack, 7 > . \\ & < r\_r4 > . < mov\_m7\_out, 2 > .\widehat{< r\_r4 >}) \end{aligned}$$

Figura 4.13: Especificación Creación Piezas Metálicas

Tras realizar el análisis de prestaciones bajo la herramienta BAL, podemos utilizar los resultados obtenidos para dirigir nuestro trabajo en dos vías diferentes:

- Encontrar el número mínimo de recursos necesarios para obtener las prestaciones esperadas del sistema.
- Evaluar el comportamiento del sistema para encontrar alguna mejora.

Este ejemplo se ha estudiado para ver si el sistema podría ser optimizado para conseguir un mejor tiempo de producción, es decir, aumentar el *throughput* del sistema.

Diferentes configuraciones (incremento/decremento del número de robots, capacidad de las cintas transportadoras, ...) han sido estudiadas y las conclusiones obtenidas han sido las siguientes: Después de una primera evaluación del sistema el throughput obtenidos fue 0.137, lo cual significa que se obtenían 13.70 piezas en 100 unidades de tiempo. Observando el comportamiento del sistema, se descubrió que el robot $R1$ era un cuello de botella, ya que solamente existía un único robot de este tipo y se debían suministrar piezas a 6 máquinas diferentes ($2M1+4M2$). Tras esta observación, se cambio la configuración del sistema aumentando la cantidad de robots del tipo $R1$ a dos unidades, obteniéndose ahora un throughput de 0.119, siendo este inferior al obtenido en la primera evaluación del sistema. Este problema disminuía al incrementar el número de robots ($R1$) a 4 unidades, pero con este cambio no se obtuvo una mejora significante (throughput 0.14). Este problema surge porque la capacidad de la cinta transportadora $C3$ no está preparada para esta modificación. Cambiando la capacidad (N=$\{2,4,2,2,2,4,2,8,9,12,12,10,1,2,2,2,1\}$) se obtiene una mejora significante en el funcionamiento del sistema, obteniéndose en este caso 23 piezas por cada 100 unidades de tiempo (throughput 0.23). Así que se propuso utilizar 2 robots $R1$ e incrementar la capacidad de la cinta transportadora $C3$ a una capacidad superior a 50 piezas. Otra conclusión obtenida es que con una máquina extra el sistema produce el 20 % más de piezas por unidad de tiempo.

Capítulo 5

Anteproyecto de Tesis

El marco general dentro del cual se pretende desarrollar el trabajo de investigación es *La Automatización del Análisis de Prestaciones de Sistemas Concurrentes*. Debe entenderse que este marco tiene, por una parte, una componente *matemática*, la cual se basará en la construcción de un modelo que nos permita crear descripciones concisas y precisas de sistemas concurrentes, así como una componente *computacional*, la cual nos automatice la tarea del análisis de los sistemas, como podría ser la simulación, verificación, síntesis y análisis de prestaciones.

Basándonos en estas dos componentes, la primera tarea a realizar para la futura línea de investigación, será la creación de una descripción formal de los sistemas basándonos en el álgebra de procesos BTC, álgebra a la cual se extenderá su sintaxis y su semántica operacional. Para ello se analizarán y estudiarán los lenguajes formales de especificación ya existentes, de los cuales se adoptarán ideas y se propondrá una nueva notación para BTC, mediante la cual, haciendo uso de su poder de especificación, se modelarán estos sistemas.

Por tanto, con la construcción de este nuevo formalismo, dispondremos de un nuevo leguaje de especificación, el cual nos proporcionará la capacidad de realizar especificaciones formales de sistemas, de las cuales se podrán realizar un análisis exhaustivo bajo diferentes condiciones y escenarios, además de poder ser verificadas y simuladas.

En el tercer capítulo se presentó la existencia de varios simuladores destinados al estudio de protocolos de WSN, pero cada uno de estos centra sus estudios en un aspecto particular de las redes de sensores. Por ejemplo TOSSIM es un simulador diseñado concretamente para las redes de sensores TinyOS. Además, estos simuladores no son capaces de abstraer ciertas características del sistema como sería deseado, por ejemplo la comprobación de la vivacidad del sistema, la no existencia de interbloqueos (deadlocks) o el análisis de prestaciones de un protocolo de enrrutamiento.

Una alternativa a los tradicionales simuladores es la utilización de los métodos formales y con ellos la utilización de herramientas que faciliten su utilización. Por tanto, una vez finalizada la primera fase de la investigación, nos centraremos en el desarrollo de una herramienta especializada en el modelado de redes inalámbricas de sensores. Para ello se extenderá la

funcionalidad de BAL, incorporándole las siguientes funcionalidades:

- Capacidad de especificación y análisis de aplicaciones y protocolos diseñados para WSN.
- Capacidad para trasladar especificaciones realizadas mediante Redes de Petri a una especificación basada en BTC, así como la capacidad de trasladar especificaciones basadas en BTC a otros modelos formales.
- Se mejorará su interfaz gráfica, permitiendo a los diseñadores visualizar varios indicadores, por ejemplo el consumo individual de cada uno de los sensores, proceso de generación del árbol de enrutamiento, etc.
- Capacidad de simulación, la cual podrá ser guiada por el usuario, o por el contrario la propia herramienta guiará la simulación.
- Escalabilidad.
- Estimación del consumo de energía de cada uno de los nodos que componen la red, como del tiempo de vida de la red.
- Capacidad de modelar los diferentes modos de comunicación entre sensores.

En conclusión, con la extensión del poder de expresión de BTC y la mejora de la funcionalidad de BAL, se pretende conseguir una herramienta con la cual analizar y estudiar las aplicaciones y protocolos diseñados para WSN. Por ejemplo, unos posibles casos de estudio donde se podría aplicar esta herramienta serían: el análisis y evaluación de la escalabilidad de protocolos para WSN, análisis del coste de construcción del árbol de enrrutamiento y el análisis del consumo de energía.

5.1. Objetivo General

El objetivo principal del trabajo futuro de investigación es el de automatizar el análisis de prestaciones en sistemas concurrentes realizado mediante técnicas formales. En concreto se pretende diseñar e implementar una herramienta que realice dicho análisis de prestaciones de una manera eficiente y que presente al mismo tiempo una interfaz amigable cara al usuario. Una vez desarrollada, se pretende analizar sistemas reales, en concreto se realizará un análisis exhaustivo de las redes inalámbricas de sensores las cuales, están proliferando rápidamente en los últimos tiempos. El análisis de prestaciones realizado se basa en técnicas formales, y en concreto para modelar los sistemas se ha optado por el álgebra de procesos BTC debido principalmente al hecho de ser un álgebra temporizada y que al mismo tiempo es capaz de tener en cuenta el estado de los recursos de que dispone el sistema.

5.1.1. Objetivos Específicos

Este objetivo principal anteriormente presentado se descompone en varios objetivos parciales:

- Desarrollo de un analizador sintáctico para las especificaciones de sistemas realizadas mediante el lenguaje BTC.
- Desarrollo de un analizador de prestaciones temporales, en concreto este analizador se centrará en minimizar el tiempo máximo necesario para finalizar la ejecución del sistema a estudio.
- Desarrollo de una interfaz gráfica que además de incluir tanto el analizador sintáctico como el de prestaciones, incluya un editor de texto que posibilite que las especificaciones de los sistemas sean introducidas directamente a través de la herramienta (manteniendo la opción de importarlas de un fichero texto). Este editor de texto debería también incluir asistentes para ayudar al usuario en la introducción de las especificaciones y evitar posibles errores sintácticos.
- Mejorar la eficiencia de los algoritmos usados en el análisis de prestaciones mediante su paralelización.
- Ampliar la capacidad expresiva del álgebra de procesos BTC con el fin de recoger más parámetros de los sistemas a evaluar.
- Aplicación de la herramienta diseñada a sistemas reales.

5.2. Justificación

Los métodos formales poseen un amplio conjunto de técnicas y herramientas ya desarrolladas para verificar protocolos, software y sistemas hardware. Los métodos formales y las herramientas basadas en estos, han sido utilizados para describir a un alto nivel de precisión el comportamiento de ciertos sistemas y así poder analizar y estudiar estos modelos. A través, de este análisis nos ayudará a descubrir errores que no eran descubiertos con las tradicionales pruebas de testeo.

El método más común utilizado en el análisis de protocolos, es la simulación del comportamiento del protocolo. Para ellos los simuladores realizan un gran número de ejecuciones del comportamiento de un mismo sistema, sin embargo no exploran todas las posibles ejecuciones, siendo posible que no realice aquellas donde el comportamiento del protocolo es erróneo.

Creemos que mediante los métodos formales, se puede construir un modelo que sirva de ayuda a los diseñadores en el desarrollo de sistemas orientados a las WSN. Tradicionalmente los protocolos han sido un objeto de análisis de los métodos formales, y los protocolos orientados a WSN no son una excepción. Para ello se deberá de construir un modelo matemático intuitivo y sistemático, mediante el cual se pueda realizar una especificación formal de WSN, siendo

esta el punto de partida del análisis formal, así como la construcción de una herramienta que automatice este proceso.

5.3. Plan de trabajo

A partir de los objetivos parciales que hemos propuesto, vamos a descomponer el trabajo futuro en módulos y tareas a desarrollar. En primer lugar se presentará la relación de los módulos y su descomposición en tareas para seguidamente describir en detalle el alcance de cada una de dichas tareas. Posteriormente se presentará una planificación temporal. Aún así, el grado de concreción de las tareas disminuye a medida que se avanza en el tiempo. Ello se debe en parte a la dependencia de unas tareas de los resultados de otras y, en parte, a la búsqueda de nuevas soluciones; lo que hace muy difícil determinar con precisión el resultado y finalización de dichas tareas. Así pues, la siguiente planificación debe interpretarse como una declaración de intenciones.

Algunas tareas definidas en el plan de trabajo, las cuales han sido presentadas en el capítulo anterior, ya han sido realizadas durante el desarrollo del trabajo de investigación. Durante el desarrollo del trabajo de investigación, algunas tareas se solaparán en el tiempo, iniciándose una antes de haber terminado la anterior. Los módulos y tareas que se proponen para el desarrollo del proyecto de investigación son los siguientes:

Módulo 1 Gestión del trabajo, coordinación y difusión de los resultados

Tarea 1.1 Gestión y coordinación del trabajo a realizar

El trabajo a realizar se ha dividido en cuatro sub-categorías, cada una de ellas constituida por uno de los cuatro grandes aspectos a desarrollar. Estos aspectos o líneas son: implementaciones, modificaciones en el álgebra BTC, aplicación a sistemas reales y difusión de resultados, con lo que el trabajo futuro queda dividido en un total de 6 módulos.

Tarea 1.2 Elaboración de una página Web

Se realizará una página web donde se centralizará toda la información relativa a la herramienta BAL. La página dispondrá de una zona con acceso restringido y una zona pública en la que se podrá acceder a un resumen de los resultados alcanzados y a la versión de la herramienta que en cada momento se disponga. Además, según evolucióne la investigación, en dicha página se ira incorporando los distintos tipos de información asociados a la investigación.

Tarea 1.3 Difusión de resultados

Se dará difusión a los resultados obtenidos a través de los siguientes mecanismos:

- Informes periódicos del trabajo realizado que se harán públicos en la página web del proyecto.
- Publicaciones en revistas y congresos de reconocido prestigio.

Módulo 2 Elaboración del catálogo de funcionalidades de la herramienta a diseñar

El objetivo de este módulo es el de establecer claramente qué va a ser capaz de hacer la herramienta, qué cosas se podrían incluir opcionalmente y qué queda fuera de su ámbito. También se debe hacer un estudio detallado de herramientas similares disponibles en el mercado ya que aunque no existe ninguna con las mismas características que la que se pretende desarrollar, si se puede sacar conclusiones constructivas de las carencias o errores de ellas.

Tarea 2.1 Diseño de las prestaciones de la herramienta

Esta tarea es de vital importancia para proporcionar una visión clara y concreta del objetivo a alcanzar. Las características funcionales que debe tener la herramienta se dividirán en tres categorías:

- Funciones básicas: con las que se pretende tener una primera versión de la herramienta centrada en análisis de prestaciones basadas en parámetros temporales y que además incluirá tanto el analizador sintáctico como el entorno gráfico.
- Funciones avanzadas: donde se incluirán las funcionalidades referidas a evaluación de distintos parámetros de los sistemas concurrentes a estudio (por ejemplo, consumo energético). Con la incorporación de estas funciones avanzadas, se pretende que la herramienta sea también capaz de realizar model-cheking, simulación o verificación a demanda del usuario.
- Funciones opcionales: estas funciones se irán describiendo durante el transcurso del proyecto porque se prevé que irán apareciendo nuevas necesidades.

Tarea 2.2 Aprender de errores

Como se ha comentado anteriormente, en el mercado existen distintas herramientas dedicadas de un modo u otro a realizar distintos tipos de análisis sobre sistemas. Una vez estudiadas en detenimiento y probadas experimentalmente todas las que se encontraban disponibles, se llegó a la conclusión de que no existía ninguna herramienta que cubriera las necesidades que se planteaban. Pero este estudio debe ser productivo, y antes de empezar el desarrollo de la nueva herramienta parece interesante obtener un catálogo con los errores y carencias que se han encontrado en cada una de ellas con el objetivo final de evitarlos dentro de lo posible.

Módulo 3 Diseño y elaboración de la herramienta de evaluación de prestaciones basada en parámetros temporales

En este módulo se pretende desarrollar una primera versión de la herramienta completamente funcional que se centre en las características temporales del sistema a analizar. Es de vital importancia que esta primera versión tenga unos pilares sólidos sobre los que sostener el resto de funcionalidades que se pretenden incluir en ella.

Tarea 3.1 Diseño y elaboración del analizador sintáctico

Esta tarea consiste en desarrollar un analizador sintáctico para las especificaciones de sistemas concurrentes elaboradas con el álgebra de procesos BTC. Se pretende que no sólo detecte errores sintácticos sino que además sitúe al usuario en el error y le muestre sugerencias.

Tarea 3.2 Diseño y elaboración del analizador de prestaciones temporales

En esta tarea se desarrollará un programa que realizará el análisis de prestaciones de sistemas concurrentes. Será necesario proporcionarle como entrada un fichero texto con una especificación sintácticamente correcta de un sistema concurrente realizada con el lenguaje BTC. Para realizar el análisis, basándose en las reglas de la semántica operacional de BTC, el programa deberá generar el grafo de transición de estados correspondiente a las posibles evoluciones del sistema para más tarde recorrerlo y ser capaz de identificar cuál es el camino más corto para alcanzar el estado final desde el estado inicial, esto es, cuál es el tiempo mínimo que el sistema necesita para realizar todas las acciones que en él se encuentran.

Tarea 3.3 Diseño y elaboración del entorno gráfico

Los dos programas elaborados en las tareas anteriores, en su primera versión, deben de utilizarse en línea de comando, proporcionándoles las entradas como ficheros de texto. Aunque las funcionalidades obtenidas hasta este punto son las deseadas, en esta nueva tarea se desarrollará una interfaz gráfica en la que incluir ambos programas y a la que se le añadirán más funcionalidades con el objetivo final de crear un entorno todo lo amigable posible.

Tarea 3.4 Diseño y elaboración del editor de texto

Esta tarea se centra más en mejorar la usabilidad de la herramienta que en ampliar las funcionalidades a nivel de evaluación de prestaciones. Se desarrollará un editor de texto que permita introducir las especificaciones de los sistemas en la misma herramienta sin necesidad de importarlas de un fichero texto externo. Además, se pretende que ayude al usuario en la generación de las especificaciones para lo que se crearán asistentes encargados de especificar los distintos componentes de los sistemas mediante requerimientos al usuario que tendrá que preocuparse poco de la sintaxis exacta utilizada por el álgebra de procesos utilizada.

Tarea 3.5 Realización batería exhaustiva de pruebas

En esta tarea se desarrollará un catálogo de pruebas que cubrirá todos los posibles casos de estudio. Estas pruebas se clasificarán según su objetivo en cuatro grupos claramente diferenciados:

- Comprobar la corrección del software desarrollado.
- Comprobar la completitud de la documentación desarrollada.
- Comprobar cuánto de intuitivo y cómodo tiene la interfaz gráfica desarrollada.
- Hacer un estudio de los tiempos empleados tanto por los algoritmos utilizados para la generación del grafo de transición de estados como para el algoritmo de cálculo del camino más corto.

Módulo 4 Mejora de la eficiencia de la herramienta

El objetivo global de este módulo es el de paralelizar los algoritmos con el propósito de obtener resultados de una manera más eficiente. Debido a los conocimientos y experiencia obtenida tanto en métodos formales como en evaluación de prestaciones, es conocedor de que la primera versión de la herramienta, aunque correcta, presentará limitaciones a la hora de tratar con sistemas de un volumen considerable. Esto será así principalmente en el algoritmo encargado de generar el grafo de transición de estados y en el algoritmo encargado de recorrerlo para encontrar el camino más corto debido a que dependiendo del tamaño del sistema a tratar el número de nodos generados podría ser potencialmente muy elevado lo que repercutiría en las necesidades de memoria del equipo en el que se esté ejecutando y en el tiempo necesario para concluir su ejecución. Por consiguiente, se planteará la paralelización de los algoritmos y la implementación de algoritmos de poda para evitar estos problemas.

Tarea 4.1 Paralelización de los algoritmos

En esta tarea será donde se paralelizarán los algoritmos de generación del grafo de transición de estados y el algoritmo de cálculo del camino más corto. De este modo se irán generando procesos según vaya aumentando el tamaño del grafo, los cuales a su vez se irán descomponiendo también en hilos de ejecución para poder aprovechar al máximo las características de la computación paralela. Evidentemente, esto se realizará teniendo en cuenta las características del equipo disponible para la realización de las pruebas.

Tarea 4.2 Estudio, adaptación e implementación de técnicas de poda

En esta tarea se estudiarán las distintas técnicas de poda de grafos existentes para adaptarlas y aplicarlas al algoritmo de generación del grafo de transición de estados. Con ello se

pretende nuevamente acortar los tiempos de ejecución lo más posible al mismo tiempo que disminuirán los requerimientos en cuanto a necesidad de memoria.

Tarea 4.3 Realización batería de pruebas y estudio comparativo

El objetivo de esta tarea es la de realizar un conjunto de pruebas para comprobar la corrección de las nuevas versiones desarrolladas pero sobre todo realizar estudios comparativos respecto a los tiempos de ejecución y requerimientos de memoria necesitados por las distintas versiones al analizar los mismos sistemas.

Módulo 5 Ampliación de las características analizables de la herramienta

Una vez se disponga de una herramienta que cubra los objetivos principales propuestos, esto es, realizar análisis de prestaciones temporales en sistemas concurrentes, y ésta trabaje de una manera eficiente, se planteará la inclusión de nuevos parámetros del sistema para ser estudiados. Esto requerirá primeramente una adaptación del lenguaje formal de especificación para que sea capaz de recoger estas nuevas características y la consiguiente ampliación de la herramienta para poder llevar a cabo el análisis de prestaciones de manera automática.

Tarea 5.1 Ampliación del álgebra BTC incluyendo datos

Esta tarea se centrará en modificar el formalismo empleado para que sea capaz de recoger distintas propiedades del sistema. Así, el álgebra de procesos necesitará ser modificada sintácticamente para poder incluir esta nueva información y más tarde requerirá que se incluyan y/o modifiquen reglas de su semántica operacional también para adaptarse a las nuevas necesidades.

Tarea 5.2 Modificación de la herramienta para recoger las nuevas características sintácticas de BTC

Una vez realizadas las modificaciones en el formalismo utilizado para recoger las características de los sistemas, éstas deberán ser incluidas en la herramienta. Así, deberá ser modificado el analizador sintáctico y el código encargado de generar el grafo de transición de estados ya que se basa en las reglas de la semántica operacional.

Tarea 5.3 Realización de pruebas de las nuevas características

En esta tarea se realizarán las pruebas pertinentes para comprobar la corrección del nuevo código desarrollado. Además, se pretende modelar sistemas reales a los que podamos tener acceso con el fin de poder comparar los resultados obtenidos experimentalmente con los

resultados obtenidos a través de las simulaciones o el análisis de prestaciones de nuestra herramienta.

Módulo 6 Aplicación de la herramienta a sistemas reales: redes inalámbricas de sensores

Al llegar a este módulo, se contará con una herramienta preparada para ser utilizada en sistemas reales ya que será capaz de recoger distintos tipos de información dependiendo de las características que se deseen estudiar en cada sistema en concreto. Además, será capaz de tratar con sistemas potencialmente grandes debido a las adaptaciones del código realizadas para paralelizarlo y gracias a las técnicas de poda utilizadas. Este módulo se centrará en la aplicación de la herramienta al estudio de sistemas reales, en concreto las redes inalámbricas de sensores las cuales están en pleno auge pero todavía con demasiados puntos por optimizar. Dentro del amplio campo de las redes inalámbricas de sensores, el trabajo se centrará en los tres temas donde parece encontrarse más problemas:

- Protocolos de establecimiento de red
- Localización de los sensores
- Consumo energético

Tarea 6.1 Estudio formal de prestaciones en los protocolos de establecimiento de red

Esta tarea centrará su trabajo en el estudio de distintos protocolos de establecimiento de redes inalámbricas de sensores. Una red de sensores no necesita ninguna infraestructura para poder operar ya que sus nodos pueden trabajar con distintos roles como son: emisor, receptor o enrutador de información. Por lo tanto, al iniciar una red, será necesario un gasto extra (tanto energético como temporal) para establecer dichos roles y las conexiones entre ellos. Además, el consumo producido por estos protocolos se ve incrementado en este tipo de redes debido a que en una red de sensores la topología es cambiante ya que los nodos no siempre están disponibles lo que supone una mayor utilización de dichos protocolos de establecimiento/enrutamiento.

Tarea 6.2 Estudio formal de la correcta localización de los sensores

La finalidad y el medio en el cual serán empleadas las WSN influirán en el despliegue de los sensores. Existen varios factores que determinan el tamaño de la red como el medio sobre el cual se despliega, el esquema de despliegue o la topología de red. El tamaño de la red varía con el medio en el cual la red será desplegada. Así, por ejemplo, en espacios interiores se necesita una menor cantidad de sensores que en espacios exteriores. Por otro lado, al intentar establecer una red pueden aparecer obstáculos que limiten la capacidad de comunicación entre

sensores, con lo que esto afectará a la topología y por tanto en el modo de despliegue de los sensores.

Por otro lado, el tener un número de sensores superior al necesario incrementa considerablemente el consumo de energía: las colisiones conllevan un gasto inútil de energía provocado por la necesidad de retransmisión de paquetes, aumenta el *overhearing* (coste de recibir paquetes que tienen como destino otro dispositivo) y el control de tráfico representa gastos indirectos de mantenimiento a nivel MAC.

Por tanto, en esta tarea lo que se pretende es analizar distintas posibles configuraciones para una WSN con el fin de ser capaz de determinar el número mínimo de sensores necesarios y la mejor localización para ellos.

Tarea 6.3 Análisis del consumo energético de los sensores

En el ámbito de las redes de sensores, el recurso más difícil de conocer es el consumo de energía. Por tanto, el tiempo de vida de la red quizás sea la métrica de evaluación más importante en las redes de sensores. La red solamente podrá desempeñar su tarea mientras esté viva. Por lo tanto, el tiempo de vida de la red indica la máxima utilidad que la red puede proveer. Si esta métrica es analizada previamente a su despliegue, el tiempo de vida estimado puede contribuir a la justificación del coste del despliegue de la red. Por tanto, el tiempo de vida es considerado como un parámetro fundamental en el contexto de disponibilidad y seguridad en redes, y depende del tiempo de vida de cada uno de los sensores que constituyen la red. Si el consumo de energía de cada uno de los sensores no se predice con exactitud, es posible que el tiempo de vida para la red predicho se desvié de manera incontrolada. Así, el modelo que representa a cada uno de los nodos (sensores) juega un papel importante en la predicción de dicha métrica, campo donde puede ser de gran utilidad el estudio bajo métodos formales.

En conclusión, se puede decir que el tiempo de vida de la red se considera como uno de los parámetros mas importante en la evaluación de las redes de sensores y de los algoritmos desarrollados para este tipo de redes y será el objetivo de esta tarea, realizar ese análisis mediante la herramienta diseñada.

5.4. Planificación temporal

La duración prevista para cada una de las tareas detalladas en la sección anterior es de tres años y en una primera aproximación y considerando el trabajo ya realizado, se podrían planificar según muestra en la Tabla 5.1.

	1^{er} Año				2^{do} Año				3^{er} Año			
Tarea	Tr 1	Tr 2	Tr 3	Tr 4	Tr 1	Tr 2	Tr 3	Tr 4	Tr 1	Tr 2	Tr 3	Tr 4
1.1												
1.2												
1.3												
2.1												
2.2												
3.1												
3.2												
3.3												
3.4												
3.5												
4.1												
4.2												
4.3												
5.1												
5.2												
5.3												
6.1												
6.2												
6.3												
	Trabajo realizado				Trabajo por realizar							

Tabla 5.1: Planificación Temporal

Referencias

[1] The network simulator. http://www.isi.edu/nsnam/ns/.

[2] Prism. http://www.prismmodelchecker.org/.

[3] Probe - csp animator. http://www.fsel.com/software.html/.

[4] Real academia española (rae). http://www.rae.es.

[5] Spin - model checker. http://spinroot.com/spin/whatisspin.html/.

[6] Tina- time petri net analyzer. http://www.laas.fr/tina/.

[7] Tossim. http://www.eecs.berkeley.edu/ pal/research/tossim.html.

[8] Uppaal cora. http://www.cs.aau.dk/ behrmann/cora/.

[9] Uppaal tiga. http://www.cs.aau.dk/ adavid/tiga/.

[10] Uppaal tron. http://www.cs.aau.dk/ marius/tron/.

[11] Soheib Baarir, Marco Beccuti, Davide Cerotti, Massimiliano De Piero, Sussanna Donatelli, and Giuliana Franceschinis. The GreatSPN tool: recent enhancements. 2009.

[12] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The palladio component model for model-driven performance prediction. 2009.

[13] G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. 3185, 2004.

[14] V. Bos and J. Kleijn. Formal Specification and Analysis of Industrial Systems. 2002.

[15] A. boulis, A. Fehnker, M. Fruth, and A. McIver. Ca Vi- simulation and Model Checking for Wireless Sensor Networks.

[16] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: a model-checking tool for real-time systems. page 1427, 1998.

[17] Phillip J. Brooke and Richard F. Paige. Lazy exploration and checking of CSP models with CSPSim. pages 33–49, 2007.

[18] G. Clark, S. Gilmore, and J. Hillston. The PEPA performance modelling tools. 1999.

[19] Sinem Coleri, Mustafa Ergen, and Tak-Kuen John Koo. Lifetime analysis of a sensor network with hybrid automata modelling, 2002.

[20] T.H. Cormen. Introduction to Algorithms. 2003.

[21] A. A. Desrochers and R. Y. Al-Jaar. Applications of Petri Nets in Manufacturing Systems Modeling, Control and Performance Analysis. IEEE Press, 1995.

[22] I. Dietrich and F.Dressler. On the lifetime of Wireless Sensor Networks.

[23] Nicholas J. Dingle, William J. Knottenbelt, and Tamas Suto. PIPE2: a tool for the performance evaluation of generalised stochastic petri nets. 2009.

[24] J.S. Dong, Jing Sun, Jun Sun, K. Taguchi, and X. Zhang. Specifying and Verifying Sensor Networks: and Experiment of Formal Methods. 2005.

[25] J. Ezpeleta and J. Colom. Automatic synthesis of Colored Petri Nets for the control of FMS. 1997.

[26] Mercedes E.Ñarciso Farias and M. Angle Piera i Eroles. Entorno para la planificación de la producción de sistemas de fabricación flexibles.

[27] A. Fehnker and P. Gao. Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols.

[28] Ansgar Fehnker, Lodewijk F.W van Hoesel, and Angelika H. Mader. Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. pages 253–272, 2007.

[29] M. Fruth. Probabilistic Model Checking of Contention Resolution in the IEEE 802.15.4 Low-Rate Wireless Personal Area Network Protocol. 2006.

[30] Sergio Gallina and Flavio Fama. Modelado de Celdas de Producción Flexible con Redes de Petri y Raise.

[31] A. Gill. Introduction to the Theory of Finite-state Machines. 1962.

[32] S. Gilmore, J. Hillston, R. Holton, and M. Rettelbach. Specifications in Stochastic Process Algebra for a Robot Control Problem.

[33] J.F. groote, J.J.A. Keiren, A.H.J. Mathijssen, B. Ploeger, F.P.M. Stapers, C. Tankink, Y.S. Usenko, M.J. van Weerdenburg, J.W. Wesseling, T.A.C Willemse, and J. van der Wulp. The mcrl2 toolset. 2008.

[34] A. Hinton, M. kwiatkowska, G.Ñorman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. 3920:441–444, 2006.

[35] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[36] Khan and Misic. Security in IEEE 802.15.4 cluster based networks. 2008.

[37] J. Kleijn, M. A. Reniers, and J. Rooda. A Modeling Method for Flexible Manufacturing Systems based on Colored Petri Nets. 22:249–282, 2003.

[38] J. Kleijn, M. A. Reniers, and J. Rooda. Analysis of an industrial system. 22:249–282, 2003.

[39] N. Kothari, T. Millstein, and R. Govindan. Deriving State Machines from TinyOS Programs Using Symbolic Execution. pages 271–282, 2008.

[40] D. Kozen. Quantitative MuCalculus Analysis of Power management in Wireless Networks. 27:333–354, 1983.

[41] D. Kozen. results on the propositioinal mu calculus. 27:333–354, 1983.

[42] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic Model Checking and Power-Aware Computing. pages 6–9, 2005.

[43] L. Lamport. The temporal logic of actions. 1994.

[44] Jonathan Lawrence. Practical Application of CSP and FDR to Software Design. 3525:151–174, 2005.

[45] P. Loucopoulos, B. Theodoulidis, and D. Pantazis. Business rules modelling: conceptual modelling and onject-oriented specifications. pages 323–42, 1991.

[46] Y. Luo and J.J.P. Tsai. A graphical simulation system for modeling and analysis of sensor networks. 2005.

[47] P.C. Ölveczky and S. Thorvaldsen. Formal Modeling and Analysis of Wireless Sensor Network Algorithms in Real-Time Maude. 2006.

[48] K.L. Man, T. Krilavicius, Th. Valee, and an H.L Leung. TEPAWSN: A formal Analysis Tool for Wireless Sensor Networks. 1.

[49] A.K. McIver and A. Fehnker. Formal Techniques for the Analysis of Wireless Networks.

[50] S.Ñair and R. Cardell-Oliver. Formal specification and analysis of performance variation in sensor network diffusion protocols. 2004.

[51] D. Perez, M.C. Ruiz, J.J. Pardo, and D. Cazorla. Automatic Performance Evaluation. 2009.

[52] D. Perez, M.C. Ruiz, J.J. Pardo, and D. Cazorla. BAL Tool in Flexible Manufacturing Systems. 2010.

[53] E. Perla, A. Cathain, R.S. Carbajo, M. Huggard, and C.Mc. Glodrick. POWERTOSSIMz: Realistic Energy modelling for Wirelles Sensor Network Environment. 51:921–960, 2007.

[54] H. Pham, D. Pediaditakis, and A. Boulis. From simulation to real deployments in WSN and back. 2007.

[55] M. C. Ruiz, D. Cazorla, F. Cuartero, J. J. Pardo, and H. Macía. A bounded true concurrency process algebra for performance evaluation. LNCS 3236, 2004.

[56] M.C. Ruiz, D. Perez, J.J. Pardo, and D. Cazorla. BAL Tool: what else? . 2010.

[57] S. Skiena and M.A. Revilla. Programming Challenges. 2003.

[58] S. Tripakis and C. courcoubetis. Extending PROMELA and SPING for real-time. pages 329–348, 1996.

[59] Armin Zimmermann, Knut Dalkowski, and Günter Hommel. A case study in modeling and performance evaluation of manufacturing systems using colored petri nets. 1996.