Entorno de modelado y simulación de sistemas de control en CIM

José L. Villarroel, José A. Fernández, Javier Sainz y Pedro R. Muro

En este trabajo se presenta una aplicación informática para el diseño del software de control jerárquico de sistemas flexibles de fabricación. Dicha aplicación aporta componentes que permiten el diseño de software de coordinación y de toma de decisiones, así como la simulación de los diseños realizados. La creación de un modelo es facilitada por interfases gráficas con el usuario. Se propone un esquema de representación de sistemas de manufactura como núcleo de la aplicación. Dicho esquema integra técnicas de representación estructurada del conocimiento basadas en «frames», redes de Petri de alto nivel para la descripción de la dinámica del sistema, así como el diseño orientado a objeto como base para la metodología de modelado. En este contexto de diseño, cada elemento de un sistema de fabricación es representado por un objeto cuya dinámica interna se especifica mediante una red de Petri de alto nivel.

Palabras clave: *Diseño de software de control, sistemas de manufactura, entorno de desarrollo, programación orientada a objeto, redes de Petri.*

1. INTRODUCCION

Un *Sistema Flexible de Fabricación (SFF)* consiste en un conjunto de máquinas y almacenes, un sistema automático de *transporte* y un sistema de *control* por computador, del que depende, en gran medida, la flexibilidad del sistema. El problema del control de SFF se suele abordar, debido a su complejidad, mediante una descomposición jerárquica en la que se pueden destacar los siguientes niveles:

— *Planificación de la producción*: A partir de las previsiones y pedidos de los clientes, de los objetivos de producción, del estado del almacén y de la capacidad de producción de la planta, se elabora un conjunto de *órdenes de producción* para cierto horizonte temporal. En general, cada orden lleva asociada una fecha de comienzo («earliest starting time») y una fecha debida («due date»). La ejecución de cada una de dichas órdenes implicará la realización de un conjunto de operaciones (manufactura, ensamblado, transporte, almacenado, etc.) por parte de los recursos productivos.

— *Planificación y secuenciamiento* («scheduling») de las operaciones: En este nivel, se realiza la asignación de fechas de ejecución y de recursos a cada operación individual, componiendo, de esta forma, el plan de operaciones. Dependiendo de la filosofía de producción adoptada, estas asignaciones pueden ser realizadas previamente a la producción, durante el tiempo de producción (toma de decisiones en tiempo real), o mediante una combinación de ambas aproximaciones.

— *Coordinación de los elementos de la planta* (máquinas, dispositivos de transporte, robots, almacenes, etc.) para que

J.L. Villarroel, J.A. Fernández, J. Sainz, P.R. Muro, Grupo de Ingeniería de Sistemas e Informática, Centro Politécnico Superior de Ingenieros, Universidad de Zaragoza, C/ María de Luna, 3, 50015 Zaragoza.

cooperen en la ejecución de las operaciones. La función de coordinación supone: ordenar la ejecución de las actividades locales requeridas para ejecutar cada operación, actualizar la representación del estado del sistema, así como detectar situaciones excepcionales y generar las acciones oportunas que permitan regresar al funcionamiento normal (detección, diagnosis y recuperación). La decisión de qué operaciones han de realizarse en cada instante debe realizarse en función del estado del sistema y del plan de operaciones construido en el nivel superior de control.

— *Control local* de los elementos de la planta para la ejecución de tareas asociadas a las actividades locales.

Para realizar su función, un sistema de control debe poseer un modelo del SFF donde quede reflejado conocimiento detallado sobre operaciones, rutas de proceso, máquinas, áreas de trabajo, herramientas, materiales, órdenes, etc.

El diseño de las diferentes funciones involucradas en el control de SFF ha sido abordado desde múltiples aproximaciones, entre las que cabe mencionar las basadas en técnicas de inteligencia artificial y las basadas en modelos formales como, por ejemplo, redes de Petri.

En el campo de la inteligencia artificial se han desarrollado diversas aproximaciones para el modelado de sistemas de fabricación [17]. Inicialmente las aproximaciones basadas en reglas [1,8] fueron las más utilizadas, pero éstas están siendo desplazadas por aproximaciones basadas en el modelo [4,16], donde se emplean técnicas de representación estructurada del conocimiento, generalmente basadas en el concepto de «frame».

Por otra parte, el modelado y análisis de sistemas de fabricación han constituido una aplicación de especial interés en el campo de las redes de Petri [2,15]. El objetivo es la construcción de modelos de sistemas de fabricación que verifiquen ciertas propiedades de tipo cualitativo, como ausencia de bloqueos, limitación de capacidades, exclusión mutua en el uso de máquinas, etc. Para tratar modelos complejos, como el de un SFF, se han empleado formalismos basados en redes de alto nivel, donde la composición modular es la metodología predominante [5,9]. Esta aproximación es la adoptada en este trabajo para el diseño del modelo de coordinación de un SFF.

En este trabajo se presenta una aplicación informática para el diseño de software de control jerárquico de SFF. Dicho control se supone basado en la arquitectura jerárquica [10], que se encuentra esquematizada en la figura 1. En el nivel más bajo de la arquitectura se sitúan los *controladores locales* de las máquinas, robots, almacenes, transportes, etc. El *coordinador* tiene como función gestionar y supervisar la ejecución de tareas por los controladores locales. Para ello, evalúa qué operaciones pueden comenzar en un cierto instante y cuáles de ellas pueden ejecutarse simultáneamente. De entre todas estas operaciones, es preciso decidir cuáles comenzarán efectivamente. Estas decisiones son tomadas por el *distribuidor de operaciones* que puede basarse en un plan de operaciones, previamente construido, donde se reflejan asignaciones de fechas y recursos. El *planificador de operaciones* genera dicho plan en función de los objetivos de producción marcados por la *gestión empresarial*. Los diferentes niveles de la arquitectura comparten un mismo modelo de SFF residente en la *base de conocimiento*. Sin embargo, cada subsistema tendrá una visión particular de ese modelo.

Además de la utilización conjunta de técnicas de representación estructurada del conocimiento y de redes de Petri, se

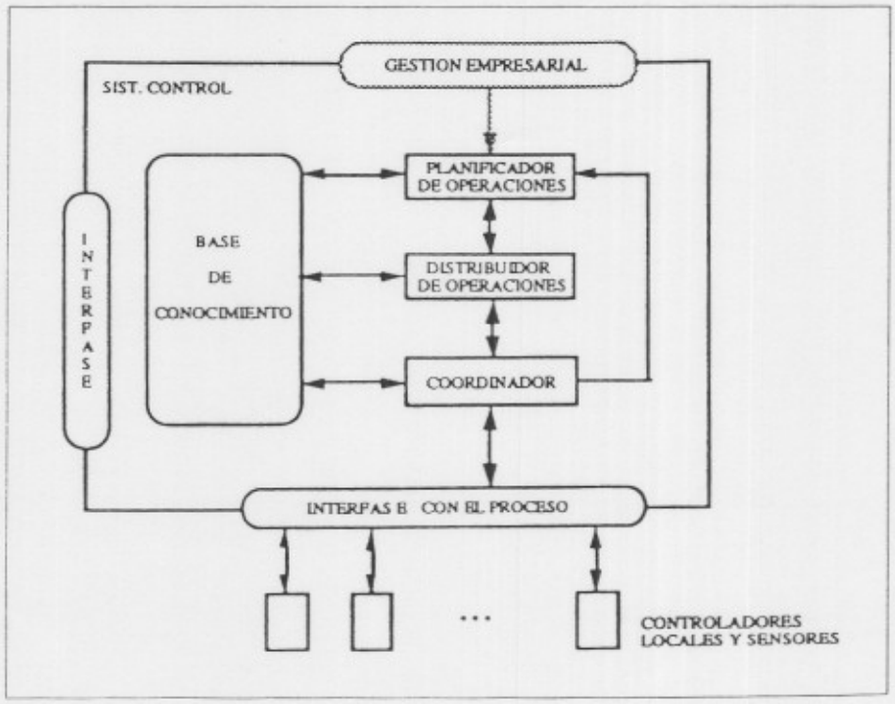


Figura 1. Arquitectura de control.

propone el diseño orientado a objeto como base de la metodología de modelado para el entorno de diseño que se presenta. Cada elemento de un sistema de fabricación es representado por un objeto cuya dinámica se describe mediante una red de alto nivel. De esta forma se combina la metodología y posibilidades de representación y manipulación del diseño orientado a objeto con el formalismo de las redes de alto nivel.

El trabajo que se presenta está estructurado de la siguiente forma. En la sección 2 se presenta la funcionalidad de los diferentes módulos que componen la herramienta de desarrollo de software de control de SFF, objeto del presente trabajo. La herramienta que se presenta está constituida por dos capas. La primera, presentada en la sección 3, está constituida por una herramienta de análisis y diseño de sistemas de eventos discretos de propósito general. La segunda capa, presentada en la sección 4, es una especialización de la anterior para el dominio de aplicación de los SFF. Por último, en la sección 5 se establecen las principales conclusiones de este trabajo.

2. PRESENTACION DEL ENTORNO DE DESARROLLO

El entorno de desarrollo, objeto de este trabajo, está estructurado en dos capas (la figura 2 muestra dicha estructura), cada una con su respectiva interfase gráfica:

1. La primera capa es una herramienta de análisis y diseño de sistemas de eventos discretos de propósito general, que está compuesta a su vez por cuatro componentes:

KRON: Constituye el núcleo fundamental del entorno. KRON (*Knowledge Representation Oriented Nets*) es un lenguaje orientado a objeto para la representación del conocimiento, especialmente orientado al modelado y manipulación de sistemas de eventos discretos. Permite la creación de modelos ejecutables donde la dinámica se representa utilizando una herramienta formal.

CONTROL: Este componente constituye el mecanismo de inferencia específico y aporta las primitivas necesarias para la ejecución de los modelos construidos con KRON.

SOLUCIONADOR DE CONFLICTOS: Los modelos construidos pueden ser no totalmente deterministas, es decir, necesitar de decisiones exteriores que determinen su evolución. Este componente permite el diseño de estrategias de decisión y asiste al CONTROL en la ejecución de modelos no deterministas.

INTERFASE GRAFICA: Permite la visualización e introducción de forma gráfico-textual de los elementos que componen un modelo y de sus interrelaciones. Asimismo, facilita la visualización del estado del sistema y de su evolución.

2. La segunda capa, construida utilizando los objetos y primitivas definidas en la capa inferior, constituye una especialización para el dominio de aplicación de los SFF. Dispone también de cuatro componentes:

MIKRON: Es un lenguaje de representación que especializa KRON para SFF. MIKRON (*Manufacturing Intended KRON*) aporta los objetos y primitivas específicas para la construcción del modelo de coordinación de los elementos de un SFF y sus relaciones, en términos propios del dominio de aplicación.

SIMULADOR: Es un simulador de eventos discretos específico para SFF. Utiliza las primitivas aportadas por Control para hacer evolucionar un modelo de coordinación simulando su interrelación con la planta de producción. Dispone

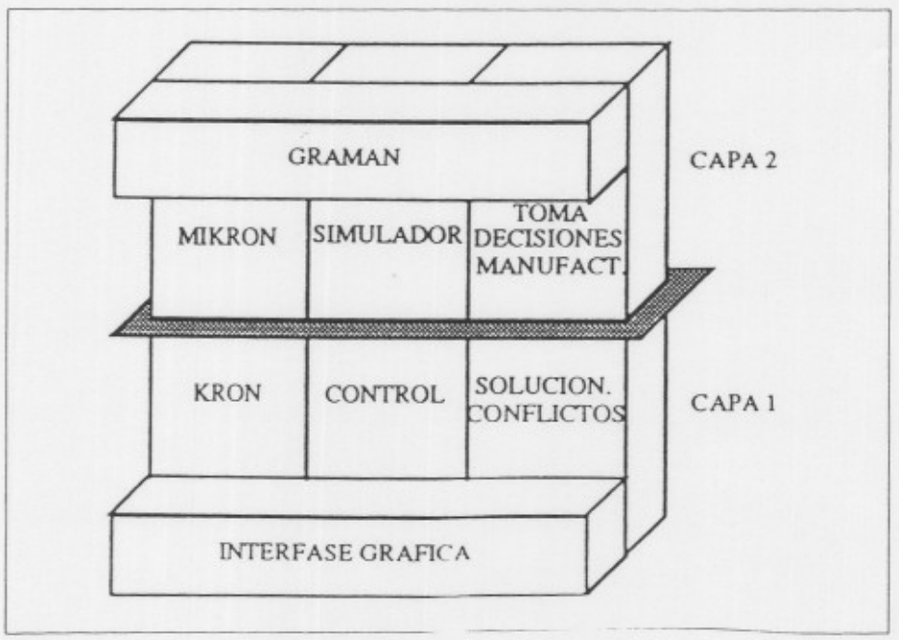


Figura 2. Arquitectura del entorno del sistema de desarrollo.

de utilidades para la especificación de condiciones de simulación, parámetros probabilísticos, recolección de datos, generación de informes, etc.

TOMA DE DECISIONES DE MANUFACTURA: Este componente, construido sobre el SOLUCIONADOR DE CONFLICTOS de la primera capa, aporta una biblioteca de políticas de decisión que facilita el diseño de estrategias de decisión en términos propios de SFF.

GRAMAN (*GRAphic MANufacturing*): Posibilita la utilización de las primitivas de diseño definidas en MIKRON de forma gráfica y la animación de las simulaciones de los modelos construidos.

Para facilitar la etapa de construcción del entorno, se ha optado por una herramienta potente de desarrollo, con buenas interfases de usuario y facilidades de manejo. Una primera versión prototipo se ha implementado en una estación de trabajo Xerox sobre el entorno LOOPS. Actualmente el sistema definitivo de desarrollo se está implementando en una estación Sun-SPARC sobre entorno KEE.

3. MODELADO DE SISTEMAS DE EVENTOS DISCRETOS

3.1 KRON

KRON (*Knowledge Representation Oriented Net*) [12,13,20], es un lenguaje de representación del conocimiento de sistemas de eventos discretos concurrentes. KRON está basado en la integración de técnicas de representación basadas en «frames», programación orientada a objeto y redes de Petri de alto nivel. La programación orientada a objeto proporciona la base de la metodología de modelado, los «frames» aportan el esquema genérico de representación del conocimiento y las Redes de Petri de Alto Nivel (RAN) constituyen el formalismo adoptado para representar el comportamiento dinámico.

Para el desarrollo de este lenguaje ha sido de particular interés la metodología de representación introducida en [4] y [14]. En este contexto, se persigue la representación modular mediante la definición de jerarquías de especialización, y se facilita la reusabilidad de los modelos mediante la integración del conocimiento en objetos.

Por su parte, las ventajas de la utilización del formalismo de las RAN [7] son conocidas (véase por ejemplo [6] en el dominio de los SFF) y se pueden resumir en las siguientes:

- Las RAN suministran un medio para describir, analizar y validar la consistencia de la dinámica del modelo, y a la vez permiten abordar la problemática de la sincronización de actividades entre entidades y especificar el flujo de información entre éstas.
- Son una herramienta formal para la cual se pueden adoptar metodologías de construcción de modelos.
- Es posible obtener automáticamente una realización software del modelo representado por la red [3,20].

Analizando las ventajas e inconvenientes de las RAN y de las herramientas provenientes de la inteligencia artificial en el contexto de la representación del conocimiento de sistemas dinámicos discretos, se pueden establecer las siguientes conclusiones [13,20]: por una parte, las herramientas de tipo «frame» carecen del formalismo necesario para describir y analizar comportamientos dinámicos, y por otra, las RAN soslayan por completo el conocimiento no relacionado con dicho comportamiento. Además, no existe una relación inmediata entre los elementos que componen una RAN y los objetos físicos o conceptuales que modela. Se trata, pues, de dos aproximaciones que pueden considerarse complementarias y cuya integración está favorecida por la existencia de múltiples analogías entre conceptos de RAN y de inteligencia artificial.

3.1.1. *Objetos básicos*

De acuerdo con la metodología de la programación orientada a objeto, en un modelo KRON, cada entidad está representada mediante un objeto que recoge toda la información concerniente a la entidad que modela y a sus relaciones con otros objetos, estructurada en atributos(1). Si la entidad posee características dinámicas, su comportamiento se representa mediante una RAN integrada con el resto de conocimiento de la entidad. Desde el punto de vista de la dinámica, se pueden distinguir las siguientes clases de objetos:

- *Objetos de Marcado*. Representan entidades sin comportamiento dinámico, al nivel de abstracción considerado. Tienen la misma semántica que las marcas de las RAN.
- *Objetos de Estado*. Son los objetos que representan entidades dinámicas. Para la representación del estado, dispone de los llamados *atributos de estado* (misma semántica que los lugares en una RAN). El estado de una entidad dinámica está definido por la distribución de objetos de marcado en los atributos de estado del objeto de estado que representa a la entidad.
- *Objetos de Acción*. Las acciones o actividades que modifican el estado de una entidad dinámica (transiciones en una RAN), son representadas en KRON mediante objetos de acción. Estos incluyen las *relaciones de red* (arcos de una RAN) que especifican las precondiciones y poscondiciones de la actividad modelada, y su descripción.

3.1.2. *Metodología de modelado*

Los objetos están organizados en jerarquías de especialización, en las que cada objeto hereda todos los atributos de sus objetos padre, incluida la dinámica, es decir, la RAN subyacente. La metodología de modelado subyacente en KRON consiste básicamente en los siguientes pasos:

1. Identificación de las entidades, dinámicas y no dinámicas, que componen el sistema a modelar.

(1) En literatura inglesa: *slots*.

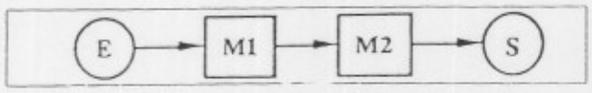


Figura 3. Esquema de fabricación en línea («flowshop») del ejemplo.

2. Agrupación de las entidades en clases.
3. Diseño de un modelo o prototipo para cada clase a partir de la especialización de prototipos ya existentes. Si se trata de una entidad no dinámica el prototipo será un objeto de marcado y si se trata de una entidad dinámica el prototipo estará compuesto por un objeto de estado y un conjunto de objetos de acción.
4. Creación de un ejemplar («instance») por cada entidad identificada en el sistema a partir del prototipo correspondiente.
5. Representación de las interacciones entre las diferentes entidades del sistema mediante el establecimiento de relaciones de *sincronización* entre objetos de acción.
6. Establecimiento del estado del sistema mediante la asignación de objetos de marcado a los atributos de estado de los objetos de estado correspondientes.

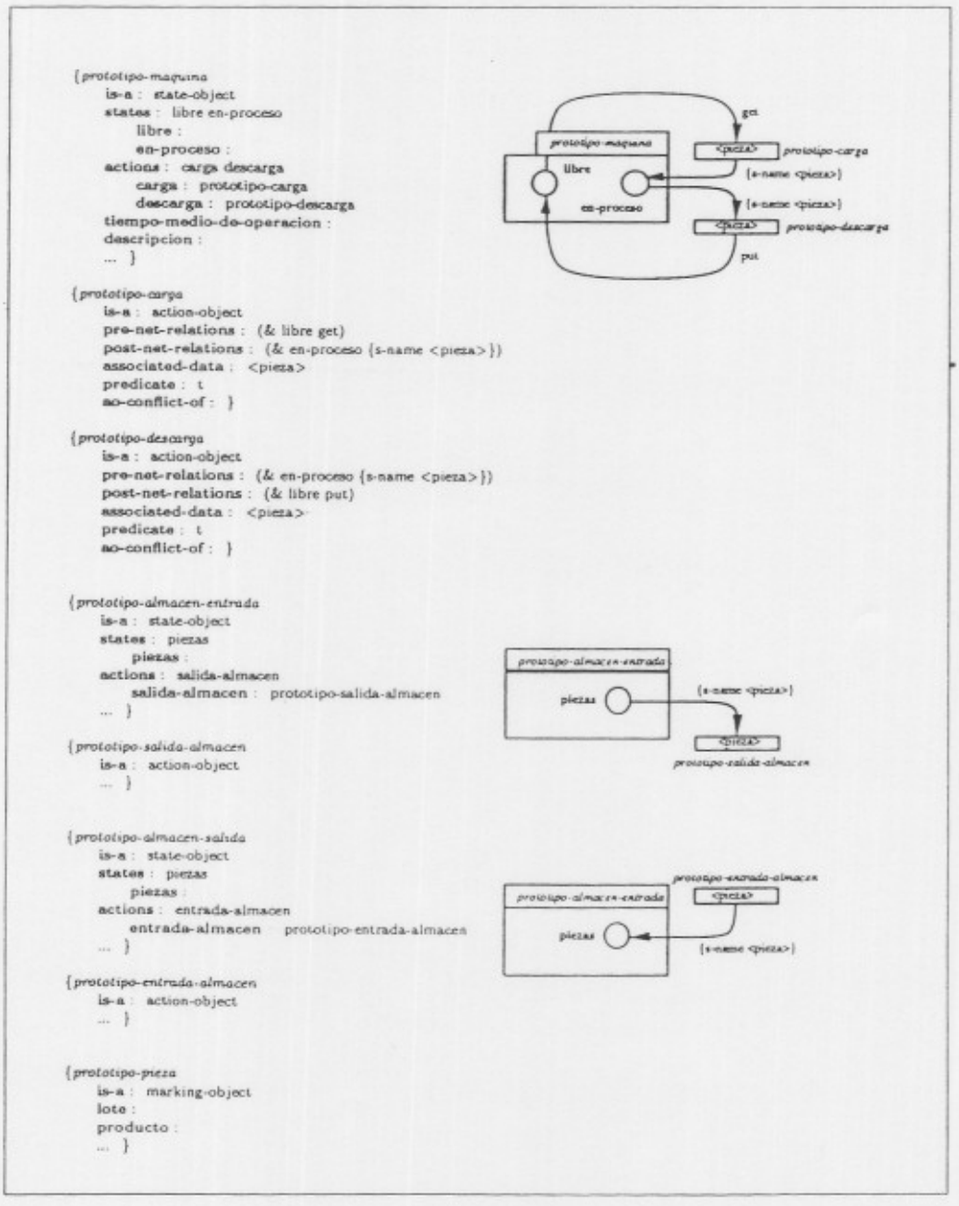


Figura 4. Prototipos que modelan las entidades del ejemplo de fabricación y RAN subyacentes a las entidades dinámicas.

3.1.3. Ejemplo de modelado en KRON

Para ilustrar estas ideas se presenta a continuación un sencillo ejemplo. Se trata de una línea de fabricación (figura 3) compuesta por un almacén de entrada, uno de salida y dos máquinas en serie («flowshop»). Mediante un análisis se identifican las siguientes entidades: el almacén de entrada, el de salida, las máquinas y las piezas que deben ser procesadas. Las máquinas y los almacenes son entidades con dinámica, no así las piezas. En la figura 4 pueden observarse parcialmente los prototipos de las entidades identificadas, así como las RAN subyacentes de las entidades dinámicas.

En una máquina se consideran dos actividades fundamentales: la carga y la descarga de piezas. Estas actividades están modeladas respectivamente por los objetos de acción: **prototipo-carga** y **prototipo-descarga**. Cada una de estas actividades cambia el estado de la máquina, representado por los atributos de estado **libre** y **en-proceso**. De forma similar puede razonarse con los prototipos de los almacenes. La consecución del modelo final prosigue con la creación de un ejemplar por cada pieza, donde se recogerá el conocimiento particular de cada una, de dos ejemplares del prototipo de máquina (**M1** y **M2**), y de un ejemplar de cada tipo de almacén.

Máquinas y almacenes interaccionan entre sí. Por ejemplo, la descarga de la máquina **M1** supone la carga de la máquina **M2**. Dicha interacción se puede reflejar en el modelo estableciendo una sicronización entre los objetos de acción **descarga-M1** y **carga-M2**, lo cual significa que las actividades modeladas son, en realidad, visiones diferentes de la misma acción (condicionada ahora por la unión de las precondiciones y poscondiciones de ambas). Finalmente, se establece el estado del sistema (marcado de la RAN subyacente) disponiendo ejemplares de objetos de marcado en atributos de estado. En la figura 5 puede observarse esquemáticamente el modelo final sincronizado correspondiente a un estado determinado.

Como conclusión, con KRON se consigue dotar a cada objeto de una potencia de representación más amplia que la estrictamente dinámica. Los sistemas se construyen mediante la interconexión de objetos previamente definidos y organizados en jerarquías, con lo que se favorece la reusabilidad y el prototipado rápido.

3.2. Control

La dinámica de un sistema modelado con KRON está representada por una RAN subyacente. Sus arcos están etiquetados con expresiones (condiciones de arco) que especifican cuándo y cómo un objeto de acción puede disparar (ejecución de la actividad modelada) en función del marcado y la actualización de éste, resultado del disparo. En el ejemplo de la figura 5, el objeto de acción entrada puede disparar con respecto a **{<pieza> = P122}** o bien con respecto a **{<pieza> = P342}**. El disparo del objeto de acción entrada con respecto a **{<pieza> = P122}** está representando la salida de la pieza **P122** del almacén de entrada para ser cargada en la primera máquina de la línea. Dicho disparo supone:

1. La retirada del atributo de estado **piezas** de **E** del objeto de marcado **P122** y del atributo **libre** de **M1** del objeto de marcado neutro **{•}**; y
2. el depósito del objeto de marcado **P122** en el atributo de estado **en-proceso** de **M1**.

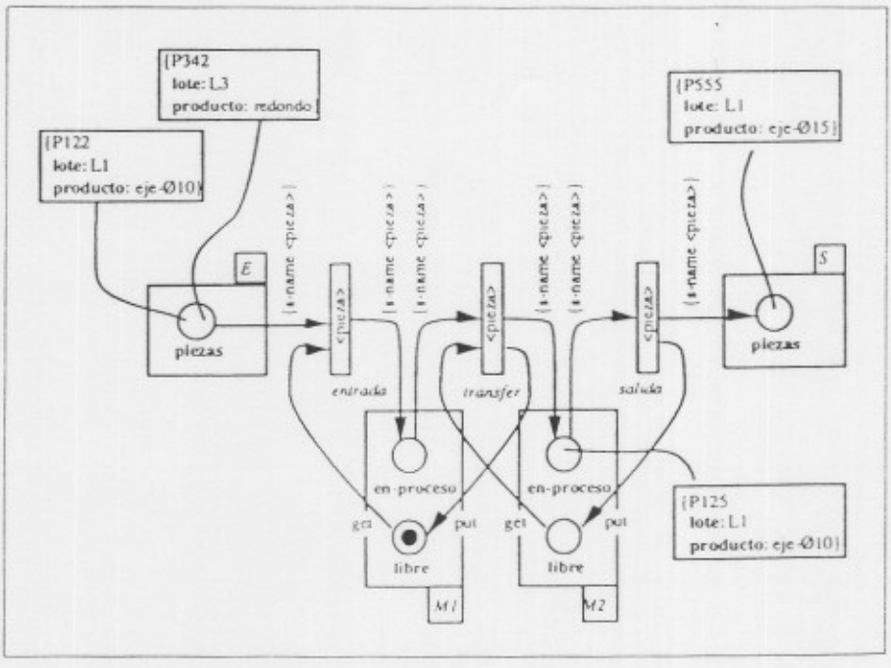


Figura 5. RAN subyacente en el modelo final de la línea de fabricación.

Esta actualización del marcado representa que la máquina **M1** deja de estar libre y pasa a estar procesando la pieza **P122**.

Para permitir la ejecución de los modelos construidos con KRON, el componente CONTROL aporta un conjunto de primitivas de manipulación que implementan las reglas de evolución de una RAN (disparo de objetos de acción, actualización del marcado, etc.). Sobre estas primitivas se puede utilizar cualquier algoritmo de control, sin embargo, este módulo aporta también un algoritmo de control por defecto de altas prestaciones. Las primitivas de manipulación de la dinámica y el algoritmo de control citado se encuentran desarrolladas en detalle en [20].

La eficiencia de las primitivas de manejo de la dinámica de CONTROL está basada en la eliminación de recálculos utilizando ideas del algoritmo RETE de análisis de sensibilización de reglas (desarrollado para el lenguaje de reglas OPS5), en la línea de [18]. La idea es almacenar los resultados intermedios de los cálculos que se realizan para establecer la disparabilidad de un objeto de acción en una estructura de datos de tipo árbol dentro del propio objeto. Dicha la estructura de datos permanecerá inalterada hasta que ocurra un cambio de marcado en los atributos de estado precondición de dicho objeto de acción. De esta forma es posible evitar recálculos innecesarios mientras no haya cambios en el estado del modelo.

3.3. Solucionador de conflictos

Un modelo KRON puede especificar indeterminismo en la dinámica. Así, pueden existir situaciones en las cuales sea necesario tomar una decisión que determine la evolución del modelo. Retomando el ejemplo de fabricación en línea a partir del estado mostrado en la figura 5, el objeto de acción entrada puede ser disparado con respecto a **{<pieza> = P122}** (entrada en la línea de **P122**) o bien con respecto a **{<pieza> = P342}** (entrada en la línea de **P342**). Será preciso decidir cuál será el disparo efectivo de dicho objeto de acción, esto es, qué pieza entrará en la línea.

Este tipo de situaciones, en las cuales es preciso tomar decisiones, están reflejadas en la RAN subyacente de un modelo mediante estructuras denominadas *conflictos*. Un conflicto es un objeto de acción o agrupación de ellos que constituye un punto potencial de decisión. Cada conflicto de un modelo KRON está representado por un objeto denominado *objeto conflicto*. Cuando en la evolución de un modelo surge un problema de decisión, éste es resuelto por el objeto conflicto que lo representa mediante la aplicación de una política de control que retorna una solución.

Mediante el módulo SOLUCIONADOR DE CONFLICTOS, los conflictos de una RAN subyacente pueden ser puestos de manifiesto de forma automática. Una vez detectado un conflicto, es preciso caracterizarlo para, posteriormente, diseñar la política de control que resolverá el problema de decisión asociado. Este módulo dispone de una biblioteca de políticas de control, genéricas para sistemas de eventos discretos, de la que el usuario puede disponer para diseñar sus propias estrategias de decisión. Cada conflicto dispone asimismo de una política de control por defecto.

3.4. Interfase gráfica

La primera capa del entorno presenta al usuario una interfase gráfico-textual que facilita el manejo de las primitivas de los tres componentes presentados. Se trata de una interfase multiventana, dirigida por menús, que posee la funcionalidad típica de un entorno orientado a objeto: visualización de jerarquías, visualización y edición especializada de objetos y métodos, creación de ejemplares y clases, etc.

De entre los aspectos específicos cabe resaltar las ventanas especializadas en RAN que permiten visualizar las redes subyacentes en un modelo KRON, en un formato similar al de las figuras 4 y 5. Dichas ventanas permiten también, de forma gráfica, la creación y modificación de las redes, el establecimiento de relaciones de sincronización entre objetos de acción, la animación de la evolución de un modelo, el control de dicha evolución por el usuario, etc.

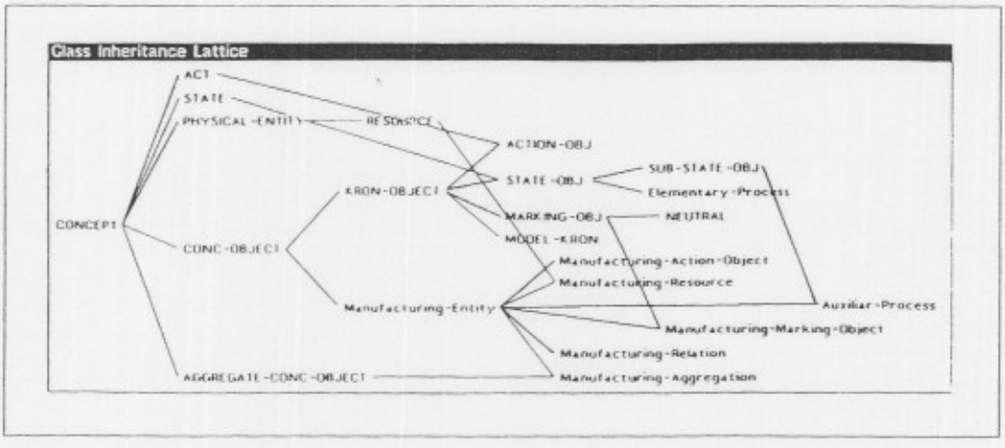


Figura 6. Raíces de las jerarquías de MIKRON.

4. MODELADO DE SISTEMAS DE MANUFACTURA

Un Sistema Flexible de Fabricación es un sistema de eventos discretos y, por lo tanto, se puede abordar el diseño de su modelo de coordinación y toma de decisiones mediante la primera capa del entorno presentada en la sección 3. Sin embargo, y dada la complejidad inherente de estos sistemas, se ha ampliado el entorno con una segunda capa especializada en SFF.

4.1. MIKRON

MIKRON (*Manufacturing Intended KRON*) [11,13,20] es el resultado de la ampliación de KRON con un conjunto de objetos y primitivas específicas del dominio de los SFF. El modelo de coordinación de un SFF se construye en torno a varios tipos de primitivas de representación: recursos, materiales, productos, operaciones y planes de proceso. MIKRON aporta también una metodología específica de modelado. En ella se contemplan los sistemas de fabricación desde dos puntos de vista diferentes:

1. *Funcional u orientado a las actividades.* Se realiza la descripción de las actividades y operaciones y su encadenamiento para realizar la producción.
2. *Estructural u orientada a los recursos.* Se describen los modelos de funcionamiento de los recursos de producción y las interconexiones entre ellos.

Las primitivas de representación de MIKRON se implementan mediante objetos prototípicos predefinidos que se organizan en jerarquías de especialización, ampliables por el diseñador, y que parten de las jerarquías existentes en KRON. Son un total de cinco jerarquías, como se muestra en la figura 6:

— Los recursos de fabricación se hallan representados en la jerarquía cuya raíz es **Manufacturing-Resource**.

— La jerarquía descendiente de **Manufacturing-Marking-Object** contiene los prototipos de los materiales y productos que pueden circular por un SFF y las operaciones que se pueden realizar.

— Los descendientes de **Manufacturing-Action-Object** son los prototipos de los objetos de acción necesarios para modelar las RAN existentes tanto en los recursos de producción como en las operaciones.

— En la jerarquía **Auxiliar-Process** se incluyen los modelos de algunos procesos no principales, que pueden resultar interesantes en un determinado modelo de un recurso (como modelos de roturas, etapas de puesta en marcha, etc.).

— Los objetos de la jerarquía descendiente de **Manufacturing-Aggregation** permiten el diseño con diferentes niveles de agregación (sectores, celdas, máquinas, etc.).

En la figura 7 puede observarse, como ilustración, parte de la jerarquía descendiente de **Manufacturing-Resource**. Dicha jerarquía contiene los objetos de estado que modelan diferentes recursos que pueden aparecer en un SFF.

Los objetos de la jerarquía de recursos de manufactura heredan las redes subyacentes de objetos que modelan procesos básicos: secuencia, sincronización, decisión, etc. Estas redes, así como otro tipo de informaciones, son especializadas al descender por la jerarquía. A modo de ejemplo, en la figura 8 puede observarse el conjunto de objetos que forman el modelo de una máquina genérica de transformación y en la figura 9 la RAN subyacente.

Todos estos objetos poseen métodos que permiten actualizar el contenido de sus atributos, conocer propiedades de los mismos, indicar cómo se pueden sincronizar entre sí, etc. Además de su comportamiento dinámico, en los objetos de estado se recogerán aquellas características físicas y técnicas relevantes, así como la recopilación de información histórica y predictiva, necesaria en aplicaciones de simulación y planificación.

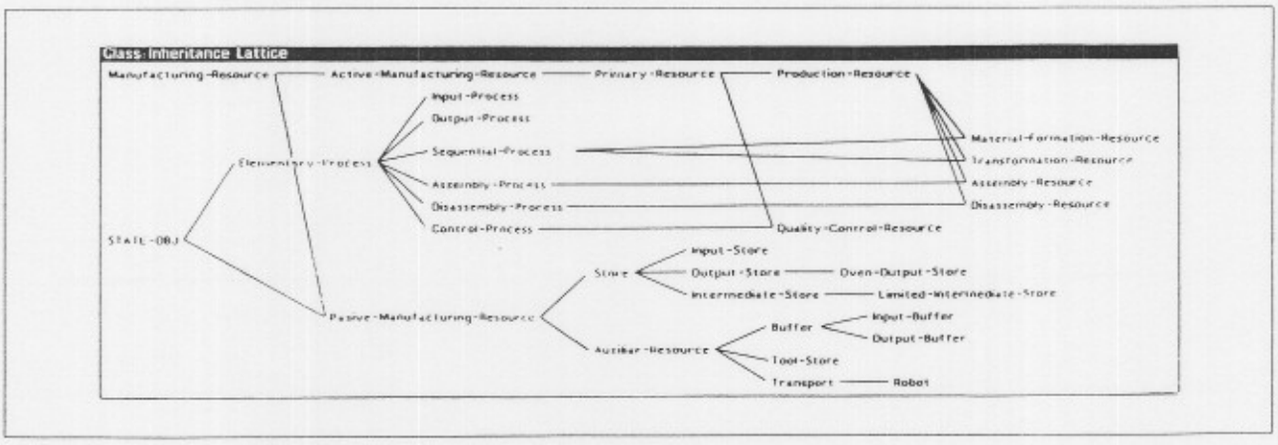


Figura 7. Jerarquía de recursos de manufactura.

```
( Transformation-Machine
    is-a :  Transformation-Resource Machine
    states :  loaded unload-ready available-capacity
        ; state slots
        loaded :
        unload-ready :
        available-capacity :
    actions :  load process unload   ; action slots
        load :  load-M
        process :  process-M
        unload :  unload-M
    ; predicted information and data collection
    pending-operations :
    statistics :
    max-part-size     ; physical characteristics
    constraints :
    set-up :
    average-processing-time :
    precision-level     ; abstraction hierarchy relations
    has-resources :
    resource-of :
    type-of-synchro     ; interface features
    input-synchro :
    output-synchro :
    icon :     ; graphic representation characteristics
    wickname :
    comment :     )

(load-M
    is-a :  Action-Obj
    pre-net-relations : (& available-capacity AFtake)
    post-net-relations : (& loaded AFobj)
    associated-data : <obj> <op>
    predicate : (resource <op>) = &
    ao-conflict-of : * )

(unload-M
    is-a :  Action-Obj
    pre-net-relations : (& unload-ready AFobj)
    post-net-relations : (& available-capacity AFput)
    associated-data : <obj>
    predicate : t
    ao-conflict-of :  )

(process-M
    is-a :  Action-Obj
    pre-net-relations : (& loaded AFprod)
    post-net-relations : (& unload-ready AFnextop)
    associated-data : <obj> <op>
    predicate : t
    ao-conflict-of :  )

AFobj : (s-name <obj>)
AFtake : (s-name one)
AFput : (s-name one)
AFprod : (s-name <obj>; operation <op>)
AFnextop : (s-name <obj>; operation (next <op>))
```

Figura 8. Frames del conjunto de objetos que representan a una máquina genérica de transformación. De Machine, heredera el nivel de agregación.

Para la creación del modelo deseado de la planta se sigue el siguiente proceso: 1) se crean ejemplares de los prototipos de los recursos que componen el sistema a modelar y se rellenan los contenidos de los atributos necesarios, y 2) se relacionan los ejemplares creados mediante la sincronización de objetos de acción que modelan la misma actividad.

Cuando un material entra en el proceso productivo, se le considera un *producto* con un *plan de proceso* asociado. Tanto materiales como productos se modelan mediante objetos de marcado que recorren la red que modela la planta de producción. Entre otros, los principales datos que contiene un producto son: plan de proceso asociado, estado de fabricación en el que se encuentra, operación que se está realizando en este momento sobre él, datos sobre la demanda y, en caso de ser un ensamblado, los componentes, materiales, etc.

Las *operaciones* son las actividades que se realizan sobre los productos en el proceso productivo. Estas pueden suponer una transformación del producto, como en el caso de las operaciones de manufactura, o no, como en las operaciones de transporte o almacenaje. En MIKRON una operación tiene una doble perspectiva: como objeto de estado y como objeto de marcado. Como objetos de estado representan la dinámica que la operación tiene como parte de un plan de proceso. Las RAN que las modelan provienen de los procesos elementales como en el caso de los recursos. Los planes de proceso se forman mediante la sincronización de las mismas. Como objetos de marcado recorren la RAN que representa el plan de proceso, indicando con su información (producto sobre el que se está realizando, recurso en el que se está llevando a cabo, etc.) el estado actual de fabricación de cada producto. En la figura 10 se representa la jerarquía básica de operaciones.

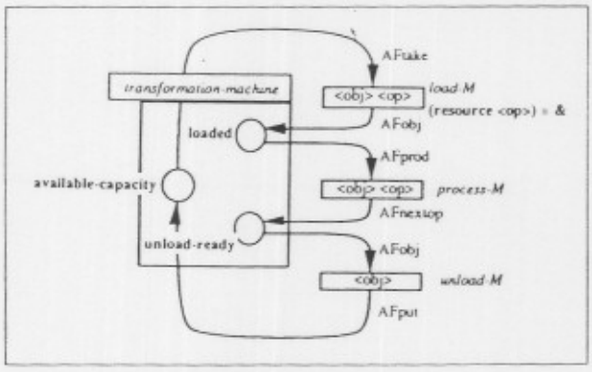


Figura 9. RAN subyacente de una máquina de transformación.

La construcción de un modelo de plan de proceso se realiza de forma similar a como se construye el modelo de la planta: 1) se crean ejemplares de los prototipos de operaciones que vayan a formar un plan, y 2) se sincronizan estas operaciones, formando la secuencia de operaciones deseada.

El modelo de coordinación de un SFF debe reunir las restricciones de la perspectiva estructural, correspondiente al modelado de los recursos y su sincronización, y de la funcional, correspondiente a los planes de proceso, en un único modelo. Para ello se sincronizará el modelo del sistema de fabricación con los planes de proceso. De esta forma ambas perspectivas evolucionarán acompasadamente.

MIKRON proporciona la posibilidad de crear modelos a distintos niveles de abstracción. Existen objetos para representar entidades a distintos niveles de abstracción (máquina, celda, sector y planta) y mecanismos para relacionar un objeto con sus objetos componentes al nivel de abstracción inferior. Así por ejemplo, el valor de la capacidad de una celda puede ser inferido a partir de las capacidades de las máquinas que la componen.

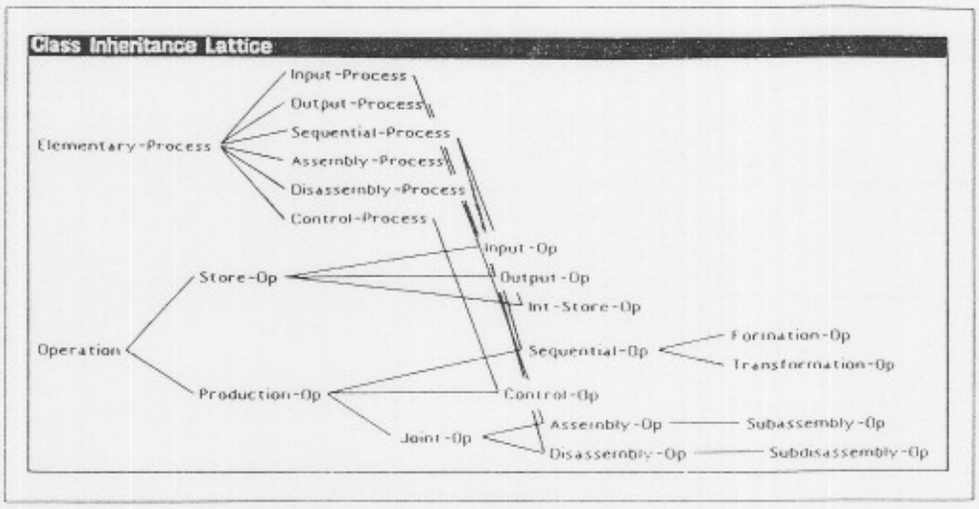


Figura 10. Jerarquía de operaciones.

4.2. Simulador

El objetivo de este módulo es facilitar el análisis del comportamiento de un SFF controlado utilizando un modelo de control diseñado con el entorno. Mediante MIKRON es posible diseñar el modelo de coordinación de un SFF que, aumentado con una estrategia de toma de decisiones (cuyo diseño habrá sido asistido por el módulo específico que se presenta en la sección 4.3) constituye el modelo de control del SFF. Dicho modelo, ejecutado por el módulo de control, podría ser funcionalmente utilizado (no se consideran aspectos de tiempo real) como un sistema de control que sigue la arquitectura comentada en la introducción. Para realizar el control, el sistema se comunica con la planta de producción mediante un conjunto de señales tales como: órdenes de comienzo de operaciones, mensajes de fin de operación, señales de sensores y señales de alarma y error.

El simulador del entorno de diseño tiene como objetivo sustituir a la planta real de producción, simulando el intercambio de señales con el modelo de control. El simulador está realizado mediante técnicas de simulación de eventos discretos y utiliza el típico mecanismo de agenda.

El simulador está controlado por un gestor de alto nivel que permite acceder a diversos componentes: agenda de eventos, generador de parámetros aleatorios, monitor de eventos (componente que constituye la interfase del control con la planta de producción o bien con el simulador), recolector de datos, parámetros de simulación.

4.3. Toma de decisiones de manufactura

La toma de decisiones en manufactura surge como una especialización de la estrategia de resolución de conflictos comentada para la primera capa. En los sistemas de manufactura, el uso de una buena estrategia de toma de decisiones, permite mejorar sensiblemente los objetivos de producción (obtención de productos en la fecha debida, minimización del trabajo en proceso, minimización de «stocks», etc.). Los conflictos, en el caso de un sistema de fabricación, están relacionados directamente con decisiones de planificación de operaciones («scheduling»). El sistema de toma de decisiones dispone de una clasificación general de los conflictos de acuerdo con el problema de planificación que representan. Para cada conflicto, el usuario tiene la posibilidad de elegir dentro de una biblioteca de políticas de control parametrizables la más adecuada para tratar de optimizar los objetivos de producción deseados. Dichas políticas están construidas en base a las reglas de despacho más comunes.

En la figura 11 puede observarse un ejemplo de situación de conflicto donde es necesario tomar una decisión. Las operaciones **OP12_22, OP22_2** y **OP32_11** compiten por ser ejecutadas en el recurso M1. Dicho recurso únicamente puede procesar una operación cada vez. Por tanto, es preciso decidir el orden en que las operaciones serán procesadas por **M1** de tal forma que se intente optimizar un cierto objetivo de producción. Así por ejemplo, si el objetivo es minimizar el tiempo de proceso de las piezas, el usuario puede optar por la elección de una política del tipo Tiempo de Proceso más Pequeño («Shortest Procesing Time» **OP12_22 —> OP22_2 —> OP32_11**); mientras que si el objetivo fuera satisfacer los pedidos de piezas a tiempo se podría optar por políticas del tipo Fecha Debida más Temprana («Earliest Due Date» **OP32_11 —> OP12_22 —> OP22_2**). Ambas estrategias pueden ser elegidas por el diseñador de entre las existentes en la biblioteca de políticas de control.

Más adaptadas a la arquitectura jerárquica de control para la que se pretende diseñar software, son políticas de control que decidan tomando como referencia un plan de operaciones previamente construido. Actualmente esta posibilidad no está contemplada en el entorno, sin embargo, en un futuro próximo, se pretende incorporar a la toma de decisiones en manufactura un módulo de planificación de operaciones y una biblioteca de políticas de control que sean intérpretes de los planes generados. De esta manera, el software diseñado con el entorno, se adaptará fielmente a la arquitectura de control propuesta.

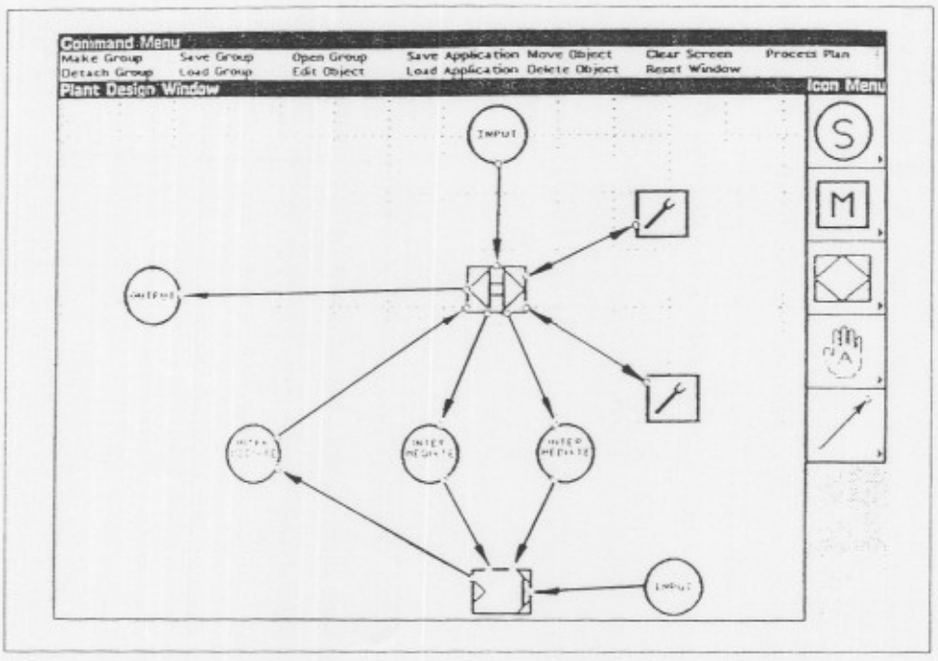


Figura 12. Ventana de GRAMAN que muestra un ejemplo de diseño de celda flexible.

4.4. GRAMAN

GRAMAN, «GRAphic MANufacturing» es la interfase utilizada por el diseñador de modelos de manufactura. Está especializada en SFF y permite la manipulación gráfica de las primitivas que ofrecen los componentes de la segunda capa y, también, la realización de simulaciones animadas. El objetivo de su definición es el de facilitar la labor del diseñador y ocultar, al máximo posible, las características internas del modelo. En [19] se hace una definición preliminar de GRAMAN como una herramienta autónoma. En [20] se redefine GRAMAN como una interfase para MIKRON y en este trabajo se amplia su funcionalidad como interfase para la segunda capa del entorno.

Siguiendo la metodología presentada en el punto anterior, el modelo de coordinación de un SFF se realiza en GRAMAN distinguiendo entre la descripción de la planta de *producción* y la descripción de los *planes de proceso*.

Desde el punto de vista del usuario, GRAMAN permite la descripción de una planta de producción mediante la definición de un conjunto de *bloques constructivos* y sus *interconexiones*. Cada bloque constructivo está asociado a un recurso del sistema.

En la figura 12 puede observarse un ejemplo de modelo de celda flexible realizado con GRAMAN. Cada recurso de un SFF se representa gráficamente mediante un bloque constructivo o icono. En su interior puede colocarse un subicono que especifica la funcionalidad básica del recurso representado. La colocación de un bloque constructivo en el plano de representación supone la creación de un ejemplar del prototipo deseado. Cada objeto contiene la información sobre su representación gráfica en GRAMAN. Los elementos de los cuales se pueden crear instancias son accesibles a través de una serie de menús. Dichos menús se caracterizan por la sencillez de su modificación, se reconfiguran reproduciendo las jerarquías de recursos definidas en un momento dado.

Las entradas o salidas del recurso son representadas mediante los puntos de intercambio; círculos situados en el perímetro del icono y asociados cada uno de ellos a un objeto de acción. Las relaciones estructurales son arcos orientados cuyo origen y destino son puntos de intercambio de bloques diferentes. Definen los posibles caminos del flujo de productos entre los recursos del sistema. El establecimiento de una relación estructural entre dos recursos supondrá, internamente, sincronización de dos objetos de acción. Uno modelará la descarga de uno de los recursos y el otro la carga del otro.

En GRAMAN los planes de proceso se diseñan gráficamente mediante redes de Petri ordinarias. Se ha optado por esta aproximación debido a la sencilla estructura causal que suelen tener los planes de proceso y a la suficiente expresividad gráfica de la red de Petri para estos casos. De igual forma que en el diseño de una planta de producción, el diseño de un plan de proceso provoca la generación automática de un modelo MIKRON interno.

La mencionada representación gráfica del modelo de coordinación de un SFF es utilizada para presentar el estado de dicho modelo y sus cambios en la animación de simulaciones. En esta presentación gráfica se hace abstracción del modelo interno y se utilizan términos del dominio de aplicación de los SFF.

GRAMAN aporta también ventanas especializadas para: la visualización y manipulación textual de objetos, visualización y manipulación de jerarquías, diseño de estrategias de decisión, etcétera.

5. CONCLUSIONES

En este trabajo se ha presentado un entorno de diseño de software de control jerárquico de SFF, actualmente en desarrollo por el Grupo de Ingeniería de Sistemas e Informática del Centro Politécnico Superior de la Universidad de Zaragoza. El entorno está estructurado en dos capas, una genérica de diseño de sistemas de eventos discretos y otra especializada en SFF, que está construida sobre la anterior. Este tipo de estructura permite que la primera capa sea utilizada como una herramienta autónoma y como núcleo en entornos de diseño para otros dominios de aplicación.

Las técnicas de Inteligencia Artificial para la representación del conocimiento basadas en «frames» y el formalismo de las RAN para el modelado de sistemas dinámicos y concurrentes resultan complementarias a la hora de la representación de sistemas dinámicos y concurrentes. La utilización de la programación orientada a objeto proporciona una base para la metodología de modelado, facilitando la representación modular mediante la definición de jerarquías de especialización y la reusabilidad de los modelos mediante la creación sucesiva de ejemplares de los mismos. El resultado de la integración de todas estas ideas es KRON, que es el esquema de representación del conocimiento de la primera capa. Dicha capa aporta también componentes que permiten la ejecución de los modelos construidos y el diseño de estrategias de decisión para aquellos puntos del modelo donde sean necesarias. El manejo de todas las primitivas de esta capa es facilitado por una interfase gráfico-textual.

Dada la complejidad inherente de los SFF, se ha ampliado el entorno con una segunda capa especializada en ellos. MIKRON (Manufacturing Intended KRON) es el esquema de representación del conocimiento de esta capa, resultado de la ampliación de KRON con un conjunto de primitivas específicas del dominio de los SFF: recursos, materiales, productos, operaciones y planes de proceso. MIKRON aporta también una metodología específica de modelado que contempla los sistemas de fabricación desde dos puntos de vista diferentes que consideran por una parte las restricciones de los recursos de la planta de producción, y por otra, las restricciones de los planes de proceso. Otro componente de esta segunda capa está constituido por un sistema de toma de decisiones que permite el diseño y utilización desde reglas de despacho hasta estrategias de interpretación de planes de operaciones. La segunda capa se completa con un simulador que permite verificar el comportamiento del modelo de control diseñado y con una interfase, GRAMAN (GRAphic MANufacturing), que facilita la utilización de las primitivas de la capa y permite visualizar de forma gráfica y animar los modelos construidos.

En el futuro se pretende ampliar la funcionalidad del entorno con:

- La implementación de algoritmos de análisis formal de la red de Petri de alto nivel subyacente en los modelos que permitan detectar malos funcionamientos tales como bloqueos.
- Generadores de código tiempo real que permitan obtener programas de control a partir de los diseños realizados, en la línea de [20].
- Un sistema de ayuda al diseño que guíe al diseñador y le informe del alcance de sus acciones (p.e., introducción de bloqueos).
- Construcción de un planificador de operaciones de manufactura.

Se ha desarrollando un prototipo del entorno presentado sobre LOOPS en una estación de trabajo 1186 de Xerox. El entorno definitivo, con toda su funcionalidad, se está desarrollando sobre KEE en una SPARC-station de SUN.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por los proyectos IT 10/91 del CONAI, de la Diputación General de Aragón, y ROB 91-0949, de la Comisión Interministerial de Ciencia y Tecnología.

BIBLIOGRAFIA

[1] Acock, M., y Zemel, R.: «Dispatcher: AI Software for Automated Material Handling System». *ULTRATECH-AI in Manufacturing Conference*, Society of Manufacturing Engineers, Long Beach, CA, septiembre 1986.

[2] Al-Jaar, R. Y., y Desrochers, A. A.: «A Survey of Petri Nets in Automated Manufacturing Systems». *Proc. of 12th World Congress on Scientific Computation*, IMACS, vol. 3, pp. 503-510. París 1988.

[3] Colom, J. M., Silva, M., y Villarroel, J. L.: «On Software Implementation of Petri Nets and Coloured Petri Nets Using High-Level Concurrent Languages», *Proc. 7th European Workshop on Application and Theory of Petri Nets*, pp. 207-241. Oxford, julio, 1986.

[4] Fox, M. S.: «Constraint Directed Search: A Case Study of Job-Shop Scheduling». *PhD dissertation, Computer Science Dep.*, Carniage Mellon University. Pittsburgh., PA (USA), 1983.

[5] Gentina, J.C.; Bourey, J.P., y Kapusta, M.: «Coloured Adaptive Structured Petri-Net: A Tool for the Automatic Syntesis of Hierarchical Control of Flexible Manufacturing Systems». *Computer-Integrated Manufacturing Systems*, vol. 1, pp. 39-47, febrero 1988.

[6] «Invited Sessions on Petri Nets and Flexible Manufacturing». *Proc. 1987 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 999-1018 y 1160-1186. Raleigh, North Carolina, marzo-abril 1987.

[7] Jensen, K., y Rozenberg, G. (eds.): *High-level Petri - Nets. Theory and Application*. Springer-Verlag, Berlín, 1991.

[8] Lawrence, S., y Morton, T.: «PATRIARCH: Hierarchical Production Scheduling». *Symposium on Real Time Optimization in Automated Manufacturing Facilities*, Nationale Bureau of Standards, Gaithersburg, MD, enero 1986.

[9] Martínez, J.; Muro, P. R., y Silva, M.: «Modeling, Validation and Software Implementation of Production Systems using High Level Petri Nets». En [6] pp. 1180-1185. 1987.

[10] Martínez, J.; Muro, P. R.; Silva, M.; Smith, S.F.; Villarroel, J. L.: «Merging Artificial Intelligence Techniques and Petri Nets for Real Time Scheduling and Control of Production Systems». In *Artificial Intelligence in Scientific Computation: Towards second generation systems*. R. Huber y otros (eds.), S.C. Baltzer AG, Scientific Publishing Co., pp. 307-313, IMACS, 1989.

[11] Muro, P. R.; Villarroel, J. L.; Martínez, J., y Silva, M.: «A Knowledge Representation Tool for Manufacturing Control Systems Design and Prototyping». *Proc. of INCOM'89, 6th IFAC/IFIC/IFORS/IMACS Symposium on Information Control Problems in Manufacturing Thechnology*. Madrid, septiembre 1989.

[12] Muro, P. R., y Villarroel, J. L.: «KRON: Redes Orientadas a la Representación del Conocimiento». *Actas de la III Reu-nión Técnica de Inteligencia Artificial, AEPIA'89*. Madrid, noviembre 1989.

[13] Muro, P. R.: «Aplicación de Técnicas de Inteligencia Artificial al Diseño de Sistemas Informáticos para el Control de Sistemas de Producción». Tesis doctoral. Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, abril 1990.

[14] Sathi, A.; Fox, M. S., y Greenberg, M.: «Representation of Activity Knowledge for Project Management.» *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. PAMI-7, núm. 5, septiembre 1985.

[15] Silva, M., y Valette, R.: «Petri Nets and Flexible Manufacturing.» In *Advances in Petri Nets 1989*, pp. 374-417, LNCS 424, Springer-Verlag, 1989.

[16] Smith, S. F.; Fox M. S., y Ow, P. S.: «Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems». *AI Magazine*, pp. 45-60, Fall, 1986.

[17] Smith, S. F.: «A Survey of AI Based Scheduling Systems». *Fall Industrial Engineering Conference*. Boston, diciembre 1986.

[18] Valette, R., y Bako, B.: «Software Implementation of Petri Nets and Compilation of Rule-based Systems». *11th International Conference on Application and Theory of Petri Nets*, París 1990.

[19] Villarroel, J. L.; Martínez, J., y Silva, M.: «GRAMAN: A Graphic System for Manufacturing System Design». *Proc. of the IMACS Symposium on System Modelling and Simulation*. Cetraro, pp. 311-316, Italia, septiembre 1988.

[20] Villarroel, J. L.: «Integración Informática del Control de Sistemas de Fabricación Flexible». Tesis doctoral. Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, septiembre 1990.

Colabora con tu revista:
envía artículos breves con puntos de vista nuevos y temas de debate.

Ninguna revista puede mantenerse viva sin la aportación de todos

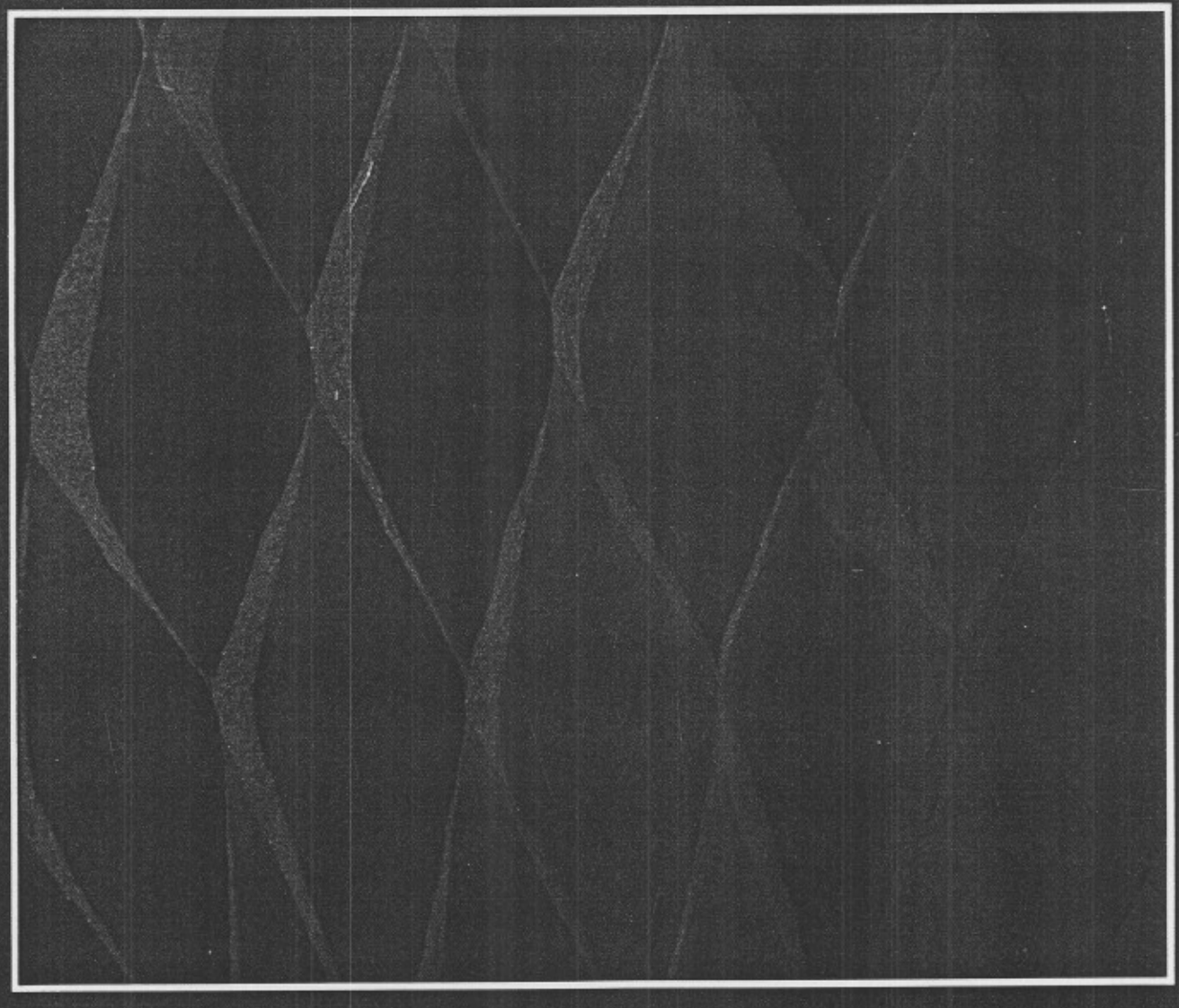
INFORMATICA Y AUTOMATICA

MEMORIA ASOCIATIVA PARA RECONOCIMIENTO DE PATRONES

ENTORNO DE SISTEMAS DE CONTROL EN CIM

CALIDAD DE SOFTWARE E ISO 9000

NOTICIAS AEIA: INFORMES CURSOS DE VERANO



Vol. 26 Núm. 3

Septiembre 1993