REDUCCIÓN DE LA COMPLEJIDAD EN EL CONTROL DE SISTEMAS A EVENTOS DISCRETOS MODELADOS MEDIANTE AUTÓMATAS JERÁRQUICOS DE ESTADOS FINITOS

Autor: Rafael Rivas Estrada
Tutor: Dr. Edgar Chacón Ramírez

Trabajo de Grado presentado ante la ilustre Universidad de Los Andes como requisito final para optar al grado de

MAGISTER SCIENTIAE EN COMPUTACIÓN

UNIVERSIDAD DE LOS ANDES
CONSEJO DE ESTUDIOS DE POSTGRADO
FACULTAD DE INGENIERÍA

Enero 2002

a Patricia

RESUMEN

REDUCCIÓN DE LA COMPLEJIDAD EN EL CONTROL DE SISTEMAS A EVENTOS DISCRETOS MODELADOS MEDIANTE AUTÓMATAS DE ESTADO JERÁRQUICO

Palabras clave: *Integración de Sistemas, Autómatas Finitos, Automatización Industrial.*

El presente trabajo describe el estudio y diseño de una herramienta computacional que permite reducir la complejidad en el estudio de los sistemas a eventos discretos. Para lograr el objetivo propuesto es necesario visualizar globalmente los aspectos teóricos involucrados en el tema de investigación. Esta visualización se inicia con un estudio de las técnicas de la automatización industrial, clasificación y enfoques. Al concluir el estudio sobre automatización industrial, se prosigue con una revisión sobre las características y marcos de referencia para el análisis de los sistemas industriales, se estudian las diferentes formalidades para el modelado de sistemas dinámicos a eventos discretos, logrando una clasificación de acuerdo a los niveles de abstracción deseados y del enfoque a utilizar. Una vez elegido el formalismo de los autómatas de estado se estudian los avances en el área de autómatas jerárquicos y las alternativas de su implementación. Teniendo la base teórica se procede al desarrollo de la herramienta de software, presentando los requerimientos, diseño y observaciones sobre el producto, por último se concluye con las ideas que sientan las bases para investigaciones futuras.

Tabla de Contenidos

Introducción

Una de las principales líneas de investigación del Laboratorio de Sistemas Discretos y Automatización Industrial de la U.L.A. (LaSDAI), es el estudio de la automatización integral de complejos industriales, utilizando como herramienta la integración de procesos[11][12][13][14]. Ésta aparece como una técnica para lograr el control del sistema formado por componentes de producción, planificación y gerencia. Cada una de estas partes poseen una dinámica propia y muchas veces incompatibles desde un punto de vista operacional e informático, cuyo elemento de integración usualmente es el hombre. Una aproximación para lograr un mejor entendimiento y manipulación del proceso industrial como un todo es jerarquizar las actividades utilizando modelos de referencia, como los propuestos por la ISO [18]. Se pueden resumir el números de niveles o capas de estas jerarquías de modelos piramidales en tres : Planificación, Optimización/Supervisión y Control Directo tal y como se muestra en la figura 1

Figura 1. Modelo de 3 capas para la integración de procesos

Para cada uno de estos tres niveles, se pueden sintetizar modelos que posean una correlación directa con el sistema, una de las representaciones de estos son los Autómatas Jerárquicos de Estados (AJE) que se adaptan fácilmente al modelo de capas, donde cada una de estos autómatas toma acciones que pueden afectar a los autómatas de su mismo nivel o de un nivel diferente, donde el autómata del nivel superior debe ser capaz de reflejar el comportamiento global del sistema. El trabajo propuesto se centra en el estudio y desarrollo de una herramienta de software, compuesto por múltiples procesos en ejecución interconectados que implementen un Autómata Jerárquico de Estado, aportando un mecanismo que permita reducir la complejidad del control de sistema a eventos discretos.

Introducción general al tema de investigación

La automatización integral de un proceso productivo requiere la integración de múltiples subsistemas, que van desde el control directo de la producción hasta el área gerencial y de planificación. Un conjunto integrado por estos subsistemas

puede ser visto como una organización jerárquica con forma de pirámide, tal como se observa en la figura 2. Para lograr la integración, en necesario resolver los problemas de incompatibilidad desde la base de la pirámide, donde generalmente se encuentran sistemas SCADA (*Supervisory Control and Data Adquisition*) y DCS (*Distributed Control Systems*).

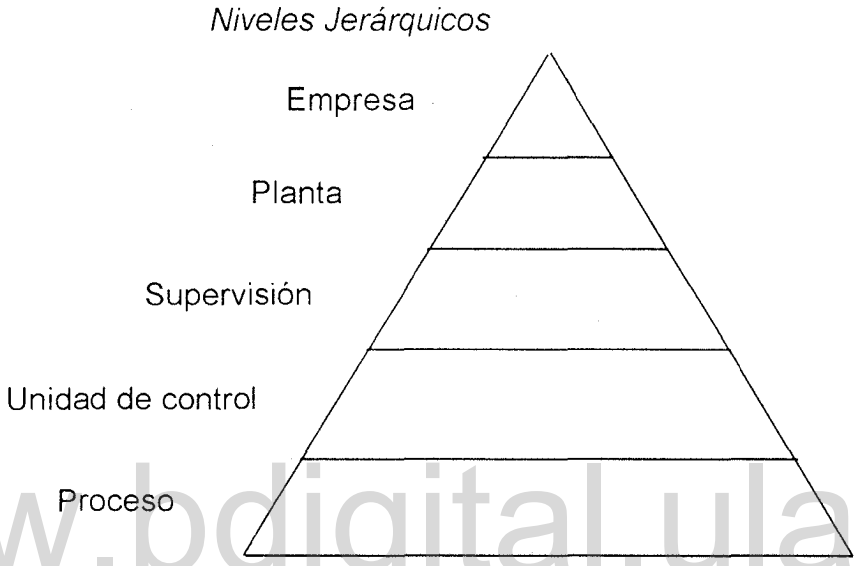

Figura 2. Pirámide de integración

Los sistemas SCADA se ocupan de labores de supervisión y de recolección de datos de un proceso remoto. Ello implica la existencia de varias RTU's (*Remote Terminal Units*) [14] que "observan" el proceso y que se comunican con una estación maestra. La existencia de múltiples protocolos específicos de fabricante (*proprietary*) para las RTU's asociadas a los SCADA's crea problemas de incompatibilidad que limitan o incluso imposibilitan lograr la integración deseada, ver figura 3.

La integración de Sistemas Automatizados (DCS's, SCADA's) es uno de los temas principales que interesa actualmente a la industria. La razón de esto es la necesidad de coordinar diferentes unidades funcionales[1] desde una sala de control y con la menor intervención posible de un operador[13]. Además, se presenta la necesidad de enviar información específica a los niveles superiores con el fin de hacer optimización y planificación de los procesos.

[1] Se define como unidad funcional a una planta o equivalente.

En el laboratorio de Sistemas Discretos y Automatización Industrial (LaSDAI), se ha venido desarrollando un estudio para la integración de sistemas ampliamente distribuidos en industrias complejas[1]. Esta propuesta persigue desarrollar un marco teórico y un prototipo de integración que permita interactuar a las unidades funcionales, entre ellas (comunicación horizontal cooperante) y con los niveles superiores (envío y recepción de eventos y consignas), además, lograr de alcanzar una representación con diferentes niveles de abstracción de la dinámica del sistema. Así mismo plantea la construcción de sistemas decisorios, de apoyo totalmente automáticos, para la implantación de supervisores de eventos discretos y el desarrollo de estrategias para el almacenamiento de los objetos que representan a las diferentes unidades funcionales.

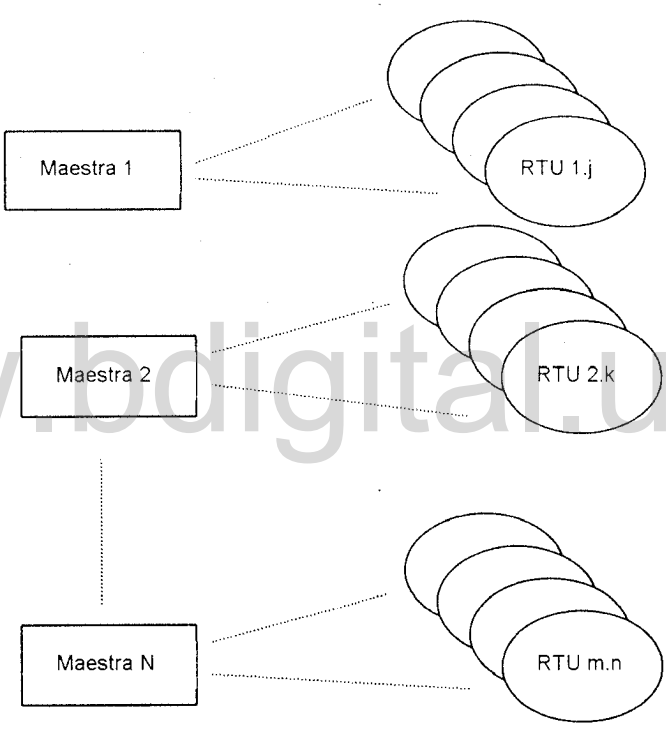

Figura 3. Grupos de Mestras y RTU para diferentes protocolos

Un proceso industrial automatizado se compone de dos grandes partes: aquella que realiza el procesamiento físico del material (usualmente a lo largo de un sistema de transporte de alguna naturaleza, en el caso de un sistema de manufactura) y una red de información y decisión que integra los elementos que están involucrados.

En la actualidad, los procesos industriales están sufriendo cambios generados por la introducción de elementos altamente flexibles (computadores, microprocesadores) en la interconexión de los componentes electrónicos

involucrados en el proceso, entre estos se pueden mencionar los sensores inteligentes con capacidades de trasmisión de información en niveles de detalles superiores. Los sensores actuales ofrecidos en el mercado pueden informar sobre su propio desempeño, condiciones de los enlaces de comunicaciones y en algunos casos, están en capacidad de trabajar en forma cooperativa con sus sensores pares. Esta capacidad de interconexión abre una posibilidad que había estado tradicionalmente fuera del ámbito de la automatización, a saber, la posibilidad de monitorizar, coordinar, controlar y dirigir el proceso como un todo en lugar de sólo una suma de partes más o menos automatizadas, esto es lo que se ha denominado islas de automatización.

La tendencia actual en automatización industrial es integrar los subsistemas que automatizan partes del proceso total, es decir, se busca integrar los subsistemas de gerencia y planificación con los de coordinación general, y éstos a su vez con los de control de producción, hoy en día existen productos comerciales que implementan estas ideas como lo son Gensym[2], rsi-resolution para nombrar algunos.

Formulación del problema

En muchos ambientes de modelado de sistemas a eventos discretos, las transiciones de estado y los eventos asociados a estos, constituyen los componentes básicos para la construcción de modelos. Los diagramas de autómatas de estados finitos y las tablas de transiciones son los mecanismos más simples para recolectar estos componentes en un todo. Estos modelos son conceptualmente interesantes, debido a su simplicidad inherente y por el hecho de que ellos pueden ser formalmente descritos por autómatas de estados finitos y su comportamiento descrito mediante lenguajes formales.

Desde los primeros trabajos en teoría de control para Sistemas Dinámicos a Eventos Discretos [6] (DEDS por sus siglas en inglés) propuestos por Ramadge y Wonham en 1987 [35][36][37] , y los que se han llevado a cabo posteriormente [23][38], los sistemas a eventos discretos han sido modelados de una manera exitosa en el marco de los autómatas de estados finitos. Paralelamente, ha surgido una cantidad de trabajos importantes en el área de la teoría de control conjuntamente con un sector afín a las ciencias de la computación [29] [11][12][13][24][28].

Una revisión de estos trabajos permite concluir que actualmente ésta es un área en total desarrollo, donde se pueden realizar aproximaciones que reduzcan la complejidad de los modelos obtenidos mediante la equivalencia de autómatas,

[2] Ver en Internet la página de Gensym Corporation (Burlington, Massachusetts) http://www.gensym.com

permitiendo que los autómatas de estados finitos de mayor jerarquía puedan no sólo reflejar el estado del sistema, sino también permitir "cerrar" el lazo de control, por medio de la disgregación de consignas emitidas por el autómata principal, llegando hasta los niveles inferiores de la jerarquía del sistema.

Antecedentes o trabajos previos existentes en la literatura del área vinculados directamente con el problema

Entre los trabajos desarrollados que tienen una vinculación con el problema a tratar se pueden mencionar los desarrollados por Wonham sobre Control Supervisorio de DEDS [35], Zamaï, Chaillet-Subias y M. Combacau sobre Supervisión Jerárquica sobre DEDS [15], los presentados por Wong y Wonham sobre Control Jerárquico [26] y en los modelos de automatización integral de complejos industriales [11][12]. Las fuentes de los trabajos han surgido en el área de teoría de control, ingeniería industrial, ingeniería eléctrica, e ingeniería informática. También es importante resaltar el rol de los Sistemas Híbridos en los Procesos de Control[30], estos sistemas permiten la unión de la componente continua de los sistemas con sus modelos, representados en la componente discreta, y en algunos casos involucrando la creación de una interfaz que permita la interacción de ambos componentes.

Objetivo del trabajo de investigación

El objetivo principal de este trabajo de investigación es el desarrollo componentes de software que representen autómatas de estados jerárquicos que permitan reducir la complejidad al manipular sistemas de estados con un gran número de estados. Estos modelos deben ser capaces de mostrar la dinámica de un complejo industrial a diferentes niveles de abstracción. Así como se puede mostrar el estado general del sistema se podrán observar los efectos de la propagación correcta de las consignas de producción desde los niveles superiores del complejo industrial (Gerencia y planificación) hasta los niveles de control directo.

Justificación de la relevancia del trabajo de investigación

Al lograr los objetivos planteados, se dispondrá de una herramienta que permitirá cerrar los lazos de control de los complejos industriales desde los niveles superiores de los modelos de pirámide de integración. Este será un aporte importante en el área de integración de sistemas industriales complejos, ya que se reduce el número de estados de los autómatas y la necesidad de la gerencia media de conocer a fondo los detalles de estos tipos de sistemas.

Estructura del trabajo

(verificar nuevo esquema)

El presente trabajo se divide en 5 capítulos, cada capítulo toca un área de interés de del trabajo desarrollado. En el Capítulo I trata sobre la Automatización de los procesos Industriales, se introduce el tema con una clasificación de los Sistemas Dinámicos y luego se describe el modelo CIM y la pirámide de integración.

En el Capítulo II se estudia los Sistemas a Eventos Discretos, se describen y luego se estudian los diferentes formalismos para su modelaje. En el Capítulo III, es una continuación del Capitulo anterior, pero se concentra en el estudio de los Autómatas Jerárquicos de Estados Finitos.

En el Capitulo IV, se presentan los conceptos básicos para el diseño y construcción de los Autómatas Jerárquicos de Estados Finitos Jerárquicos por medio de programación orientada a objetos. En el Capitulo 5 se presenta el diseño en si de la herramienta y sus características principales, luego se finaliza con la Conclusión, Recomendaciones y la Bibliografía consultada.

Capítulo 1: Automatización de Procesos Industriales

1.1. Introducción

En los últimos años hemos sido testigos de la evolución y modernización de los procesos que intervienen en la planificación, diseño, producción y modernización de los productos que consume nuestra sociedad. El factor común en toda esta revolución ha sido el computador.

Gracias a esta herramienta, se ha podido estudiar en un tiempo aceptable la mayoría de los escenarios diferentes, obtenidos a partir del estudio de las diferentes teorías de mercado, estudio de perfiles de consumidor, niveles de inventario y hasta de proyecciones climatológicas. También se ha podido observar el avance de las herramientas que permiten a los creativos plasmar de una manera agradable pero exacta sus diseños. Hoy en día, cada vez más mesas de dibujo de los diseñadores son sustituidas por computadores condicionados como estaciones de diseño con altas prestaciones gráficas, y con una infinidad de conjuntos de herramientas que soportan totalmente la labor del diseñador, integrando el diseño y la elaboración de productos y herramientas[20].

Al igual que en las áreas mencionadas, la tarea de producción también se ha beneficiado con esta revolución computacional, permitiendo cada vez más que dispositivos mecánicos-electrónicos con capacidades para transmitir y recibir información se integren al proceso productivo. Estos procesos se han divido en dos grandes grupos: los que están en capacidad de medir, estimar, o convertir magnitudes físicas como temperaturas, flujos, niveles, etc., en datos digitales, los cuales al ser transmitidos a los centros de recolección de datos (sensores de mayor jerarquía) permiten descubrir o estimar el estado del proceso de producción, a estos dispositivos se les conoce como sensores. A la par de los sensores se encuentran los actuadores, cuya característica fundamental es la de poder aceptar consignas en formas de mensajes, éstas son a su vez transmitidas a actuadores de menor jerarquía, hasta llegar a los niveles más cercanos al proceso físico de producción. El avance antes mencionado comienza a apartar al trabajador cada vez más del proceso físico, en el que sólo se dedicaba a tomar lectura de medidores analógicos con posible errores en la apreciación de la medida por razones de índole físico, falta de rigurosidad en la periodicidad de las muestras, cansancio, olvido, temor, etc..

1.2. Clasificación de los sistemas dinámicos

Antes de iniciar el estudio de los principales enfoques para llevar a cabo la integración de sistemas industriales, se debe tener conocimiento de los diferentes tipos de maquinarias y equipos que conforman los sistemas dinámicos encontrados en la industria, que van desde el extremo de los sistemas totalmente discretos, integrados por computadores, robots y redes de comunicaciones y descritos en su mayoría a través de los Sistemas a Eventos Discretos, hasta llegar al otro extremo donde se tienen torres de destilación, mezcladores, bombas,

válvulas y tuberías presentes en la mayoría de los sistemas continuos, estos últimos descritos mediante sistemas de ecuaciones diferenciales o ecuaciones en diferencias. Como una primera aproximación, a continuación se muestra una clasificación en base a la naturaleza de los procesos que conforman los diferentes sistemas de producción:

- Sistemas de Producción basados en procesos de Manufactura:
  Se caracterizan fundamentalmente por el diseño, construcción, ensamblaje de piezas, como los encontrados en las líneas de ensamblaje de autos, computadores, equipos electrodomésticos, y otros.

- Sistemas de Producción basados en Procesamiento por Lotes:
  Las industrias que caen en esta categoría se caracterizan por la producción de bienes en intervalos de tiempos finitos. Esto se consigue por medio de la adecuación de la planta para obtener un solo producto a través de la mezcla de la materia prima y la aplicación de energía. Una vez que se logra la cantidad del producto deseado, o la permitida por la materia prima, es posible adecuar la planta para la producción de otro producto. Ejemplos de este tipos de industria son las plantas de procesamiento de productos alimenticios como salsas, cereales, panaderías, industrias metalúrgicas como SIDOR, ALCASA, etc..

- Sistemas de Producción basados en Procesos Continuos:
  Se caracterizan fundamentalmente por la operación continua para la obtención de los productos intermedios y finales. En este tipo de industrias, las paradas de producción no programadas pueden acarrear alta perdidas y, en algunos casos, necesidad de sustituir algunos subsistemas que conforman la planta, como por ejemplo sistemas de transporte de materia (tuberías) . Entre las industrias pertenecientes a estas categorías se tienen refinerías, oleoductos, y sistemas de tratamiento de aguas potable.

Los sistemas de producción están compuestos por subsistemas semi-autónomos que trabajan de manera cooperativa. Esta forma de operación implica un intercambio de información permanente, compuesta por medidas de valores y consignas de operación que permiten la operación del sistema como un todo. La cantidad y el tipo de los diferentes subsistemas incide en la complejidad del sistema total. Es esa complejidad la que se desea reducir, transformando los subsistemas semi-autónomos en subsistemas autónomos donde la intervención del hombre sea mínima. Una vez lograda la transformación antes mencionada es necesario definir el control global del sistema. Una posible aproximación es definir un modelo de todo el sistema, donde cada uno de los subsistemas sea tratado como una caja negra cuyo comportamiento es totalmente conocido. A continuación se muestran algunas aproximaciones para el modelado de sistemas discretos y sistemas híbridos.

1.3. El modelo CIM (Computer Integrated Manufacturing)

La introducción del microprocesador en la industria ha dado inicio a una nueva revolución. Las máquinas herramientas han pasado de cero o pocos componentes electrónicos a herramientas con capacidad para mantener dentro de sí uno o más microprocesadores. A medida que una herramienta aumenta la cantidad de componentes electrónicos, se independiza más de la energía física de la persona que la manipula. Esto ha traído como consecuencia la necesidad de diseñar y construir herramientas con capacidad de operar de manera automática, delegando en el operador la labor de supervisión directa. Hoy en día existen herramientas que son supervisadas y programadas por computadores. Ante este ambiente de computadoras controlando y programando herramientas programables, y en otros casos, computadoras asistiendo en el diseño (herramientas CAD) y en la construcción de piezas (herramientas CAM), se obtienen ambientes industriales con una alta cantidad de computadoras y redes de comunicación. Estás últimas están en capacidad de unir los diferentes computadores que se encuentran en una planta de producción con las herramientas automatizadas.

Para lograr la automatización global de una industria, es imprescindible lograr la integración total de todos los componentes automatizados disponibles en la instalaciones de la industria. La Organización Internacional de Normas, ISO. por sus siglas en Inglés, ha establecido un modelo de referencia para lograr tal integración. Este es un modelo por capas o niveles, donde cada capa es responsable de una función especifica y se comunica con la capa inferior para obtener su estado y enviarle consignas de operación. Dicho modelo recibe el nombre de Modelo para la Fabricación Integrada por medio del Computador (Computer Integrated Manufacturing - CIM.) [7] [25], cuya arquitectura se muestra en la figura 4. Además del diseño por capas, el modelo se caracteriza por que a medida que trabaja en las capas inferiores, la precisión es mayor y el nivel de procesamiento de información es menor. A medida que se trabaja en las capas superiores, la precisión disminuye y la cantidad y abstracción de la información aumenta.

Fábrica
Unidad de producción
Celda de trabajo
Estación de trabajo
Equipo

Figura 4 Arquitectura del Modelo CIM

A continuación se describen cada uno de los niveles del Modelo CIM:

- Nivel de equipo: Es el nivel inferior en la arquitectura CIM. Consiste de los controladores para recursos individuales, tales como un robot, y los equipos sensores y actuadores asociados al mismo.

- Nivel de estación de trabajo: Una estación de trabajo consiste de uno o más equipos que efectúan una actividad única. Se consideran en este nivel un robot, una máquina herramienta, una unidad de transporte o almacenamiento. Una estación de trabajo recibe órdenes de la coordinación de la celda de ensamblaje (nivel inmediato superior) y ejecuta tareas o secuencias de operaciones asignadas a la celda de ensamblaje.

- Celda de trabajo: Cada celda de trabajo contiene los sistemas de control para el secuenciamiento de las estaciones de trabajo asignadas a la misma. Maneja el secuenciamiento del material a través de las diferentes estaciones de trabajo.

- Unidades de producción: En este nivel se maneja la coordinación de los recursos y tareas necesarias en cada unidad de producción. Agrupa varias celdas de ensamblaje y coordina y distribuye las actividades entre las diferentes celdas.

- Fábrica: Es el nivel más alto del modelo, y los procesos incluidos en este nivel incluyen los procesos de planificación, gerencia de producción, incluyendo la planificación a largo plazo. Está asociado a los aspectos financieros y las demás funciones administrativas.

1.4. La Pirámide de automatización

Del mismo modo como la ISO creó el modelo CIM para integración de procesos en industrias de manufactura, también presentó su propuesta para la industria de

procesos continuos. En este último caso, la propuesta recibe el nombre de Pirámide de Integración, (ver figura 5) y su fin es normalizar el proceso de integración de las diferentes entidades presentes en la industria de procesos continuos. Este enfoque se caracteriza por la división jerárquica de los procesos presentes en la planificación, diseño, producción y comercialización de bienes. Esta división por niveles permite definir interfaces entre niveles, donde cada nivel se caracteriza por el tipo de información y la funcionalidad. A continuación se describen cada uno de los diferentes niveles:

- Nivel de Proceso:

  Se refiere a los instrumentos (sensores y actuadores) que están en contacto directo con los procesos físicos.

- Unidad de Control:

  Los elementos que realizan el control de los procesos trabajan de acuerdo a una parametrización recibida desde el nivel supervisorio. Estos sistemas actúan de manera automática, manteniendo el control regulatorio y/o de secuencias de los procesos productivos. Mediante la retroalimentación del proceso.

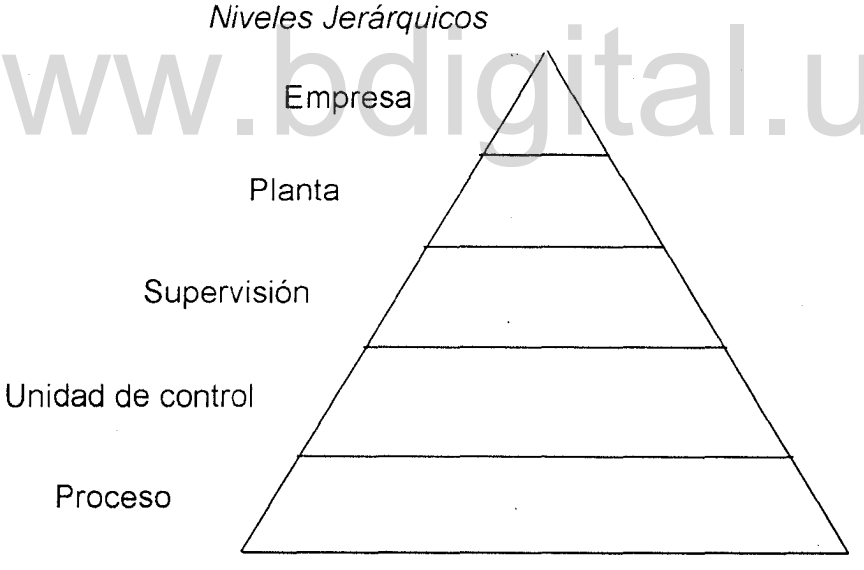

Figura 5. Pirámide de automatización

- Nivel de Supervisión:

  El control supervisorio tiene como función la coordinación de las diferentes unidades de procesamiento o transformación de material, mediante la parametrización de los controladores. A nivel de supervisión, los operadores de la planta controlan el proceso. La fijación de los parámetros

de operación es en la mayoría de los casos establecida por el operador, aunque puede ser apoyada por sistemas en línea.

- Nivel de Planta:
  Es la responsable del logro de metas de producción fijadas a la planta, mediante el manejo y coordinación de los recursos (económicos, humanos y equipos) para la ejecución de diferentes actividades. Funciones como el establecimiento de parámetros y criterios de producción son realizadas en este nivel. Los resultados obtenidos de las funciones realizadas a nivel de planta , son enviadas al nivel de Empresa. El nivel de planta coordina las diferentes unidades de procesamiento bajo su responsabilidad. Labores como optimización y planificación de las tareas de producción se realizan a este nivel. El monto de información necesaria para estas labores es muy alto. Sin embargo, el tiempo promedio de separación entre la ejecución de estas actividades varía entre los días y las semanas.

- Nivel de Empresa:
  Al igual que el más alto nivel de CIM, a nivel de empresa, el sistema comprende los problemas de gerencia de producción, establecimiento de estrategias y niveles de producción, que están asociadas a políticas globales de la empresa, el factor financiero y el mercado. A este nivel se coordinan las actividades entre las diferentes plantas o unidades completas de producción. El nivel de información manejada es muy grande. Esta información proviene de las plantas y de los factores externos a la empresa. El establecimiento de estrategias y niveles de producción tienen una larga duración ya que involucran funciones de planificación.

1.5. Elementos de automatización

Una vez definidos los modelos de referencia para la integración de procesos en una arquitectura de automatización integral, es pertinente conocer la naturaleza de los elementos a integrar, los cuales se clasifican en elementos físicos y elementos lógicos. La justificación de la presencia de estos dependerá exclusivamente de la naturaleza de la producción.

Los elementos físicos son:

- Sistemas de almacenamiento: Corresponden a los dispositivos y lugares donde se almacena la materia prima, productos intermedios, productos finales.
- Elementos de transporte: Movilizan materiales desde un depósito o lugar de transformación a otro punto. Ejemplos de elementos de transporte son las correas transportadoras, tuberías, líneas de transmisión, etc..
- Sistema de sensores y actuadores: Son dispositivos que se encuentran en contacto directo con el proceso físico, por medio de los cuales se adquiere la

información necesaria para la determinación del estado del proceso y de la modificación del mismo mediante la modificación de sus entradas.

- Sistemas de procesamiento de información e interfaz humano–computador: La información sobre el proceso controlado debe ser almacenada, procesada y mostrada a los operadores mediante equipos orientados al control y a la interfaz con el operador. El computador pasa a ser un elemento fundamental en la automatización, puesto que es el soporte de los sistemas lógicos que controlan el proceso y permiten la transferencia de información que asegura la posibilidad de una coordinación final.

Los elementos lógicos descritos a continuación se caracterizan por ser sistemas programados, protocolos de comunicación, sistemas de interrogación de dispositivos y bases de datos.

Sistemas programados para control y supervisión:
Están encargados de manipular los datos de mayor precisión como lo son los valores capturados por los sensores, y a su vez transmitir hasta los actuadores físicos los comandos básicos para realizar operaciones directas sobre el proceso. Como ejemplos de este tipo de órdenes se pueden citar el apagar o encender una bomba, aumentar o disminuir unos grados la temperatura de una caldera, etc..

- Sistemas de control de producción:
  Son los encargados de recolectar la información de los diferentes sistemas de programados para el control y la supervisión, con el fin de obtener el estado global del proceso de producción. Además, están en capacidad de recomendar o asignar los diferentes puntos de operación para los procesos controlados por los sistemas descritos en el párrafo anterior.

- Sistema de optimización y secuenciamiento de la producción:
  Soportan, planifican y optimizan el proceso de producción global. Por manipular un nivel abstracción mayor, utilizan una gran cantidad de información cuya manipulación requiere herramientas de computación avanzadas. Además, están capacitados para el soporte del diseño de la producción total.

Como puede observarse en la discusión anterior, existe una gran cantidad de elementos físicos y lógicos que intervienen directamente en el proceso de producción. La integración de todos esos componentes es fundamental para lograr una producción automatizada adecuada.

El enfoque por niveles permite manejar la complejidad de la integración. Pero aun así se debe modelar cada uno de los niveles funcionales, para definir cual es la información requerida de los niveles inferiores y por los niveles superiores de cada

uno de los niveles, es decir, hay que lograr definir exactamente cual es la interfaz entre niveles de cada uno de dichos enfoques.

1.4 Posibilidades de evolución en la Automatización de Procesos Industriales

La investigación en la electrónica, software, herramientas, han permitido visualizar una nueva generación de conceptos y herramientas en el área de la Automatización Industrial [20], entre los elementos que permitirán nuevos desarrollos en el futuro tenemos:

- Sistemas de manufactura inteligente. Sistemas que utilizan los conocimientos obtenidos a partir de entes basados en inteligencia artificial para reconocer situaciones, para visualizar e implementar estrategias para esas situaciones y para extender su propias bases de conocimiento a partir del manejo de situaciones parcialmente desconocidas.
- Tecnologías avanzadas para el manejo de bases de datos, que permitirá construir nuevos sistemas distribuidos con la capacidad de integrar totalmente la gerencia de los datos sobre la totalidad del sistema. Está nueva tecnología permitirá tener un único u *holístico* flujo de datos que describirá totalmente el ciclo de vida de la elaboración de productos, permitiendo a la vez una mayor descentralización del manejo de datos hacia los subsistemas locales.
- Programación de alto nivel, usando nuevos lenguajes de programación de sistemas para hacer la tarea de programación más sencilla, manejable y con una mejor relación costo efectividad.
- Arquitecturas de sistemas abiertos, habilitan sistemas y subsistemas de diferente estructuras y composición que pueden comunicarse transparentemente los unos con los otros de una manera implícita, sin necesitar módulos de comunicación especiales.
- Computación gráfica, con interfaz Humano-Computador embebida en todos los niveles de la comunicación entre el hombre y la máquina, permitiendo la solución a muchos de los problemas sufridos por operadores por largas horas de inmovilidad o posturas incomodas ante un computador.
- Máquinas morfológicas, que pueden ajustar su configuración morfológica para ajustarse a los requerimientos de la manufactura de un producto.
- Nuevas tecnologías en las máquinas, que incluyen el uso de laser para moldeado de piezas, plasma, rayos de electrones y otras técnicas.
- Ensamblaje totalmente automatizado usando robots "inteligentes".

Capitulo 2: Sistemas Dinámicos a Eventos Discretos

2.1. Introducción

Un DEDS [6] es un sistema en el que las variables dinámicas sufren cambios en instantes de tiempos discretos asíncronos debido a la ocurrencia de cambios cualitativos discretos en el sistema, llamados eventos. Muchos sistemas hechos por el hombre, como por ejemplo las redes de computadoras, sistemas de manufactura, sistemas de tráfico, protocolos de comunicación, pueden ser modelados por medio de los DEDS. Para una introducción multimedios interactiva a los sistemas a eventos discretos se recomienda visitar la página en Internet http://vita.bu.edu/cgc/MIDEDS. La dinámica de estos sistemas evoluciona de acuerdo a la ocurrencia de eventos. Ejemplos de eventos son las llegadas de mensajes en una red de computadoras, la ejecución completa de una tarea en un sistema de manufactura, la transmisión de una señal de reconocimiento en un protocolo de comunicación, etc..

Los DEDS difieren de los Sistemas Dinámicos de Variables Continuas (CVDS por sus siglas en Inglés) en dos aspectos principales. Primeramente, los estados de un DEDS pueden tomar valores lógicos o simbólicos en lugar de valores numéricos. Segundo, estos estados cambian con la ocurrencia de eventos, los cuales pueden ser descritos de una manera no numérica, en un instante de tiempo asíncrono. Los CVDS tienen vigencia dentro del dominio continuo del tiempo (por ejemplo, un proceso químico) o dentro del dominio de tiempo discreto (por ejemplo, un tomador de muestras) dependiendo si la entrada se hace de manera continua o si se aplica en instantes fijos de tiempo [6][40].

2.2 El concepto de Evento

Al igual que el termino "sistema", el termino "evento" no es formalmente definido en si. Éste es un concepto primitivo descrito de una manera intuitiva. Pero se debe estar enfatizar en pensar que un evento ocurre instantáneamente y posiblemente cause la transición de un estado a otro.

Un evento puede ser relacionado con la ejecución de una orden especifica (presionar un botón, por ejemplo). También puede ser visto como la ocurrencia espontánea no controlada, como por ejemplo la llegada de un paquete a una tarjeta de red, recibir un mensaje de correo, una falla no programada en el servicio de energía eléctrica, etc.

2.3. Modelos para CVDS

Un modelo para cualquier sistema dinámico consiste de una base de tiempo, entradas, estados, salidas y funciones para determinar el próximo estado y la salida dependiendo del estado actual y las entradas. La evolución de las variables continuas en un CVDS está descrita por una ecuación diferencial, mientras que la

evolución de un CVDS en tiempo discreto es descrito por medio de una ecuación en diferencias.

2.4. Modelos para DEDS

La descripción matemática de las características dinámicas de un sistema se denomina modelo matemático. El primer paso en el análisis de un sistema dinámico es elaborar su modelo. Los modelos pueden tomar muchas formas distintas, según el sistema particular de que se trate y las circunstancias. Un vez obtenido el modelo de un sistema se pueden usar diversas herramientas analíticas y computacionales con el objeto de su adecuado análisis y síntesis.

La importancia de modelos para los DEDS se debe a la necesidad de estudiar la dinámica de los sistemas creados por el hombre, y para ofrecer una teoría que permita comprender los controladores basados en computadoras para ejecutar tareas supervisoras o de alto nivel de inteligencia en DEDS o CVDS El marco del modelaje para un CVDS, basado en ecuaciones diferenciales, han sido bien establecidas [27], ya que muchos de los sistemas ya sean mecánicos, eléctricos, térmicos, hidráulicos, económicos, biológicos, etc., pueden ser caracterizados por ecuaciones diferenciales. Paralelamente se ha realizado un gran esfuerzo en para poder modelar DEDS, pero no se ha logrado obtener un modelo de consenso análogo a los existentes en los CVDS.

Una simple vía para modelar un DEDS es realizar una simulación del sistema. Esta simulación puede describir el comportamiento del sistema de una manera bastante exacta, aunque éste no sea muy útil para propósitos analíticos. Un modelo muy útil puede ser una caracterización matemática del sistema, que puede proveer una idea sobre su comportamiento, y además puede ser analizado mediante herramientas matemáticas. El registro de sucesivas ocurrencias de cambios cualitativos, junto con los tiempos de ocurrencia de esos cambios, es llamado la traza del DEDS. La diferencia fundamental entre la traza y el lenguaje aceptado radica en la necesidad de registrar los tiempos de ocurrencias. La teoría de trazas de sistemas ha sido de gran utilidad para describir el comportamiento de DEDS [43].

2.5. Modelos Lógicos, Temporales y Estocásticos

Los modelos de DEDS pueden ser clasificados en tres grupos dependiendo del nivel de abstracción en la representación matemática de la traza del sistema [40], estos son el nivel lógico, nivel temporal y el nivel estocástico.

2.5.1. Nivel Lógico

En el nivel más abstracto o nivel lógico, una traza es simplemente una secuencia de eventos listados según el orden de la ocurrencia. La colección de todas las

posibles trazas de un DEDS en el nivel lógico es llamado también el lenguaje generado por el sistema. Entonces, en el nivel lógico, una traza de un DEDS puede ser descrita por la secuencia:

$$s = \sigma_1\ \sigma_2\ \sigma_3\ \ldots;\ \text{donde s es la traza y } \sigma_i \text{ es el evento i-esimo}$$

Mientras la traza del sistema es usada para describir su comportamiento, varios formalismos son usados para generar el sistema de trazas. Algunos de los formalismos propuestos para generar trazas en el nivel lógico son: Autómatas de Estados Finitos[3], Redes de Petri, Máquinas de Turing, Autómatas de Estados Extendidos, Lenguajes de Programación tales como los Procesos Secuenciales Comunicantes o el Cálculo de Sistemas Comunicantes, etc..

2.5.2. Nivel Temporal

En el nivel temporal, una traza puede ser descrita por la secuencia de pares:

$$s = (\sigma_1, t_1)\ (\sigma_2, t_2)\ (\sigma_3, t_3)\ \ldots$$

donde $\sigma_1\ \sigma_2\ \sigma_3\ \ldots$ es la traza lógica del sistema y t_i, para i = 1,2, ..., es el tiempo en el cual ocurrió el evento.

Algunos de los formalismos usados para la generación de trazas temporales de los sistemas son: la Redes de Petri Temporizadas, Grafos de Flujos de Datos, Lógica Temporal, Lógica en Tiempo Real, Álgebra Mini-Max, etc.

2.5.3. Nivel Estocástico

En el nivel estocástico, el comportamiento del sistema es modelado como la secuencia de pares de variables aleatorias, de una manera simple, este puede ser descrito como:

$$s(\omega) = (\sigma_1(\omega), t_1(\omega))\ (\sigma_2(\omega), t_2(\omega))\ (\sigma_3(\omega), t_3(\omega))\ \ldots$$

donde σ_i representa los eventos la traza lógica del sistema en función de ω y t_i el tiempo esperado de ocurrencia en función de ω, para i = 1,2, ...

Algunos de los formalismos usados para generar trazas estocásticas de un sistemas son: Procesos de Markov, Redes de Petri Estocásticas, Redes de Colas, etc..

2.6. Modelos Transicionales y Algebraicos

[3] Máquinas de Estados Finitos

Los modelos DEDS en cada uno de los niveles de abstracción –lógico, temporal y estocástico- pueden ser también clasificados en modelos transicionales y modelos algebraicos dependiendo si el modelo representa al sistema de una manera implícita o explicita. El comportamiento de un sistema en ambos representaciones está descrita por el conjunto de trazas y el conjunto de trayectorias del sistema.

2.6.1. Modelos Transicionales

Los modelos transicionales representan los DEDS de una manera más explicita. Estos consisten de un conjunto de estados, un alfabeto de entrada o conjunto de eventos, y un conjunto de reglas de producción correspondientes a las posibles transiciones entre estados del sistema.

Ejemplos de modelos transicionales incluyen los Autómatas de Estados Finitos, Redes de Petri, Máquinas de Turing, etc., en el nivel lógico; Grafos de Eventos Temporizados, Redes de Petri Temporizadas, Grafos de Flujos de Datos, etc., en el nivel temporal; Cadenas de Markov, Redes de Colas, etc., en el nivel estocástico.

2.6.2. Modelos Algebraicos

Los modelos algebraicos representan los DEDS de una manera más implícita. Los sistemas son definidos de una manera axiomática o lógica, y varios operadores algebraicos son usados para describir los aspectos dinámicos del sistema. Cada modelo algebraico está caracterizado por una sintaxis y una semántica, donde la sintaxis corresponde al conjunto de símbolos y operadores usados para representar el sistema, en cambio la semántica describe el significado de esos símbolos y operadores.

2.7. Comparación de Modelos y Aplicaciones

Los formalismos usados para modelar DEDS en diferentes niveles de una abstracción de la traza tienen diferentes valores descriptivos. Se puede decir que un formalismo "A" tiene más poder descriptivo que un formalismo "B", si cada modelo de B tiene un modelo de A equivalente. Dos modelos son equivalentes si describen el mismo conjunto de trazas.

Por las definiciones antes vistas, las descripciones dadas por modelos del nivel lógico tienen menos poder descriptivo que los modelos del nivel temporal, y estos, a su vez, tienen menos poder descriptivos que los modelos pertenecientes al nivel estocástico. Sin embargo, si el interés se enfoca en representar de una manera más exacta el comportamiento de la traza, el formalismo algunas veces simplifica

la descripción de los aspectos individuales. Por lo tanto, los aspectos lógicos de las trazas de un modelo temporal pueden tener una estructura más simple que la que permite el modelo lógico por sí mismo. Por esta razón, se debe tener cuidado al comparar los formalismos entre los diferentes niveles en términos de su poder descriptivo. Los formalismos en el mismo nivel, sin embargo, pueden ser comparados fácilmente. El formalismo de las Redes de Petri, por ejemplo, tiene más poder descriptivo que el formalismo de las Autómatas de Estados Finitos en el nivel de lógico. Los modelos de DEDS exhiben un patrón jerárquico basado en su poder descriptivo, tal como se muestra en la figura 6, donde se muestran varios formalismos ordenados verticalmente según su nivel de abstracción, estando el nivel lógico en la parte superior, en el nivel medio los pertenecientes al nivel Lógico temporal, y en la parte inferior los pertenecientes al nivel Estocástico.

<table>
<tr><td>Autómatas de Estados Finitos, Redes de Petri</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>Autómatas de Estados Finitos Extensibles</td></tr>
<tr><td>Modelos de lenguajes de programación.</td><td></td><td>Procesos de Markov</td></tr>
<tr><td>Todas las trazas posibles (no temporizadas)</td><td>Lógica Temporal</td><td>Semi-Procesos de Markov</td></tr>
<tr><td colspan="2">Todas las trazas posibles temporizadas</td><td>Semi-Procesos Generalizados de Markov</td></tr>
<tr><td colspan="3">Simulaciones de la colección de todas las trazas, determinísticas o estocásticas, temporizadas o no</td></tr>
</table>

Figura 6. Jerarquía de Modelos de DEDS

La elección de el tipo de formalismos para describir un DEDS depende mucho de la clase de aplicación en la cual se está interesado. Por ejemplo, un modelo del nivel de abstracción lógica puede revelar si el sistema es libre de abrazos mortales (deadlock) o no. Un modelo del nivel de abstracción temporal puede ser analizado para determinar la tasa máxima de salida de un sistema de un sistema de manufactura; un modelo del nivel de abstracción estocástico puede mostrar las distribuciones que describen el comportamiento de un sistema de inventario.

Los modelos lógicos han sido usados para diseñar algoritmos de control de transmisión en una red de comunicación; los modelos basados en lenguajes de programación han sido usados para estudiar las propiedades de seguridad y estudios de alcanzar de estados finales[3]. Los modelos temporales han sido utilizados para acelerar el cálculo del computo en algoritmos de procesamiento de señales; por otra parte los modelos estocásticos han sido usados para determinar el desempeño, confiabilidad, y disponibilidad de un sistema de computación; los modelos de redes de colas han sido utilizados para análisis de perturbación de

sistemas, etc[40]. Con una variedad de modelos disponibles para describir el comportamiento de los DEDS, la mayoría basados sobre aplicaciones de interés, la necesidad de un marco de trabajo unificado para la caracterización es obvia.

2.8. Control de DEDS

La teoría matemática tradicionalmente se ha ocupado del modelado, diseño y control de los CVDS descrito por medio de ecuaciones diferenciales. En cambio, el desarrollo de técnicas de control para los DEDS apenas recientemente ha comenzado.

Un objetivo de control usualmente es especificado como las restricciones necesarias para que un sistema tenga un comportamiento deseado. El objetivo es construir un controlador e interconectarlo de alguna manera a un sistema dado, para que el comportamiento del sistema en lazo cerrado siga unas especificaciones preestablecidas. La invención de un controlador que asegure que el sistema controlado cumpla con un comportamiento deseado a menudo lleva a evaluar los compromisos entre el desempeño del sistema controlado y el costo de aplicar la política de control asociada [42].

En el control clásico de los CVDS, por ejemplo, el objetivo del control es especificado por estudios de estabilidad, repuestas de frecuencia del sistema y el controlador es diseñado para minimizar el error entre el comportamiento deseado y el comportamiento controlado. En el control estocástico de CVDS, los objetivos del control son especificados como una medida del comportamiento promedio del sistema, y el controlador es diseñado de una manera tal que permita que el sistema en lazo cerrado tenga una medida de probabilidad diferente sobre un conjunto de trayectorias posibles del sistema en lazo abierto, permitiendo que el sistema en lazo cerrado satisfaga la medida del comportamiento promedio del sistema. En este punto hay que diferenciar dos tipos de eventos, aquellos eventos cuya ocurrencia es totalmente independiente, son llamados eventos no controlables, ya que es imposible iniciar o inhibir su aparición, en cambio aquellos que podemos programar o detener su ocurrencia son los llamados eventos controlables. Por medio de la inhibición o inserción de estos eventos en los momentos apropiados será posible garantizar un comportamiento deseado [5]

Para los DEDS en el nivel lógico, los objetivos de control son especificados como un conjunto de propiedades cualitativas, por ejemplo, invarianza, seguridad, restricción del comportamiento sobre un lenguaje predefinido, etc.. Los controladores son diseñados para asegurar que el comportamiento en lazo cerrado cumpla con las especificaciones de la menor manera restrictiva.

Los modelos lógicos, algebraicos o transicionales, pueden ser usados para representar un DEDS, con el fin de sintetizar un controlador que pueda lograr que

el sistema tenga las propiedades cualitativas deseadas cuando éste opere en lazo cerrado. El control es aplicado de dos posibles maneras:

- Inhabilitando algunos de los eventos que pueden ocurrir.
- Forzando la ocurrencia de algunos eventos.

Si el objetivo del control depende no sólo del orden en el cual ocurren los eventos, sino que también es importante el tiempo en que puede ocurrir, los modelos temporales de eventos discretos son usados para analizar los sistemas. Una función objetivo puede elegir una secuencia óptima en un sistema flexible de manufactura para maximizar la tasa de producción.

En el caso de DEDS en el nivel estocástico, los controladores son diseñados para alcanzar un comportamiento promedio, por ejemplo, caudal promedio de producción, retardo promedio, etc., en lazo cerrado.

2.9. Modelos para plantas y supervisores.

Los DEDS que son controlados, son llamados plantas [40], las cuales son modeladas por medio de Autómatas de Estados Finitos Deterministas (AFD). Sea P una planta, ésta es representada como la quíntupla:

$$P \underset{=}{^{def}} (X, \Sigma, \alpha, x_0, X_m)$$

donde X denota el conjunto de estados de la planta, Σ denota el conjunto finito de eventos; $\alpha : \Sigma \times X \rightarrow X$ es la función de transición de estados, $x_0 \in X$ es el estado inicial de la planta; y $X_m \subseteq X$ es el conjunto de estados deseables o seguros (estados marcados) de la planta.

Un supervisor para una planta dada es otro DEDS modelado como una Máquina de Estados Finitos. Sea S un supervisor, éste es representado por la quíntupla:

$$P \underset{=}{^{def}} (Y, \Sigma, \beta, y_0, Y_m)$$

donde Y denota el conjunto de estados del supervisor; Σ es el conjunto finito de eventos; $\beta : \Sigma \times Y \rightarrow Y$ es la función de transición de estados; y $Y_m \subseteq Y$ es el conjunto de estados marcados del supervisor.

2.10. Control por composición síncrona

El supervisor se ejecuta de manera síncrona con la planta logrando de esta manera el control de la planta. La realización de la composición síncrona de la planta y el supervisor es denotada por la máquina de estados P □ S, donde "□" representa el operador de composición síncrona. La composición síncrona de la

planta y el supervisor significa que sólo están permitidas aquellas transiciones presentes tanto en la planta P y el supervisor S, entonces, si una de las transiciones no son permitidas en el supervisor, ésta no podrá ser ejecutada en la planta, y de esta manera es como el supervisor controla la planta. Formalmente P □ S es otra máquina de estado determinista, representada por la quíntupla:

$$P \square S \underset{=}{\overset{def}{}} (Z, \Sigma, \gamma, z_0, Z_m)$$

donde $Z = X \times Y$ es el conjunto de estados de la máquina de estados obtenida por la composición; Σ es el conjunto finitos de eventos que aceptan S y P, $\gamma : \Sigma \times Z \rightarrow Z$ es la correspondencia de estados para P □ S. Sea $\sigma \in \Sigma$ y $(x, y) \in X \times Y = Z$, se puede definir

$$\gamma(\sigma,(x,y)) \underset{=}{\overset{def}{}} \begin{cases} (\alpha(\sigma,x),\beta(\sigma,y)) & \text{si } \alpha(\sigma,x)! \text{ y } \beta(\sigma,y)! \\ indefinido & \text{en cualquier otro caso} \end{cases}$$

$z_0 = (x_0, y_0)$ denota el estado inicial de la máquina de estados finitos P □ S; y $Z_m = X_m \times Y_m$ denota los estados marcados de P □ S.

Todas las ideas mostradas en este capítulo son la base para el estudio de la controlabilidad de los sistemas dinámicos a eventos discretos, y por lo tanto, el eje de este trabajo. En los siguientes capítulos se propone la construcción de un sistema programado que permita ser la base para la implementación de las ideas propuestas en esta parte.

Capitulo 3: Autómatas Jerárquicos de Estado

3.1. Introducción

En el capítulo anterior se mencionaron las principales técnicas para modelar DEDS, una de éstas, el paradigma de autómatas de estados finitos o autómatas finitos (AF), cuya estructura básica son las transiciones entre estados y los eventos asociados a éstas. Los diagramas de estados con la representación de las transiciones son el mecanismo más simple para describir el comportamiento de un DEDS. Estos modelos son conceptualmente atractivos debido a su simplicidad y al hecho de que estos pueden ser formalmente descritos por un AF y su comportamiento puede ser descrito por medio de lenguajes formales.

Un autómata de estados finitos es un modelo matemático de un sistema que posee entradas y salidas discretas. El sistema puede tener un número finito de configuraciones y sólo puede estar en una de éstas en un instante de tiempo. A cada configuración se le conoce como *estado*. El estado de un sistema puede resumir la información concerniente a entradas pasadas que son necesarias para determinar el comportamiento del sistema ante subsecuentes entradas. El mecanismo de control de un ascensor es un buen ejemplo de los sistemas de estados finitos. El mecanismo no debe recordar todos las solicitudes de servicios satisfechas para aceptar una nueva solicitud de servicio, este sólo debe conocer en qué piso se encuentra actualmente, la dirección del movimiento (subiendo o bajando) y la colección de solicitudes de servicio no satisfechas.

En el área de las ciencias de computación existen muchos ejemplos de AF, y la teoría de AF es usada en muchos casos como una herramienta para la construcción de modelos. Un primer ejemplo es un circuito de conmutación, como el usado para encender la unidad de control de una computadora. A circuito de conmutación está compuesto por un número finito de compuertas lógicas, cada una de las cuales puede estar en una de dos condiciones, usualmente denotadas como 0 ó 1. Estas condiciones pueden, en términos electrónicos, ser dos niveles de voltaje a la salida de una compuerta. El estado en una red de conmutación con n compuertas será entonces uno de los 2^n estados obtenidos a partir de todas las configuraciones posibles de las n compuertas. Aunque el voltaje en cada compuerta puede pertenecer a un conjunto infinito de valores, la circuitería electrónica está diseñada para que sólo dos valores correspondientes a 0 y a 1 sean establecidos, y todos los posibles valores restantes serán automáticamente ajustados a 0 ó 1. Los circuitos electrónicos de conmutación son diseñados de esta manera para que puedan ser vistos como un sistema de estados finitos, aunque separando el diseño lógico de una computadora de la implementación electrónica.

Es de observar que algunos programas de soporte para compiladores como los editores de texto y los analizadores de sintaxis sobre ciertos lenguajes son a menudo diseñados como autómatas de estados finitos. Por ejemplo, un analizador de léxico busca símbolos en un programa fuente escrito en un lenguaje de

programación de alto nivel, para localizar las cadenas de caracteres correspondientes a identificadores, constantes numéricas, palabras reservadas, operadores, etc.. En este proceso el analizador de léxico necesita recordar sólo un monto finito de información, tal como la longitud de un prefijo de una palabra reservada.

La computadora también puede ser vista como un sistema de estados finitos. Teóricamente, el estado del procesador central, la memoria principal y los dispositivos de almacenamiento secundario son a veces uno de un conjunto finito muy grande de estados. Se puede asumir que un número fijo de discos, cintas, que están disponibles y que la memoria no se extiende infinitamente[21]. El ver una computadora como una sistema de estados finitos puede no ser muy real o atractivo desde el punto de vista matemático, ya que establece un límite a la memoria o al número de dispositivos disponibles, pero es capaz de capturar la esencia real de la operación de la computadora. Para capturar totalmente la esencia de la computadora, es necesario pensar que se tiene una memoria potencialmente infinita, aunque en cada computadora físicamente construida, la capacidad es memoria es finita.

El cerebro humano también puede ser visto como un sistema de estados finitos, el número de células en el cerebro o neuronas es también finito, probablemente 2^{35} o más. Cada una de las neuronas puede ser descrita por medio de un grupo de bits, por lo tanto, la teoría de AF puede ser usada para describir el comportamiento de cerebro humano. Sin embargo, el número de estados es tan grande que esta aproximación no proporciona resultados atractivos y útiles sobre el comportamiento del cerebro humano.

La principal razón para el estudio de los sistemas de estado finitos es la naturaleza del concepto, que indica que un sistema puede evolucionar entre diferentes estados, esta posibilidad de describir un sistema en base a estados es atractiva ya que puede ser capaz de captar la noción característica de ese sistema.

3.2. Autómatas de estados finitos

Un AF consiste de un conjunto finito de estados y un conjunto de transiciones entre estados, que ocurren ante la entrada de símbolos pertenecientes a un alfabeto Σ. Por cada símbolo de entrada hay exactamente una transición de salida del estado (es posible que la transición lleve al mismo estado de origen). Todo AF posee un estado del conjunto que se conoce como el estado inicial, en el cual es el estado en el que se encuentra el AF cuando comienza a operar. Algunos estados son designados como finales o estados marcados.

Un grafo dirigido, llamado *diagrama de transición*, está asociado con el AF de la manera siguiente:

- Los vértices del grafo corresponden a los estados del AF.
- Si existe una transición desde un estado *q* a un estado *p* ante una entrada *a*, existe un arco etiquetado con *a* desde el estado *q* hasta el estado *p* en el diagrama de transición.
- El AF acepta una cadena *x* si la secuencia de transiciones correspondientes a los símbolos de *x* llevan desde el estado inicial hasta un estado final o estado de aceptación.

Formalmente se puede denotar un AF por una quíntupla $(Q, \Sigma, \delta, q_0, F)$, donde Q es el conjunto de estados finitos; Σ es el alfabeto finito de símbolos de entrada; δ es la función de transición que ejecuta la correspondencia entre $Q \times \Sigma \rightarrow Q$, por lo tanto, $\delta(q,a) = p$ si existe una transición desde un estado *q* a un estado *p* ante una entrada *a*; $q_0 \in Q$ es el estado inicial; y $F \subseteq Q$ es el conjunto de estados finales o estados de aceptación.

Es importante hacer las siguientes observaciones:

- Los nombres "máquina de estados" y "generador" son también usadas para hacer referencia a los autómatas de estados finitos.
- El autómata se dice que es determinístico porque δ es una función sobre $Q \times \Sigma$. En contraste, la estructura de transiciones de un autómata no determinístico es definida por medio de la relación sobre $Q \times E \times \Sigma$, donde E es el evento nulo
- La selección de los estados marcados es una decisión al construir el modelo que dependerá del interés de la aplicación para la que se construye el autómata.

Ejemplo de un Autómata de Estados Finitos:

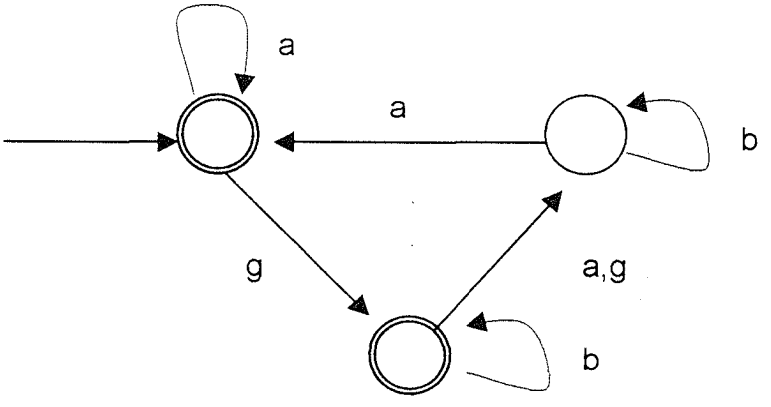

Figura 7. Ejemplo de un Autómata de Estado Finito

Observando el dígrafo de la figura 7, se asume que los nodos representan estados y los arcos etiquetados representan transiciones entre esos estados. Este dígrafo provee una completa descripción de un Autómata de Estados Finitos. El conjunto de los nodos es el conjunto de los estados del autómata, Q = { *x, y, z* }. El conjunto de las etiquetas de las transiciones es el conjunto de eventos (alfabeto) del autómata, Σ = { *a, b, g* }. Los arcos en el gráfico proveen una representación gráfica de la función de transición del autómata, la cual se denota como

$$\delta: Q \times \Sigma \rightarrow Q$$

$\delta(x, a) = x$	$\delta(x, g) = z$
$\delta(y, a) = x$	$\delta(y, b) = y$
$\delta(z, b) = z$	$\delta(z, a) = \delta(z, g) = y$

La notación $\delta(y, a) = x$ significa que si el autómata está en el estado *y*, entonces ante la ocurrencia del evento *a*, el autómata deberá hacer una transición instantánea hacia el estado *x*. La causa de la ocurrencia del evento *a* es irrelevante, el evento puede ser una entrada externa a el sistema modelado por el autómata, o puede ser un evento espontáneamente generado por el sistema modelado por el autómata.

Hay tres observaciones que son importantes resaltar:

- Un evento puede ocurrir sin que se produzca un cambio de estado, como en $\delta(x, a) = x$.
- Dos eventos distintos pueden ocurrir en un estado dado causando exactamente la misma transición, $\delta(z, a) = \delta(z, g) = y$. Este hecho es interesante ya que no se puede distinguir entre el evento *a* y el evento *g* únicamente observando la transición entre el estado *z* hacia el estado *y*.
- La función δ es una función parcial en el dominio $Q \times \Sigma$, esto es, no es necesario que una transición esté definida para cada evento en Σ sobre cada estado de Q, por ejemplo $\delta(x, b)$ y $\delta(y, g)$ no están definidas.

Dos elementos son necesarios para definir completamente un autómata de estado finito: un estado inicial, denotado por q_0, y un subconjunto F de Q que representan los estados de Q que son marcados. El objetivo del conjunto F es darle un significado especial a un conjunto de estado del autómata. Los estados marcados a menudo son referenciados como estados de aceptación o estados finales. En la figura 7 el estado inicial se identifica porque existe una flecha que lo apunta y los estados marcados son identificados con doble círculos.

Ejemplo 2:
Se puede dibujar un AF como un control finito, el cual se encuentra en algún estado de Q, leyendo una secuencia de símbolos pertenecientes a Σ escritos

sobre una cinta como se observa en la figura 8. Para realizar un movimiento, el AF en el estado *q* lee el siguiente símbolo *a* que entra en el estado δ(q,*a*) y mueve hacia el siguiente símbolo a la derecha. Si δ(q,*a*) es un estado de aceptación, entonces el AF considera que la palabra escrita hasta ese momento ha sido aceptada.

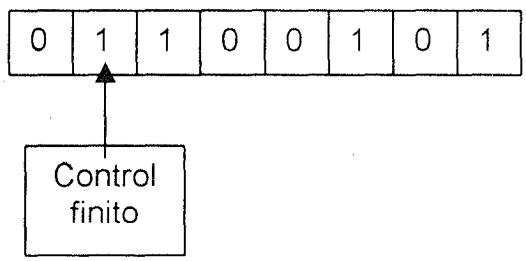

Figura 8. Autómata de estados finitos

Para describir formalmente el comportamiento de un AF sobre una cadena, se debe extender la función δ para ser aplicada a un estado y a una cadena en vez de una estado y a un símbolo. Se puede definir la función δ de $Q \times \Sigma^* \rightarrow Q$. de forma tal que $\hat{\delta}(q,w)$ es el estado del AF después de leer *w*, iniciándose en el estado *q*. En otras palabras, $\hat{\delta}(q,w)$ es el único estado p tal que es unido en el diagrama de transición por medio de un arco etiquetado con *w* que se inicia en *q*. Formalmente se define como:

1. $\hat{\delta}(q,\varepsilon) = q$
2. para todas las cadenas *w* y símbolos de entrada *a*

$$\hat{\delta}(q,wa) = \delta(\hat{\delta}(q,w),a)$$

Entonces, según (1), se establece que si no se lee ningún símbolo de entrada, el AF no cambia de estado, y según (2), se establece cómo encontrar el estado después de leer una cadena no vacía *wa*. Esto es, primero se encuentra el estado $p = \hat{\delta}(q,w)$, después de leer *w*, y luego se calcula el estado δ(q,*a*).

Una cadena *x* se dice que es aceptada por un autómata de estados finitos M = (Q,Σ,δ,q_0,F) si $\delta(q_0,x) = p$ para algún *p* en F. El lenguaje aceptado por M, designado por L(M), es el conjunto {*x* | $\delta(q_0,x)$ está en F}. Un lenguaje es un conjunto regular (o simplemente regular) si este es aceptado por un autómata de estados finitos.

3.3. Autómatas de estados finitos jerárquicos

En muchos problemas prácticos de control, un DEDS consiste en una gran cantidad de componentes que operan concurrentemente. Por lo tanto, la complejidad en la representación aumenta exponencialmente al aumentar el

número de componentes. Por otra parte, si se desea obtener una visión global de todo el conjunto es necesario generar el modelo global, una de las principales técnicas para construirlo es por medio de la composición de los autómatas de estado que modelan de los subsistemas, lo que implica que ya no es sólo la representación la que se torna compleja, sino también, la manipulación y análisis de la representación, ya que entre los algoritmos que usualmente son usados para realizar estas operaciones caen, en algunos casos, en la clasificación np-completo.

Para disminuir la complejidad del formalismo de los autómatas de estado finitos, se ha introducido el concepto de Autómatas Jerárquicos Finito (AJF) [46] como una simplificación de los diagramas de estado propuestos por Harel [9] que extiende los autómatas de estados finitos añadiéndole las características de Jerarquía y Ortogonalidad, específicamente:

1. Los estados son organizados en super-estados y sub-estados como un mecanismo para alcanzar la noción de profundidad.
2. Los estados están compuestos ortogonalmente como una manera para permitir la concurrencia.
3. Existen transiciones que son permitidas en todos los niveles de la estructura jerárquica.

Los AFJ han sido diseñados como una herramienta para la especificación y modelado de procesos complejos, como lo son los sistemas de manufactura, los sistemas de telecomunicaciones, los sistemas de distribución de recursos, los sistemas de control de tráfico aéreo, etc.. En Drusinski [8] se demostró que los diagramas de estados presentan una reducción significante en la descripción. De hecho, en su propuesta demostró que la complejidad se reduce exponencialmente con los Autómatas de Estados Finitos equivalentes. En los autómatas propuestos por Brave & Heymann[46], se logra un reducción sustancial al proponer los Autómatas Jerárquicos de Estados Finitos Asíncronos (AJFA), los cuales se caracterizan por un mínima interacción entre componentes paralelos de la jerarquía.

Esto significa que no existe un sincronismo entre los componentes ortogonales por medio de los eventos compartidos. Todas las interacciones se asumen que serán modeladas por medio de las transiciones de alto nivel o en las restricciones de control. Esto permite que análisis como el cálculo de alcanzabilidad pueda ser hecho dentro del contexto de los AJFA con una reducción exponencial de la complejidad, en comparación con el modelo del proceso realizado por medio de AF.

Brave y Haymann[46] lograron desarrollar un algoritmo eficiente para probar alcanzabilidad, que se basa fundamentalmente en el uso de la estructura jerárquica del proceso y logrando demostrar las ventajas de la representación por

medio de los AFAJ. El Algoritmo de prueba de alcanzabilidad es usado para resolver el problema "forbidden configuration" en el área de control.

3.4. Descripción formal de los Autómatas Jerárquicos Finitos (AJF)

Gráficamente, los estados son representados por los cuadrados . La jerarquía es representada por la inclusión de cuadrados dentro de otros cuadrados. En el AJF mostrado en la figura 9, los estados *a* y *e* son sub-estados inmediatos del estado *f*, y los estados *b* y *c* son sub-estados inmediatos de *a*. Los estados *b*, *c* y *d* son considerados también como sub-estados de *f*.

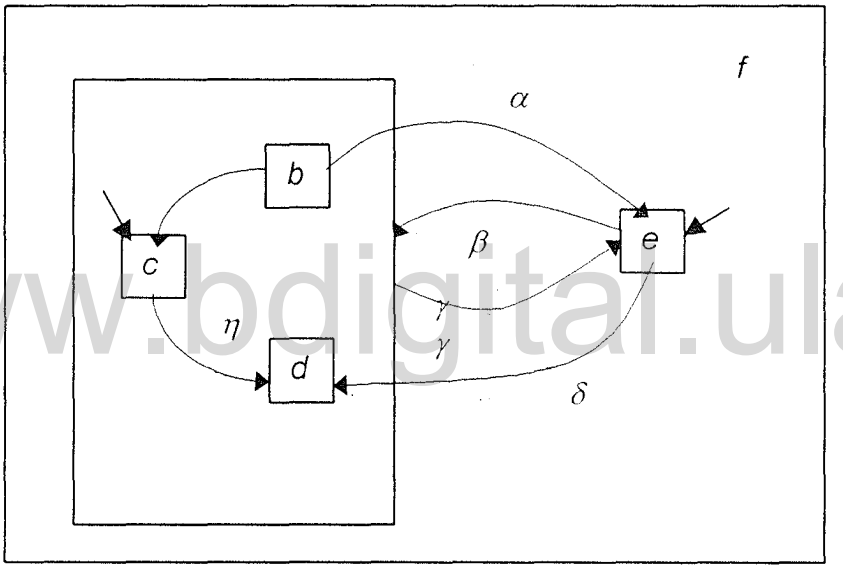

Figura 9. Ejemplo de un AJF

La figura 10 muestra el AF equivalente al AJF de la figura 9. Los símbolos desde α hasta η corresponden a los eventos asociados a las transiciones. El estado *a* es llamado un *estado OR* que puede estar en el estado *b*, *c*, o *d* (sólo uno a la vez), el evento γ permite la transición entre el estado *a* (*b*, *c* o *d*) hasta el estado *e*.

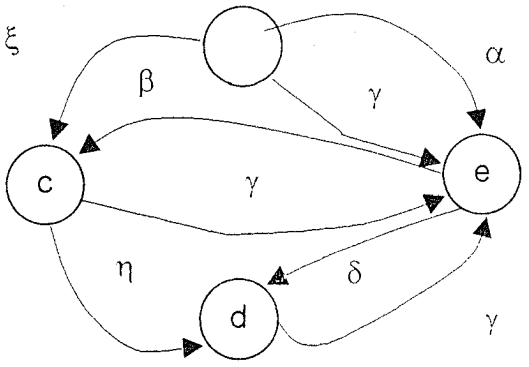

Figura 10. AF equivalente al AFJ de la figura 8

La discusión desarrollada en este capítulo conduce a concluir cómo las propuestas de Brave y Haymann permiten, teóricamente, lograr una reducción en la manipulación de la complejidad por medio de la jerarquización de los AF.

Capitulo 4: Conceptos para la implementación del Autómata Jerárquico Finito

4.1. Introducción

Varios autores han demostrado que el comportamiento de un sistema de producción continuo puede ser representado mediante Autómatas de Estados Finitos [11][16][19]. Utilizando el concepto de *sistemas híbridos*, se han construido diferentes aplicaciones académicas y algunas aplicaciones en el ambiente industrial. Por ejemplo, se pueden mencionar los trabajos de Lygeros en automatización de vehículos[21] y algunos otros en control del procesos realizados en Alemania por el grupo de Tittus [30]. Paquetes de simulación como Simulink ya incluyen elementos para el desarrollo de sistemas incluyendo las dinámicas continuas y discretas. Esta herramienta es conocida como STATE FLOW.

Aceptando que la evolución continua de un sistema físico puede abstraerse en un comportamiento descrito mediante un conjunto de Autómatas de Estados Finitos, se puede asumir que los principios de control supervisorio desarrollados por Wonham [35][37] y otros autores son aplicables para la supervisión de procesos, incluyendo los aspectos de control supervisorio jerárquico .

El objetivo de la discusión que sigue a continuación es el de definir una vía para la construcción de aplicaciones basadas en los de modelos de integración de plantas, que tomen en cuenta la jerarquía natural de los sistemas y que incluyan las diferentes vistas de un proceso de producción. La coordinación automática de un sistema de producción continua se verá simplificada si se toman en cuenta las diferentes vistas del proceso tales como producto, recursos y tecnología de producción.

4.2. Implementación de sistemas discretos jerárquicos.

Sea el siguiente un ejemplo sencillo [13], para describir el enfoque de solución, el cual consiste en la construcción de un sistema de supervisión para la distribución de agua a una población. El sistema está compuesto por un subsistema de captación de agua, una planta de tratamiento y un sistema de distribución en sí, tal como se muestra en la figura 11.

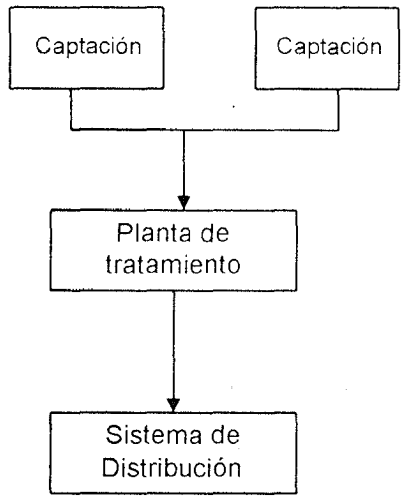

Figura 11. Sistema de supervisión para la distribución de agua a una población

Cada uno de los subsistemas posee su propia dinámica descrita aquí en términos de una máquina de estados finitos. El sistema de supervisión para este proceso complejo se muestra en la figura 12.

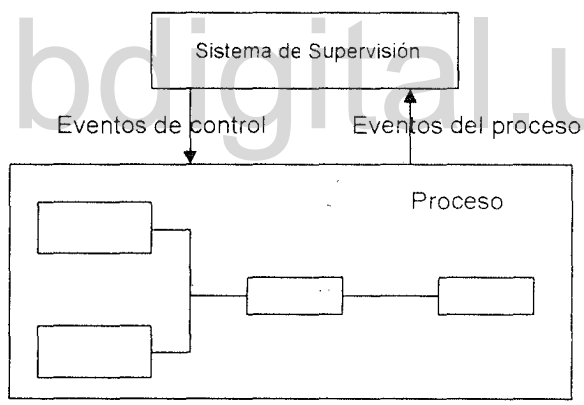

Figura 12 Sistema de supervisión

Cada subsistema es, a su vez, un sistema que está compuesto por elementos con una dinámica propia y el proceso en sí puede variar de acuerdo a un conjunto de condiciones, tales como la calidad del agua de entrada, las condiciones atmosféricas, el consumo poblacional, etc.. La coordinación global del sistema es realizada por el elemento que ve el conjunto de los subsistemas. Además, internamente cada subsistema es autónomo.

El subsistema captación se puede modelar a partir de los elementos componentes del mismo, como son la aducción, el filtrado y el sistema de bombeo, tal y como se muestra en la figura 13.

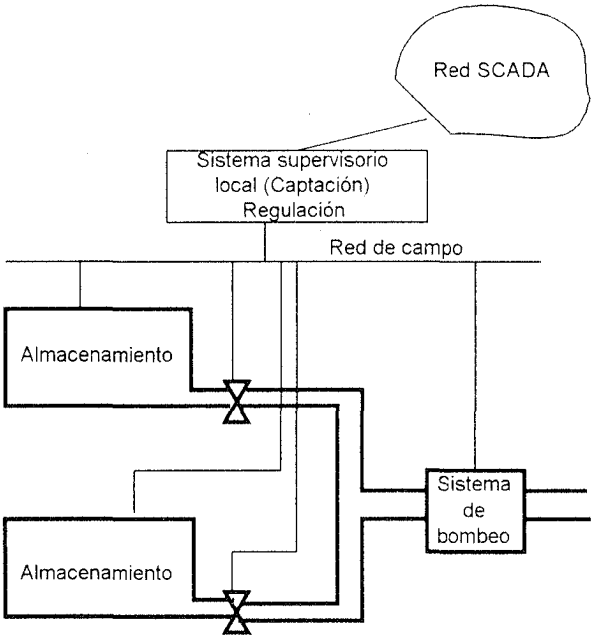

Figura.13 Subsistema de captación y bombeo de agua

Para este caso, se tiene un conjunto de elementos que conforman el subsistema de captación: Almacenamiento, Sistema de Bombeo, y el Sistema de Supervisión

Para el elemento de almacenamiento las variables que describen el estado son:

- Nivel del agua
- Calidad del agua
- Estado del tanque (mantenimiento)
- Operación del tanque (válvula abierta o cerrada)
- Sistema de bombeo
- Capacidad de bombeo
- Capacidad de bombeo actual
- Bombeo actual

Cada una de las variables tiene un modelo de comportamiento, y la composición de estos modelos especifica el comportamiento de cada componente del subsistema. De manera general, se puede decir que el comportamiento de este subsistema se puede modelar mediante la máquina de estados finitos descrita a en la figura 14 y que resulta de la composición de los comportamientos de los componentes del subsistema.

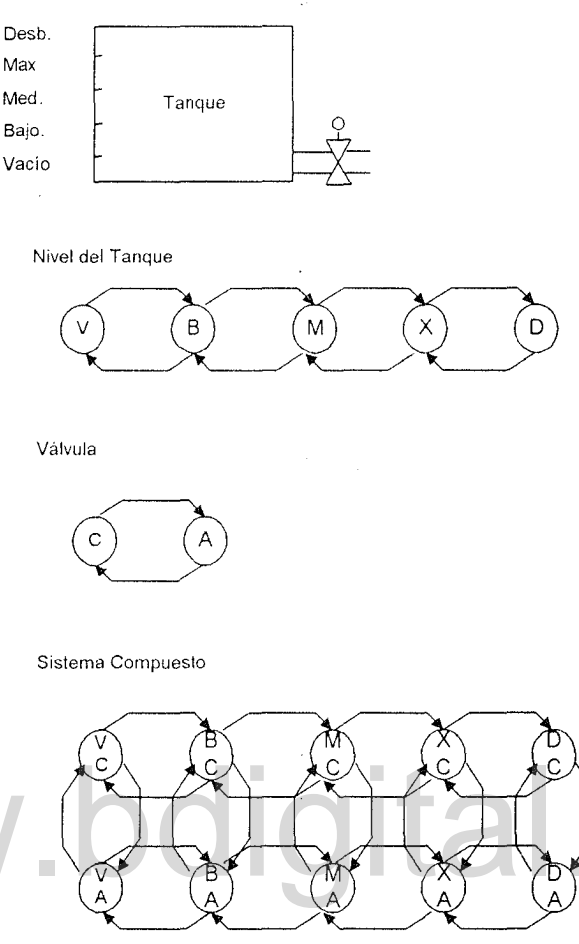

Figura 14. Comportamiento de un tanque

En la figura 14 se observa el Autómata de Estado Finito resultante al componer los AEF del Tanque y el AEF de la válvula, este AEF resultante posee 10 estados, los cuales pueden ser agrupados para reducir el numero de estados. La reducción de estados se logra al convertir el Autómata de Estados Finitos (AEF) en un Autómata Jerárquico de Estados Finitos (AJEF), que sea capaz de ofrecer diferentes vistas, donde cada una de las vistas representa un nivel diferente de abstracción.

La evolución de un AJEF es totalmente dependiente de los eventos capturados, y cada nivel de abstracción solo depende de algunos eventos del nivel de abstracción de la jerarquía inferior. En el nivel inferior de abstracción se tienen los mismos estados y transiciones presentes en la AEF original. Para crear el nivel inmediato superior, se procede a crear un número finito mayor de uno y menor que n de particiones conexas, donde n es el número de estados del nivel actual de abstracción. Cada nueva partición es un super-estado y las transiciones que llevan tanto hacia fuera como hacia dentro del super-estado son activadas por medio de eventos especiales. Estos eventos permiten definir el alfabeto de

eventos del nuevo nivel de abstracción y cada super-estado de la partición conformará cada uno de los estados. Se debe tener claro que varios eventos de un en un mismo nivel pueden corresponder a un sólo evento en el nivel superior. Por otro lado, aquel super-estado que contengan el estado inicial, se convertirá en el estado inicial del nuevo nivel de abstracción. Esto mismo aplica para aquel o aquellos super-estados que contengan uno o más estados finales, estos serán los nuevos estados finales en el nuevo nivel de abstracción.

La vista jerárquica del sistema de tanques para la supervisión del mismo se muestra en la figura 15, donde los datos que interesan a nivel de supervisión son:

- ON (Operación Normal: Agrupa a los estados con la válvula abierta y niveles Medio, Máximo y Desborde).
- OB: Válvula abierta y nivel bajo
- FO: Válvula cerrada y/o nivel del tanque en vacío.

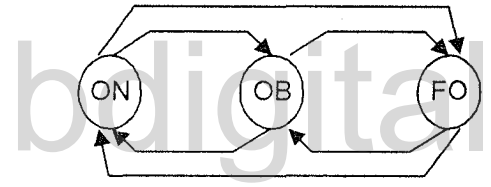

Figura. 15. Vista jerarquizada del autómata del tanque

En este capitulo se mostraron los conceptos necesarios para la abordar el diseño y construcción de los Autómatas Jerárquicos de Estados Finitos a partir de un Autómata de Estados Finitos. Es importante destacar que esta técnica permite manipular Autómatas de Estados Finitos que contienen un cierto número de estados por medio de un Autómata Jerárquico de Estados Finitos con dos o mas niveles de abstracción, donde cada nivel es a su vez modelado por un Autómata de Estados Finitos con un numero de estados menor o igual al original.

Capítulo 5:Diseño del Autómata Jerárquico de Estados Finitos

5.1. Introducción

La implementación de los de los AJEF es un pasó importante en la línea de investigación que se desarrolla en LASDAI, ya que nos permitirá tener una plataforma de trabajo básica para el comienzo de nuevas investigaciones en el área, eso nos habilitará para corregir y adaptar las herramientas usadas en los la construcción de sistemas basados en los AJEF. Las limitación impuestas implican el uso de software libre sobre plataforma UNIX o clones de esta. La figura 16 muestra el Diagrama de Flujo de Datos[45] del comportamiento del sistema propuesto, existe una fuente generadora de eventos. Estos son detectados por el AJEF, específicamente por el AEF de más bajo nivel de abstracción dentro de la jerarquía del AJEF, a este autómata se le conoce como máquina cero, este AEF consume el evento y dependiendo de su naturaleza lo propaga a los niveles superiores por medio de un mecanismo de comunicación de AEF que será expuesto más adelante.

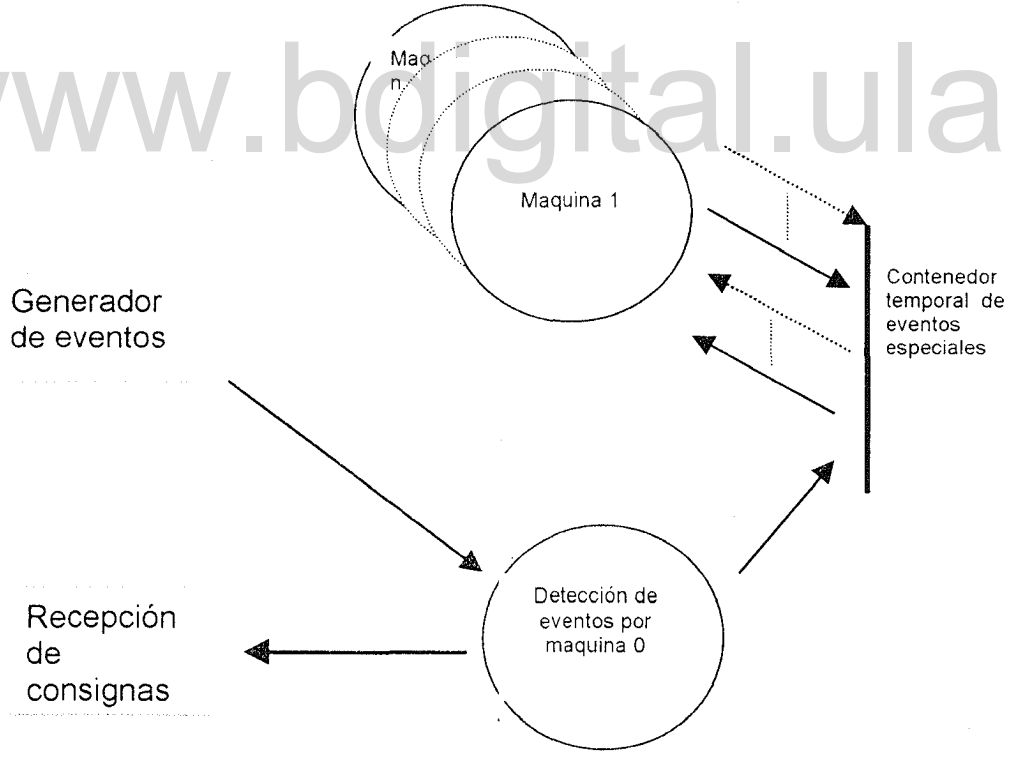

Figura 16, Diagrama de datos del comportamiento de un AJEF

5.2. Lista de requerimientos

Los requerimientos usados para la construcción de la aplicación se clasifica en:
-requerimientos de información:

- Se debe permitir conocer el estado o súper estado de cualquier nivel del AEJ.
- La configuración del AEJ puede ser suministrada mediante un archivo o por medio de un puerto lógico de comunicación del proceso.
- Se debe registrar en línea las transacciones que se están activando en cada nivel del AEJ.
- Se debe registrar los símbolos que producen cada uno de los AE que conforman el AEJ.

-de requerimientos de interfaz

- Cada autómata se modela como un proceso independiente sobre el mismo dominio de direcciones.
- Cada autómata acepta eventos del autómata de nivel inferior[4]
- Cada autómata produce eventos especiales a los AF de nivel o niveles superiores.

-de restricción

- El número de estados de cada AF debe ser > 0.
- El número de niveles de un AEJ debe ser > 0.
- Debe ser construido en ANSI C
- El mecanismos de comunicación entre procesos que conforma el sistemas puede ser IPC System V, o BSD Sockets.

5.3. Diseño del autómata jerárquico de estados finitos

Para la considerar el diseño de un prototipo de AJEF se pensó en la técnica de jerarquización en capas, como la usada por la ISO en el modelo ISO [18][3][48]. En esta técnica, las funciones de comunicación se distribuyen en un conjunto jerárquico de capas. Cada capa realiza un conjunto de funciones relacionadas entre sí, necesarias para comunicarse con otros sistemas. Cada capa se sustenta en una capa inmediatamente inferior, la cual realizará funciones mas primitivas, ocultado detalles a las capas superiores, en ningun momento se considera la interacción directa entra la capa N+1 y la Capa N-1 sin pasar por la capa N, ver figura 17. Una capa proporciona servicios a la capa inmediatamente superior. Idealmente, las capas deberían estar definidas para que los cambios en una capa no implicarán cambios en las otras capas.

[4] Este requisito se modifica en la siguiente versión al permitir la comunicación entre AF del mismo nivel.

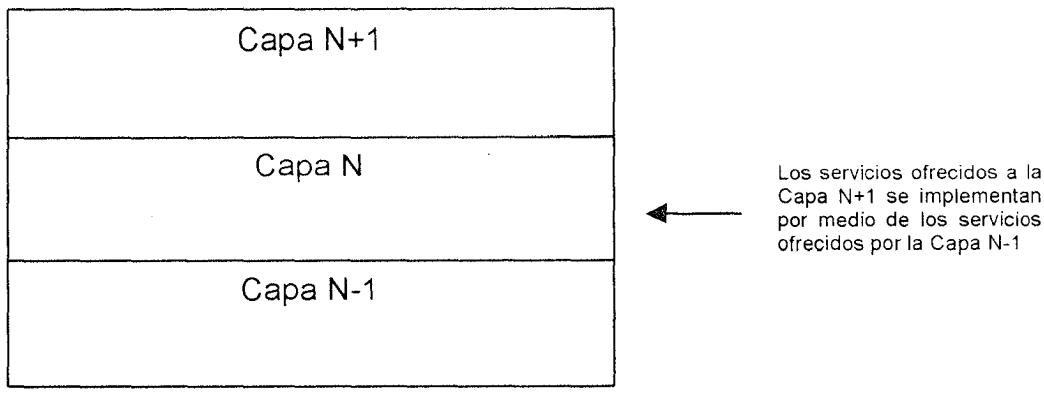

Figura 17 Características del Diseño por capas

Un sistema por capas se puede observar como un conjunto ordenado de mundos virtuales, cada uno construido en términos de otro que está por debajo y su vez proveyendo las bases para la implementación del que está por encima. Para el diseño del AJEF se proponen un sistema de tres capas como se muestra en la figura 17. La capa inferior se encarga de las primitiva para la comunicación entre los diferentes AEF con componen el AJEF. Esta capa sirve de midleware que permite la comunicación entre procesos. Esta integrada por las primitivas que permiten el uso del sistemas de colas implementado en el núcleo del sistema operativo. La siguiente se encarga de la adquisición de los mensajes recibidos en la capa inferior, y también se encarga del envío los eventos del la capa inferior. En la capa superior se encuentra el AEF de nivel *i*-esimo en si, éste debe detectar eventos en los mensajes recibidos por la capa inferior, disparar las transiciones pertinentes, y produciendo los símbolos pertinentes a las transiciones efectuadas. La figura 18 muestra el diseño por capas del AJEF.

La implementación de un AJEF se realiza a través del diseño orientado por objetos, ver figura 19, donde cada máquina de estados finitos se puede observar como una clase de objetos que contiene un conjunto finito de objetos parametrizados que representan los estados, y un conjunto de objetos parametrizados que representan los eventos. Estos eventos disparan transiciones (métodos) asociadas a cada uno de los diferentes estados, las cuales pueden generar instantáneamente eventos hacia otras AEF. El tener estados y eventos representados por medio de objetos parametrizados, trae como consecuencia la creación de autómatas parametrizados.

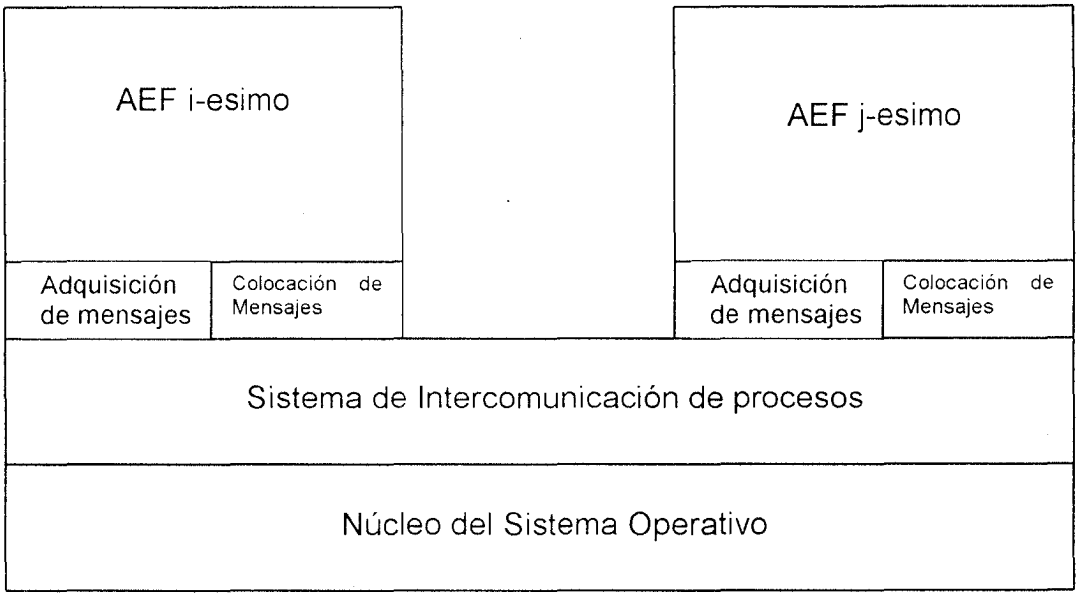

Figura 18, Diagrama de Diseño del AJEF

Nov. 1999 AF[TipoEstado: TS, TipoEvento: TE]		
1	Especificación Sintáctica: creaAF() → AF, destruyeAF() → get_actual() → TipoEstado, set_estadoinicial() → TipoEstado, activa_Evento(TipoEvento) → AF,	Especificación semántica: creaAF(): Crea un AF. destruyeAF(): elimina un AF. get_actual(): Observador: Obtiene el estado actual un AF. set_estadoinicial(): define el estado inicial del AF activa_Evento(e): envía el evento e un AF.
2	Declaraciones: ce: *lista[TipoEstado] ea: TipoEstado ei: TipoEstado ef: lista[TipoEstado]	ce: Vector de listas de estados que representa el conjunto de estados. ea: Estado actual. ei: Estado inicial. ef: Conjunto de estados finales.

Figura 19. Especificación en MEDEE de la clase Autómata Finito (AF)

Tomando lo anterior como base, se puede definir que la clase Autómata Jerárquico de Estados Finitos está compuesta por un número finito de objetos parametrizados de la clase Autómata Finito. En la figura 20 se observa el diagrama en UML [33] de la clase AJEF, donde explícitamente se definen las equivalencias entre los diferentes eventos que activan las transiciones en los niveles superiores de la jerarquía de abstracción, es decir, que al detectarse un evento en el nivel más bajo de la jerarquía, la transición que éste activa instantáneamente verifica las posibles proyecciones verticales del evento capturado. Esto hace que se actualicen inmediatamente todas las vistas o niveles en la AJEF.

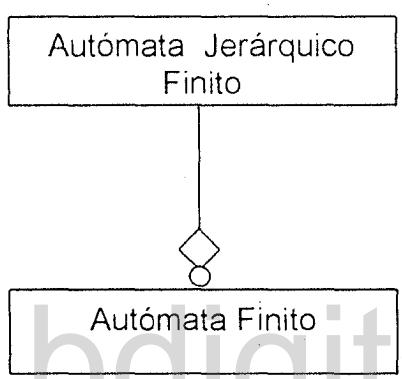

Figura 20. Diagrama en UML de la Clase Autómata Jerárquico

Un sistema modelado por medio de AJEF, (ver figura 21), puede obtener la descripción de cada nivel por medio de los mecanismos comúnmente usados, como lo son las tablas de transiciones o la especificación por medio de lenguajes formales. Estos últimos están en capacidad de explotar al máximo la característica paramétrica de la clase Autómata Finito Jerarquizado, ya que es posible definir estados representados con datos compuestos y con métodos asociados a cada estado.

Adicionalmente a la descripción de los diferentes estados, transiciones y eventos, es necesario suministrar la correspondencia entre los eventos de un nivel jerárquico y los eventos del nivel superior, lo cual se realiza por medio de la creación de alias entre eventos. Por ejemplo, se puede expresar que los eventos α y β del nivel de abstracción 2 son equivalentes al evento Σ del nivel 3 por medio de la siguiente sentencia:

Sigma.alias(Alpha || Beta);

En resumen, el tener la AJEF representada por medio de un objeto de la clase paramétrica, permite manipular de una manera sencilla autómatas de estado complejas por medio de lenguajes de programación de alto nivel como por ejemplo

JAVA o C++, permitiendo consultar, aceptar/enviar eventos desde/hacia cualquier dispositivo lógico conectado a una red de datos.

Nov. 1999 AJ[TipoEstado: TS, TipoEvento: TE]		
1	Especificación Sintáctica: creaAJ() → AJ, destruyeAJ() → get_actual() → TipoEstado set_nivel() → TipoEstado activa_Evento(TipoEvento) → Aj, creaAlias(TipoEvento, TipoEvento) →TipoEvento creaSuperEstado(lista[TipoEstado]) → TipoEstado	Especificación Semántica: CreaAJ(): Crea un AJF. destruyeAJ(): elimina un AJF. set_nivel(): Selecciona un nivel de jerarquía del AJF get_actual():observador: Obtiene el estado actual de un AJF en el nivel actual. activa_Evento(e): envía el evento e al AFJ. creaAlias(e1,e2): asocia el evento e2 al evento e1. creaSuperEstado(lista): crea un estado OR.
2	Declaraciones: caf: set[TipoEstado] nivel: AF	caf: conjunto de AF que componen el AFJ. nivel: AF que representa el nivel de abstracción actual.

Figura 21. Especificación en MEDEE de la clase Autómata Finito Jerarquizado

5.4. Funcionamiento del AJEF programado

Un Autómata Jerárquico de Estados Finitos se implementa por medio de varios procesos en ejecución. Estos procesos se clasifican en Máquina Cero y Maquina-*i*. Cada uno de los AEF que forma un AJEF es representado por un proceso del tipo Máquina *i*, por lo tanto habrán tantos procesos de este tipo como niveles tenga el AJEF. La comunicación entre los procesos se realiza por medio de una buzón de mensajes de IPC V[5].

Cada proceso Maquina-*i*, se inicia conociendo el nivel que este simula dentro del AJEF, toma la configuración de su AEF de un archivo de configuración, instancia un objeto de la clase AEF y asigna los valores del nivel correspondiente, a continuación pasa a estado bloqueado a la espera de un evento. Al recibir un evento direccionado a su nivel, ejercita al AEF bajo su responsabilidad, y si se dispara un evento especial, prepara el mensaje (ver figura 22) y lo envía hacia la cola de mensajes. Por último se bloquea a la espera de otro mensaje.

[5] IPC V es una herramienta de software que ofrece servicios de comunicación de procesos

Nivel Origen	Nivel Destino	Opciones	Datos

Figura 22. Formato del mensaje producido y consumidos internamente por el AJEF.

El proceso Máquina Cero es uno solo por cada AJEF, debe ser el último proceso en ejecutarse, este proceso está a la espera de cualquier evento dirigido hacia el AJEF, el debe verificar que cada evento pertenezca al alfabeto del AJEF y redirecionarlo a al proceso Máquina-i correcto, entre sus funciones está:

- Establecer la comunicación externa del AJEF.
- Registrar la evolución de todos los AEF dentro AJEF.
- Registrar todos los eventos entre los diferentes nivel del AJEF por medio de la supervisión de los mensajes enviados por cada proceso Máquina-i.

5.5. Desarrollos sobre la plataforma construida

Esta herramienta está operativa, sobre sistemas Linux 2.2, para su diseño se estudio el lenguaje de mejor conveniencia[17]. Por ser totalmente modular se le ha agregado una interfaz gráfica desarrollada en Java[32], también se ha desarrollado un modulo que permite la composición de autómatas[1] y se esta trabajando en el desarrollo de un protocolo para la comunicación de Autómatas Jerárquicos de Estados Finitos por medio de la extensión del proceso Máquina Cero.

CONCLUSIONES

En este trabajo se hizo un recorrido por los diferentes aspectos que están involucrados directamente con el proceso de automatización industrial. Esta investigación fue necesaria para poder entender la problemática global del objetivos de la investigación.

Se logró plasmar los diferentes enfoques propuestos para la clasificación de los procesos industriales, así como los marcos de referencia propuestos por la Organización Internacional de Estándares [18]. Gracias a estos modelos de referencia, es posible lograr una clasificación jerárquica de los procesos que intervienen en la mayoría de los complejos industriales del país. Al describir los diferentes componentes lógicos y físicos se observa que al subir el nivel de abstracción, se pueden obviar los detalles de implementación y todo lo concerniente a la toma de medidas, implementación de leyes de control y envío de puntos de operación. Todo se puede reducir a modelos que captan la esencia del proceso. Esta labor permite discriminar entre los diferentes niveles jerárquicos de operación, donde un sensor a nivel físico de una planta tiene su equivalente en niveles más altos, como puede ser un índice de precio a nivel de planificación. En el nivel físico la ley de control puede ordenar el cierre de una válvula y en el nivel de planificación puede significar el cierre de una fábrica completa. En otras palabras, en los diferentes niveles se pueden observar los componentes básicos como lo son los actuadores y sensores.

El Capítulo 2 permitió sirvio de introducción el área de los DEDS y se pudieron comparar sus formalismos para la creación de modelos con los disponibles en los sistemas dinámicos de variables continuas. Mientras en éstos últimos se tienen dos alternativas bien consolidadas, como lo son los sistemas de ecuaciones diferenciales y los sistemas de ecuaciones en diferencias, en el lado de los DEDS la historia apenas se está escribiendo. Se cuenta con una gran cantidad de formalismos basados en la teoría de trazas, que tratan de encontrar un espacio de aplicación bien definido. Se observa claramente que no existe un formalismo que satisfaga las aspiraciones de diseñadores y matemáticos a la vez.

La elección de un formalismo es definido por la naturaleza de la tarea a realizar, recursos disponibles, tiempo requerido, dominio del formalismo. Existen niveles de abstracción bien definidos que sirven de guía en el momento de diseñar y existe también una clasificación dentro de esos niveles para orientar la elección, pero aun así, una vez elegido el nivel de abstracción y el tipo de formalismo (implícito o explicito) es posible llegar uno o más formalismos. Por ejemplo, ¿Por qué se usó AEF en lugar de redes de Petri? La respuesta no está clara, simplemente se puede decir que existía una experiencia anterior con el formalismo basado en AEF, pero es igualmente interesante tratar de realizar la misma experiencia usando Redes de Petri.

Una vez elegido el formalismo de AEF, se pudieron seguir las ideas de reducción de complejidad por medio de la creación de super-estados o estados OR. Esta nueva manera de manipular los autómatas abre un nuevo camino en el estudio de los sistemas de estados finitos, logrando una reducción considerable del análisis comúnmente realizado sobre autómatas de estado, como lo es el estudio de alcanzabilidad, ya que fácilmente y trabajando con AJEF es posible describir las condiciones de suficiencia para el estudio de alcanzabilidad de estados. De una manera simple, si el estado de interés no se encuentra entre la trayectoria creada por una palabra formada por símbolos de los niveles altos de la jerarquía (eventos especiales) es posible descartar directamente el alcance del estado en estudio.

En el momento de estudiar la implementación de los AJEF, simplemente se pensó en la implantación en los lenguajes de programación comúnmente usados como C y JAVA. También se realizó un estudio de factibilidad en lenguajes basados en programación lógica como lo es ProLog [17]. La experiencia fue interesante ya que se estudió la posibilidad de que investigadores con poca experiencia en lenguajes de programación de alguna manera pudieran implementar modelos basados en AEF, como punto de partida para la construcción de AJEF.

Originalmente se pensó en la creación de un conjunto de herramientas para la construcción de AJEF, y bajo ese compromiso, se creó este trabajo. La herramienta OMT propuesta por Rumbaugh [44] permitió lograr la definición de modelos. La madurez que ha alcanzado el C++ [1] ha permitido una versión paramétrica que independiza totalmente la Clase Autómata Finito de la Clase Autómata Finito Jerarquizado. En muchas de las aplicaciones se han utilizados variables de tipo entero para representar los estados. Con esta versión propuesta basada en clases paramétricas, no se crea una clase específica, se crea una familia de clases o plantillas de clases. La clase en sí se crea al instanciar cada uno de los parámetros de clases. Es en ese momento cuando se crea una clase de Autómata de Estados Finitos específica y por ende una clase de Autómatas Jerarquico de Estados Finitos. En las pruebas realizadas se comprobó como la comunicación entre los AEF que conforman el AJEF sigue el comportamiento deseado, el número máximo de niveles probado alcanzo los diez niveles, aunque teóricamente cada AEF es representado por un proceso en ejecución, estos procesos en la mayoría de los casos permanecen bloquedados a la espera de un evento, por lo que el consumo del CPU es mínimo.

Aparentemente, el siguiente paso sería continuar enriqueciendo las clases paramétricas de Autómatas Finitos, pero esto no es conveniente, ya que actualmente LaSDAI está concentrando sus esfuerzos en el uso y estudio de objetos distribuidos como un medio para lograr la integración de sistemas. La línea de esfuerzo se debe concentrar en permitir que los objetos de la clase Autómatas Jerárquicos Finitos evolucionen a objetos distribuidos que permitan la interconexión de Autómatas Jerárquicos Finitos [39] . Estos objetos distribuidos bajo el nombre de Meta-Autómatas Jerarquicos han sido construidos usando el

modelo Corba (www.omg.org), específicamente la herramienta MICO [1]. De esta manera, los objetos de la clase podrán ser instanciados en cualquier punto de la red, y gracias a los agentes gestores de solicitudes de operaciones de objetos, ser invocados sin importar su localización en la red de datos, máquina, sistema operativo o lenguaje de programación. Otra de los trabajos desarrollados a partir de estas ideas es el desarrollo de un sistema de visualización de AJEF[32]

Se puede decir que la ejecución de este trabajo permitió desarrollar destrezas en el área de programación de sistemas que implementan DEDS por medio de procesos que son capaces de comunicarse y trabajar de manera conjunta, en paralelo se realizaron los siguientes Proyectos de Grados:

- Comparación cuantitativa en la implementación de modelos de autómatas utilizando programación lógica vs. programación tradicional [17].
- Herramienta de software para la Composición de Autómatas[1].
- Visualización Gráfica de Autómatas de Estado Jerárquico[32].
- Diseño de un protocolo para la Interconexión de Autómatas Jerárquicos Finitos[39].

Estos sistemas se han desarrollado para aplicaciones bajo la filosofía del modelo CIM. Es importante que por medio de las técnicas de extensión de Clases buscar la adaptación hacia Sistemas Holónicos[34][31] que logran un nuevo nivel en el proceso de Automatización Definitivamente, este es el siguiente paso a seguir, la experticia necesaria en el área de programación se puede decir que se domina Diseño de un protocolo para la Interconexión de Autómatas Jerárquicos Finitos.

Referencias

[1] A. Delphin, "Herramienta de software para la Composición de Autómatas", Proyecto de Grado, Escuela de Ingeniería de Sistemas, ULA, Febrero 2001.

[2] A. Puder, K. Römer, "MICO: An Open source Corba Implementation", Morgan Kaufmann Publishers, 3er edition, March 2000.

[3] A. Tanenbaum, "Redes de Computadoras" , Prentice Hall, 3ra ed., 1997

[4] Bjarne Stroustrup, "El Lenguaje de Programación C++", Segunda Edición, Addison Wesley, 1993.

[5] B. A. Branding, W. M. Wonham. "Supervisory control of timed discrete event system". Procceding 31st IEEE Conference on Desición and Control. Pp 3357 – 3362, Dec 1992.

[6] C. Cassandras, S. Lafortune, "Introduction to Discrete Event Systems", Kluwer Academic Publishers, 1999.

[7] CIMOSA, "Open System Architecture for CIM", ESPRIT Consortium AMICE, Springer, 1993.

[8] D. Drunsinske, "On syncronized statecharts", PhD Disertation, Weizmann Institute od Scinece, Rehoboth, Apr. 1988.

[9] D. Harel, "Statechart: a visual formalism for complex systems", Sci. Comput. Programing. Vol 8, 1987.

[10] D. Shobrys, D. C. White, "Planning, scheduling and control systems: why can they work together", Computers and Chemical Engineering. Vol 24, pp 163 – 173, 2000.

[11] E. Chacón, F. Szigeti O. Camacho. "Integral Automation of Industrial Complexes based on Hybrid Systems". *ISA Transactions* **(35)** 4, 1996,.

[12] E. Chacón, G. De Sarrazin, "Automatización Integral de Sistemas de Producción Continuos: Herramientas de Integración", Libro Texto Automatización Industrial, Universidad de Los Andes, Julio 1998.

[13] E. Chacón, J. Cardillo, R. Rivas, "Coordinación / Optimización de Complejos de Producción Continua mediante el uso de Sistemas Híbridos

Jerárquicos", Memorias del V Congreso Panamericado de Control y Automatización, , Mayo 2000.

[14] E. Chacón, R. Rivas, "Integration of Widely Distributed Production Control System by Using Virtual Devices", Procedings of de IFIP WG10.3 International Conference Applications in Parallel and Distrubuted System, 1994.

[15] E. Zamaï, A. Chaillet-Subias, M. Combacau, "An architecture for control and monitoring of discrete events systems", Computers in Industry. vol 36, pp 95 – 100, 1998.

[16] H. Garcia, A. Ray, "A Reconfigurable Hybrid System and Its Application to Power Plant Control", Transactions on Control Systems Technology. Vol 3, No 2, June 1995.

[17] H. Márquez, "Comparación cuantitativa en la implementación de modelos de autómatas utilizando programación vs. Programación tradicional", Tesis de Grado, Esc. De Ingeniería de Sistemas – ULA, 1998.

[18] J. Day, H. Zimmermann, "The OSI Reference Model", Proceedings of the IEEE, Vol 71, No 12, December 1993.

[19] J. G. Thistle, "On Control of System Modelled as Deterministic Rabin Automata", DEDS: Theory and Application. vol 5, pp 357 – 381, 1995.

[20] J. Hatvany, "Dreams Nightmares and Reality", Computers in Industry. vol 4, No 2, June 1983.

[21] J. Hopcroft, J. Ullman, "Introductión to Automata Theory, Languages and Computation", Addison Wesley, 1979.

[22] J. Lygeros and D. Godbole and S. Sastry, "A verified hybrid controller for automated vehicles", Special Issue on Hybrid Systems of the IEEE Transactions on Automatic Control, 1996.

[23] J. N. Tsitsiklis, "On the control discrete event dynamical systems", Mathematics Control Signals Systems., vol 2, pp 95 – 107, 1989.

[24] J. Ostrof, "A Logic for Real-Time Discrete Event Processes", IEEE Control Systems Magazine, June 1990.

[25] J. Pimentel, "Communicating Network for Manufacturing", Prentice Hall, Englewood Cliff, 1990.

[26] K. C. Wong, W. M. Wonham, " Hierarchical Control Discrete-Event System", Procceding of the 2nd European Control Conference '93, pp 509 – 512, June 1993.

[27] K. Ogata, "Ingeniería de Control Moderna", Capitulo 4, Modelo Matemáticos de los Sistemas Físicos, Prentice Hall, 1986.

[28] K. Passino, "Bridging the gap between Conventional and Intelligent Control", IEEE Control Systems, pp 12 – 18, June 1993.

[29] L. Gou, P. B. Luh, Y. Kyoya, "Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation", Computers in Industry. Vol 37, pp 213 – 231, 1998.

[30] Lennartson, M. Tittus, B. Egardt, S. Petterson, "Hybrid Systems in Process Control", IEEE Control Systems, Octubre 1996.

[31] L. Gou, P.B. Luh and Y.Kyoya, "Holonic Manufacturing: Architecture, Cooperation Mechanism, and Implementation", A Special Issue of Computers in Industry on Intelligent Manufacturing Systems, Vol. 37, pp. 213-231, 1998.

[32] M. Sánchez, "Visualización Gráfica de Autómatas de Estado Jerárquico", Proyecto de Grado (informes de Avance), Escuela de Ingeniería de Sistemas, ULA, Noviembre 2001.

[33] P. A. Muller, "Modelado de Objetos con UML", Ediciones Gestión 2000, 1997.

[34] P.B. Luh and L. Gou, "Holonic Manufacturing Systems", Proceedings of International Manufacturing Engineering Conferece (IMEC96), Storrs, USA, 1996

[35] P. J. Ramadge, W. M. Wonham. "Supervisory control of a class of discrete-event processes". *SIAM Journal of Control and Optimization*. vol 25. pp. 206 - 230. Jan 1987.

[36] P. J. Ramadge, W. M. Wonham. "Modular Feedback Logic for Discrete Event Systems". SIAM Journal od Control and Optimization. vol 25 1987. pp 1202 - 1218. Sept 1987.

[37] P. J. Ramadge, W. M. Wonham. "On the supremal controllable sublanguage of a given language". SIAM Journal od Control and Optimization. vol 25 1987. pp 327 – 659 . May 1987.

[38] P. J. Ramadge, "Some tractable supervisory control problems for discrete-event system modeled by Büchi automata"., IEEE Transactions Automatic Control. vol 34. pp 10 – 19, Jan 1989.

[39] R. Carrasquero, "Diseño de un protocolo para la Interconexión de Autómatas Jerárquicos Finitos", Proyecto de Grado (Informe de Avance), Escuela de Ingeniería de Sistemas, ULA, Noviembre 2001.

[40] R. Kumar, "Supervisory synthesis techniques for discrete event dinamical systems", PhD Thesis, The University of Texas, 1991.

[41] R. Pressman, "Ingeniería del Software, un Enfoque Práctico", Cuarta Edición, Mc Graw Hill, 1997.

[42] R. Sengupta, S. Lafortune, "An optimal control thoery for discrete event systems", SIAM Control Optim. Vol 36, n 2 pp 488 – 541, March 1998.

[43] R. Smedinga, "Using trace theory to model discrete events", Varaya and A. B. Kurzhanski, editors, Discrete Events Systems: Models and Applications, pp 81 – 99. Springer-Verlag, 1987.

[44] Rumbaugh et al., "Object-Oriented Modeling and Design", Prentice Hall, 1991.

[45] T. DeMarco, "Structured Analysys and System Specification", Yourdon Inc., 1979.

[46] Y. Brave, M. Heyman, "Control of Discrete Event Systems Modeled as Hierarchical State Machines", IEEE Transactions on Automatic Control, Vol 38, Nro. 12, Dic 1993.

[47] Y. Li , W. M. Wonham, "Concurrent Vector Discrete-Event Systems". *IEEE Transactions on AUTOMATIC CONTROL*. 40. , 1995

[48] W. Stallings, "Comunicaciones y Redes de Computadoras", Prentice may, 6ta Edición, 2000.