Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Iribarne, Luis F.; Flores, Isabel M.; Bienvenido, J. Fernando
Una Metodología de Distribución de Procesos en Problemas de Simulación
Computación y Sistemas, vol. 2, núm. 4, abril-junio, 1999, pp. 202-212
Instituto Politécnico Nacional
Distrito Federal, México

Computación y Sistemas Vol. 2 No. 4 pp.202 -212
© 1999, CIC - IPN. *ISSN 1405-5546* Impreso en México

Una Metodología de Distribución de Procesos en Problemas de Simulación

L. F. Iribarne, I. Flores, J. F. Bienvenido
Departamento de Lenguajes y Computación
Universidad de Almería, España
liribarn@ualm.es

Artículo recibido el 15 de noviembre, 1998; aceptado el 19 de abril, 1999

Resumen

En este trabajo se muestra una metodología de distribución de procesos en problemas de simulación, basada en la definición de unas jerarquías de tareas y de elementos sobre los que se aplican dichas tareas. Se realiza un análisis de la capacidad de distribución de los problemas utilizando como base el conocimiento de la aplicación convenientemente modelado. Se propone la implementación de forma distribuida sobre una plantilla de gestión de los procesos y una metodología para desarrollar un sistema distribuido en RDP. La aportación propuesta tiene como marco de aplicación el campo de la simulación de sistemas complejos, compuestos por subsistemas diferenciados con grandes requisitos de cálculo.

Palabras clave:

Sistema distribuido, sistema de agentes, resolución distribuida, simulación distribuida.

1 Introducción

En estos últimos años se está experimentado un fuerte avance en el campo de la *informática distribuida*, avance que tiende a ser progresivo, reformista y apasionante en los próximos años a la vez que consolidador, conservador e incierto. Progresivo, a la vista del desarrollo tecnológico que ha habido y se prevé que habrá en todas las áreas de la informática distribuida (hardware, software y middleware) y que ha permitido, y está permitiendo, la aparición de nuevos escenarios donde consolidar y ampliar técnicas, metodologías y entornos distribuidos, que en muchos casos obliga a los investigadores a tener que reformar su forma de pensar, sus técnicas y métodos de trabajo, e incluso, a adaptar sus resultados de investigación acorde al ritmo de expansión que impone la comunidad informática.

En este sentido gran parte de los problemas de la ingeniería en *procesos de simulación dinámica* conllevan un fuerte nivel de distribución que se manifiesta al descomponerlos en etapas o tareas que permiten un cierto grado de superposición (muchas de las cuales pueden implementarse como agentes con cierta autonomía de ejecución). Por otra parte, las técnicas de decisión basadas en simulación dinámica deben incorporar ciertos mecanismos de monitorización, modificación de planes de ejecución y alternativas de evaluación (Hybinette y Fujimoto, 1998).

En este trabajo se propone *una metodología de distribución de procesos* para la resolución de problemas complejos en tareas de simulación y diseño propias de la ingeniería. La metodología propuesta supone la construcción de planes dinámicos relativos a tareas y elementos de trabajo. El campo de aplicación de la metodología es la simulación de sistemas

complejos (Bienvenido, Iribarne, *et al.*, 1997; Kirihara, Aoyagi, *et al.*, 1997; Lalis y Menhart, 1995), con múltiples fases de trabajo y/o componentes, susceptibles de ser modelados mediante diversos subsistemas. Estos subsistemas podrán corresponderse con tareas de transformación o de evaluación de estados, y podrán implementarse como elementos con un amplio grado de independencia y autocontrol.

Los objetivos de los distintos elementos podrán estar más o menos interrelacionados, pudiendo llegar a implementarse como *agentes independientes*. Campos específicos de aplicación de la metodología son: el modelado de sistemas biológicos a pequeña, mediana o gran escala (p.e. genética, plantas o ecosistemas), sistemas sociales (p.e. simulación de comportamientos en grupos sociales y análisis de mercados) o industriales con múltiples estados y/o productos (p.e. simulación de aeronaves, modelado de estructuras y fabricación personalizada 'mass customization' (Tseng y Jiao, 1997). En nuestro caso se está aplicando a la *simulación* del microclima y de la masa vegetal dentro de *invernaderos mediterráneos*.

El trabajo ha sido desarrollado inicialmente dentro del proyecto DAMOCIA: *Diseño Asistido Mediante Ordenador para la Construcción de Invernaderos Automatizados* (Bienvenido, Iribarne, *et al.*, 1997; García, Tallón, *et al.*, 1997), financiado por la Unión Europea dentro del marco de proyectos ESPRIT (Acción especial P7510 PACE) y el Ministerio de Industria Español (PATI PC191), mención especial por resultados destacables en innovación tecnológica, CDTI 20 Noviembre de 1997. Extensiones del presente trabajo se están aplicando al proyecto CAMED, financiado por el Ministerio de Industria Español.

En siguientes apartados del presente trabajo presentamos la metodología propuesta para la distribución de procesos como técnica en resolución de problemas en simulación. A continuación, pasaremos a describir los aspectos y detalles de implementación de dicha metodología y presentaremos un ejemplo de aplicación, DAMOCIA-SIM, como parte del proyecto general anteriormente citado. Finalizaremos con un apartado de conclusiones y trabajos futuros donde exponemos posibles extensiones de las aportaciones.

2 Metodología

Hasta hace poco tiempo los sistemas para la resolución distribuida de problemas (RDP) se caracterizaban por ser un sistema de actuación concurrente entre diferentes nodos de la red que se coordinan para resolver un problema concreto, controlados por regla general, por un gestor centralizado que organiza la secuencia de ejecución. Ejemplo de esto es la arquitectura de pizarra del sistema Hersay (Erman *et al*, 1980). Los distintos componentes de un RDP se pueden desglosar en (Yang y Zhang, 1996):

(a) *Descomposición del problema* general en diferentes subproblemas o tareas básicas.
(b) *Asignación de las tareas y recursos* del sistema a entidades independientes (agentes) que funcionan en nodos diferentes de la red.
(c) *Resolución de las tareas*. Cada agente del sistema intentará resolver las tareas encomendadas.

Por otra parte, las entidades autónomas del sistema (agentes) se programan para que coexistan dentro de un marco distribuido que cumpla parte de las propiedades de cualquier sistema de agentes, algunas de las cuales se caracterizan por (Durfee y Rosenschein, 1994):

(a) *Heterogeneidad de agentes*: los módulos autónomos (agentes) pueden estar diseñados bajo arquitecturas, representaciones internas y lenguajes de comunicación diferentes.
(b) *Homogeneidad de intereses*: los módulos autónomos (agentes) son programados de tal forma que colaboran en beneficio propio y que repercutan en el bien común del sistema (resolver el problema).

2.1 Esquema de la Metodología

Teniendo presente estas características, la metodología propone un mecanismo o técnica para crear un sistema para la resolución distribuida de problemas basado en el comportamiento típico de un sistema de agentes. Esta metodología está pensada, por ejemplo, para la distribución de tareas en problemas complejos típicos de la ingeniería, en donde la carga de cálculo sea considerablemente alta y las partes en las que se descompone el problema están débilmente acopladas, es decir, permitan cierto grado de paralelización para la resolución del mismo. En este sentido, podemos pensar en problemas concretos definidos por el usuario desde una interfaz donde introducir los parámetros mínimos para llevar a cabo la descomposición jerárquica del problema en un conjunto de tareas que se distribuirán entre los agentes del sistema.

Esta metodología, al igual que la mayoría de los procesos de desarrollo de software, tiene como base el esquema del ciclo de vida clásico de la ingeniería del software (Figura 1). En nuestro caso, se trata la capacidad de distribuir el procesamiento a nivel de diseño, definiendo planes de ejecución a nivel de implementación mediante la arquitectura DACAS (Bienvenido y Marin, 1997) y a nivel de ejecución mediante la distribución efectiva, en tiempo real, del tratamiento en procesos individuales.

A nivel de diseño, detectamos dos fuentes de distribución del procesamiento, por la presencia de múltiples etapas de ejecución (tareas) y la existencia de múltiples elementos de cómputo. La capacidad de distribución potencial vendrá dada por el nivel de acoplamiento interno de las tareas y de los elementos (a mayor acoplamiento menor nivel de distribución final).

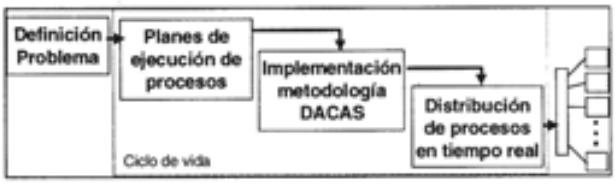

Figura 1: Ciclo de vida

2.2 Los Planes de Ejecución e Implementación

Para la *definición de los planes de ejecución* y posterior *implementación* utilizamos la arquitectura denominada DACAS (Distributed Architecture for Changeable and Adaptable Simulation). En siguientes párrafos hacemos una breve introducción a dicha arquitectura (extensiones de este trabajo se puede encontrar en Bienvenido y Marin, 1997). Esta arquitectura se caracteriza por ser una arquitectura reactiva que permite seleccionar un método de resolución de problemas a partir de una colección de ellos y dispone de una planificación deliberativa, permitiendo el cambio del plan en tiempo de ejecución.

Los planes de ejecución constituyen una evolución de la definición de comportamiento incorporada en cada caso, reduciendo siempre la capacidad de distribución a través de procesos de evaluación. Los planes se describen mediante cuatro símbolos básicos (Figura 2):

(a) secuencial, un proceso $\mathbf{P}_{k-1}$ no tendrá sus entradas disponibles, y por tanto no se podrá ejecutar, hasta que el proceso P$_k$ las genere;

(b) selectiva/alternativa, se escogerá uno de los procesos disponibles dentro de un conjunto de procesos a ejecutar, incorporados en una lista variable y accesible por el proceso selectivo (S);

(c) iterativa, las condiciones o premisas de iteración serán definidas por un proceso R;

(d) paralela, las entradas de los procesos podrán ser comunes, compartidas o diferentes. Un proceso previo (P) aceptará la información reflejada en su estado y la desglosará en bloques de información que servirán de entrada para cada uno de los procesos en espera.

2.3 Distribución de procesos

Para determinar el plan de *distribución de procesos* (motivo del presente trabajo) nos basamos en una arquitectura en dos niveles (Figura 3), que parte de la definición de dos conjuntos: (a) un conjunto de tareas, y (b) un conjunto de elementos de cómputo a partir de la implementación anterior. En el primer nivel se genera un esquema de acoplamiento tanto para el conjunto de las tareas como para el de los elementos de cómputo, y un primer plan de distribución para la ejecución de procesos, basado en el esquema de acoplamiento del conjunto de tareas. En el segundo nivel interviene un conjunto de variables que definen el entorno del problema y que modificarán el plan de ejecución de procesos determinado en el nivel anterior.

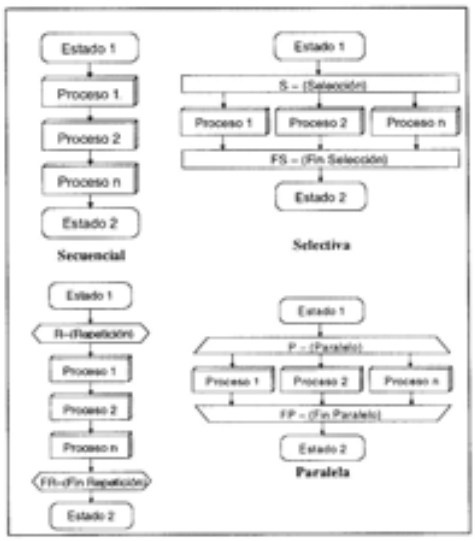

Figura 2: Símbolos implementación

El modelo propuesto presenta varios niveles de distribución de los procesos. En primer lugar muestra un nivel de distribución de las tareas a ejecutar que independiente de los objetos tratados queda representado en el plan de ejecución de tareas inicial. En segundo lugar, aparece un nivel de distribución propio de la descomposición en elementos (poco acoplados) del objeto de simulación; éste queda representado en el plan de descomposición de elementos. La composición de estos dos niveles de distribución genera la capacidad de dis-

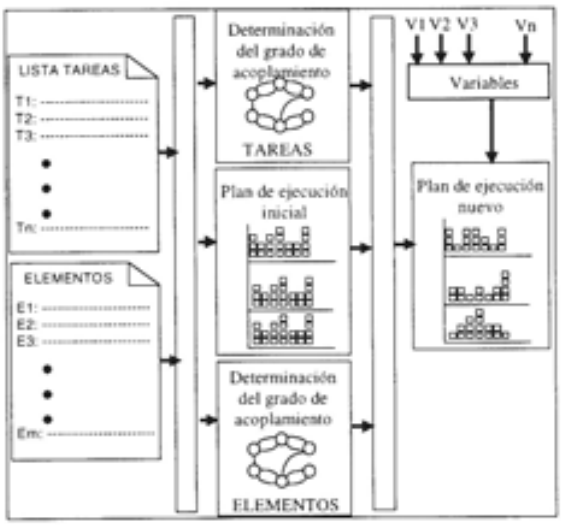

Figura 3: Arquitectura de distribución en dos niveles

tribución primaria o plan de ejecución inicial que gestiona un Proceso de control (según arquitectura DACAS). En el segundo nivel, este plan se refina teniendo en cuenta los elementos de computación disponibles (vector multidimensional $\{V_1, \ldots, V_n\}$), y el acoplamiento entre tareas y elementos, generando un nuevo plan de ejecución, que plantea la existencia de colas de procesos a ir ejecutando sobre los diversos elementos de computación disponibles.

Definición 1. *Dadas dos listas iniciales, una de tareas T={t1,t2,...,tn} y otra de posibles elementos de cómputo sobre los que aplicar las tareas, E={e1,e2,...,em}, los miembros de cada lista presentarán un grado de acoplamiento (GA) que determinará el nivel máximo de distribución.*

Para ello, hacemos uso de una matriz binaria y un grafo de dependencias obtenido a partir de ésta. En la figura 4 se muestra un ejemplo de matriz de acoplamiento con su grafo asociado para una lista T con 6 tareas. La interpretación de una fila, por ejemplo la 3ra {1 0 1 0 0 0}, será "la ejecución de la tarea 3 dependerá de la tarea 1 y de sí misma"; todas las tareas dependen de sí mismas (sus parámetros de ejecución).

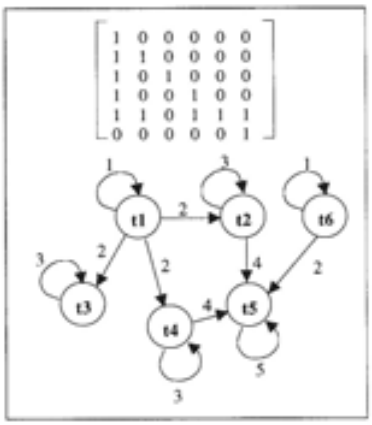

Figura 4: Acoplamiento tareas

Definición 2. *El grado de pertenencia (GP) de una tarea i se entenderá como la influencia que tiene dicha tarea i sobre aquellas tareas dependientes de la misma.*

Esto queda definido con un vector binario que coincide con los elementos de acoplamiento de la fila i de la matriz de acoplamiento, es decir, $GP_i=GA_i$. Así por ejemplo, para el caso de la figura 3, el grado de pertenencia de la tarea 3 es:

$$\mathbf{GP_3}=<1,0,1,0,0,0>$$

Definición 3. *El grado de acoplamiento de una lista de tareas T={t1,t2,...,tn} quedará definido por un conjunto de vectores binarios de pertenencias entre las mismas,*

$$GA = \bigcup_{i=1}^{n} GPi$$

Por otro lado, como se ha indicado, el grado de pertenencia de una tarea queda definido como un vector binario que refleja el número de dependencias directas con el resto de las tareas del plan. Podemos establecer el número total de estas dependencias a partir del orden del grado de pertenencia de una tarea i.

Definición 4. *El orden O(GPi) del grado de pertenencia de una tarea i para una lista de tareas T={t1,t2,...,tn} dada, determinará el número total de dependencias con el resto de las tareas de T, quedando definido de la forma:*

$$O(GPi) = \sum_{j=1}^{n} GP_i(j)$$

Aunque el tratamiento descrito ha sido para una lista de tareas T, el mismo proceso se realiza con los elementos de cómputo, E. De esta forma, se disponen de dos matrices de acoplamiento, T y E. Posteriormente, se hace coincidir cada una de las tareas con cada elemento de cómputo, de forma que cada par <ti,ej> determinará un proceso potencial. Inicialmente, el plan de ejecución de procesos será el mismo para cada elemento de cómputo, por lo que, a partir de la tabla de acoplamiento de tareas y la lista global de elementos de cómputo, se generarán NxM procesos, siendo N y M las dimensiones de las listas T y E, respectivamente.

En principio, se supone que el conjunto de tareas se aplica por igual a cada uno de los elementos de cómputo (Figura 5.a); no obstante, el plan de ejecución de procesos se verá modificado por el esquema de acoplamiento de los elementos y por las variables de entrada que definen el entorno del problema (Figura 5.b).

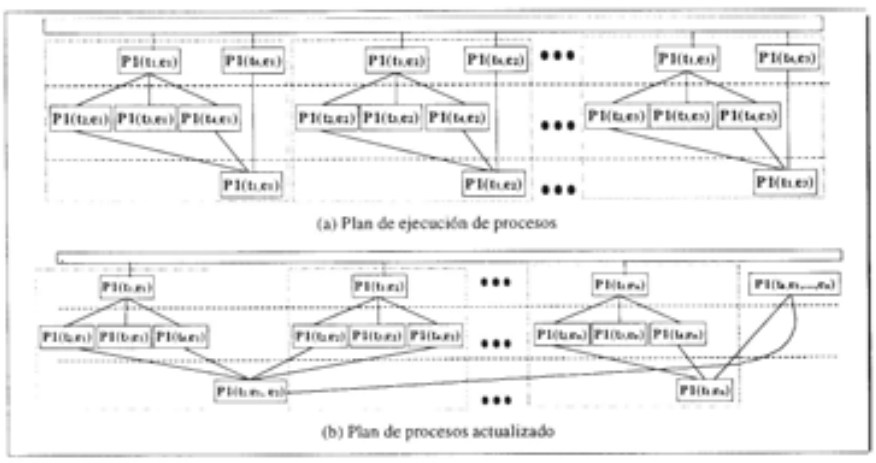

Figura 5: Plan de ejecución de procesos

3 Implementación

Veamos ahora algunos detalles de implementación de la arquitectura de distribución y posteriormente presentaremos un ejemplo de aplicación a partir de DAMOCIA-SIM, una herramienta de simulación de invernaderos para estudiar el comportamiento de los mismos como captadores de energía (se describirá con mayor detalle mas adelante).

3.1 Introducción

Como se ha comentado en la introducción de este trabajo, la metodología propone un mecanismo o técnica para crear un sistema para la resolución distribuida de problemas basado en el comportamiento típico de un sistema de agentes. Puesto que el sistema está pensado para resolver problemas con fuerte carga de cálculo y bajo acoplamiento entre sus tareas, el sistema se desarrolló bajo una red local monolítica para redes Microsoft.

Un ejemplo de problemas con fuerte carga de cálculo y bajo acoplamiento entre sus tareas son todos aquellos relacionados con la simulación en la ingeniería (pe, modelización de invernaderos, aeronaves, etc.) en donde en la mayoría de los casos utilizan cálculo por elementos finitos y gran parte de las tareas se aplican de forma iterativa sobre diferentes partes del objeto o caso a simular, sin tener dependencias las unas de las otras.

En el sistema cohabitan N computadoras, cada una de ellas con un agente autónomo ejecutándose en primer plano, a la espera de recibir un plan de trabajo al cual responder.

Una de las computadoras dispone de un programa con la interfaz de usuario desde donde definir el problema a resolver, problema que será aceptado por otro agente o planificador que se encargará de generar un plan de ejecución en un formato que entenderán el resto de los agentes en espera. Este planificador podrá residir en la misma máquina donde se ejecuta la interfaz de usuario, o en otra máquina distinta.

Una vez definido el plan de ejecución (comentado en el apartado anterior) habrá que decidir cómo distribuir el control de las tareas -reflejadas en el plan general de ejecución- entre los agentes que esperan en cada una de las máquinas del sistema, descrito en apartados anteriores.

Para llevar a cabo la distribución se ha pensado en dos posibilidades, una de ellas ya implementada y experimentada (motivo del presente trabajo) y la otra en fase de análisis. La primera de ellas consiste en hacer la distribución de forma centralizada y la segunda de forma descentralizada. En la distribución centralizada un agente autónomo (organizador) recoge el plan de ejecución generado por el planificador y estudia la forma de distribuir todas las tareas entre los distintos agentes del sistema.

El organizador sólo deberá conocer el número de máquinas (agentes) que están dispuestas a resolver el problema global y el plan de tareas generado por el planificador. Para hacer esto, se ha implementado un algoritmo de distribución que detecta, a partir del plan de ejecución, el nivel de paralelización máximo de dicho plan y efectúa la asignación directa de tareas a cada máquina.

En el caso de la distribución descentralizada, no existirá un gestor que determine la distribución de las tareas del plan, si no que serán los propios agentes del sistema los que decidan qué tareas podrán realizar, o dicho de otra forma, cuando el planificador genera el plan de ejecución, éste (el plan) se ofrece al conjunto de agentes del sistema, y son éstos los que se ponen de acuerdo para organizar y coordinar dicho plan. Como se ha comentado, en el presente trabajo se describirá el modelo centralizado.

3.2 La Arquitectura de Distribución

La arquitectura general del sistema propuesto (ver Figura 7) está compuesta por los siguientes componentes: (1) Un conjunto de k agentes que esperan un plan de ejecución; (2) Un generador de planes, GP (utilizando por ejemplo la metodología DACAS); (3) Un distribuidor de planes, DP; y (4) Un espacio de trabajo, ET.

La arquitectura está diseñada de tal forma que cada elemento de la misma puede funcionar de forma autónoma indistintamente en una única máquina o en N máquinas de la red; la única condición es que todos los componentes deben conocer el nombre del recurso compartido, el espacio de trabajo (ET).

Así, por ejemplo, si decidimos distribuir todas los componentes del sistema en máquinas distintas de la red, necesitaríamos k+4 máquinas (ver Figura 7), aunque esta situación no suele ser la normal. En la mayoría de los experimentos llevados a cabo de este sistema, se asignó cada agente a una máquina diferente y el resto de las componentes a otra distinta.

El flujo de ejecución del sistema es el siguiente, por un lado se activan los k agentes para que lean de un componente de inicio (Ai), el cual contiene el nombre de cuatro componentes que necesita todo agente para su ejecución (ver Figura 7). Los agentes permanecen a la espera hasta que se genere un componente de habilitación para cada uno de ellos. Este componente de habilitación no se genera hasta que esté completo el plan de ejecución y se conozca la forma de distribuir las tareas en el sistema.

Por otro lado, el generador de plan (GP) recoge el código de modelización del plan, utilizando alguna técnica de representación de planes (como la definida por DACAS), y lo transforma a un formato que reconoce el resto de los componentes del sistema. Seguidamente, el distribuidor del plan (DP) detecta la forma de asignar partes del plan a cada uno de los

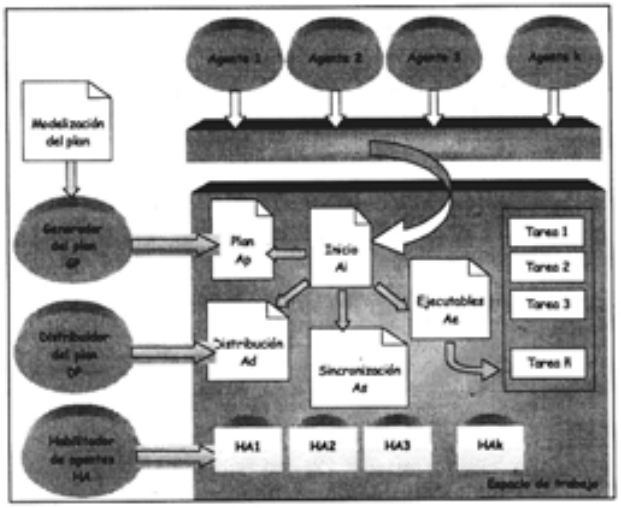

Figura 7: Arquitectura de implementación

agentes en espera, generando un informe de distribución (Ad). Por último, el habilitador de agentes (HA) genera k testigos de habilitación, uno para cada agente en espera. Una vez creados estos testigos en el espacio de trabajo ET, los agentes detectan su aparición y comienzan a funcionar. En primer lugar, todos ellos leen del informe de distribución (Ad) para comprobar qué partes del plan les corresponde.

Una vez organizados, obtienen su parte del plan desde Ap, y comienzan a trabajar. Los componentes se comunican y coordinan a través de un canal de sincronización (As). En este, los agentes irán escribiendo el código de las tareas que han ido finalizando. A partir del componente de ejecutables (Ae), convierten el código de la tarea por el nombre del archivo ejecutable de dicha tarea. Cuando un agente finaliza su parte del plan, eliminará el testigo de habilitación del ET generado por el habilitador de agentes (HA), y permanecerá de nuevo en espera.

3.3 El Algoritmo de Distribución

Veamos a continuación el algoritmo desarrollado para el distribuidor del plan de tareas (distribuidor de procesos). Para ello, supongamos que disponemos de un simple plan de ejecución como el que se muestra en la figura 8.

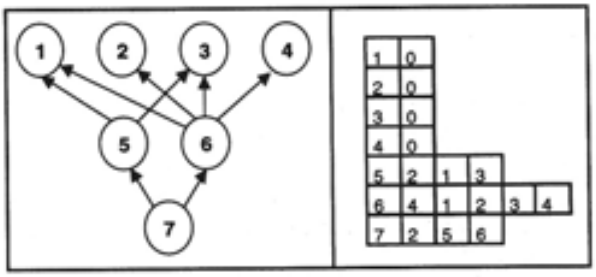

Figura 8: Ejemplo de un plan

En la parte izquierda de la figura 8 se muestra un grado de acoplamiento con dependencias entre tareas de un plan. En la parte derecha se muestra el mismo plan con el formato de tabla correspondiente al componente Ap. En esta tabla, la primera columna indica el código de la tarea, en la segunda columna el número de dependencias (orden) para cada tarea, y el resto de las columnas representan las listas de tareas dependientes para cada una de las mismas. Así por ejemplo, la tarea 5 depende de 2 tareas (la 1 y la 3).

Detectamos las tareas iniciales comprobando cuales de ellas no tienen tareas dependientes, es decir, el segundo campo es un 0. Tras examinar la tabla, comprobamos que existen cuatro tareas iniciales (1, 2, 3 y 4).

Marcamos un primer nivel indicando que se pueden paralelizar hasta 4 (nivel 1 = 4) y anotamos cuáles son (la 1, 2, 3 y 4; ver Figura 9). Posteriormente, examinamos de nuevo en la tabla aquellas tareas que se ven afectadas por la ejecución de las del primer nivel. Comprobamos que sucede esto en 5 y 6, y procedemos a eliminarlas de la tabla, comprobando ahora, si éstas (5 y 6) tienen dependencias. Como no tienen, se marcan dentro del nivel 2 (ver Figura 9). A continuación, se marcan las líneas de la tabla donde aparezcan las tareas del nivel 2, (en este caso sólo 7). Se marcan 5 y 6, y se eliminan. Por último, el nivel 3 solo contiene la tarea 7.

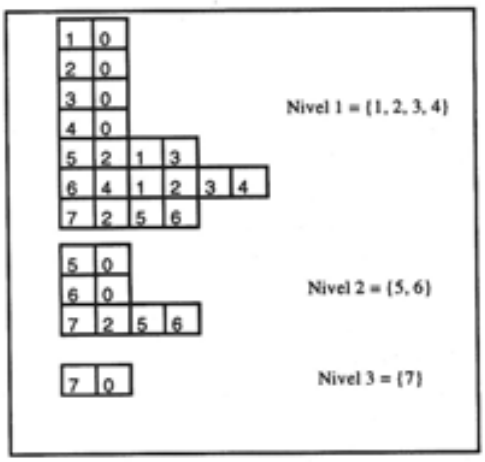

Figura 9: Niveles de distribución

En la figura 10 se muestra el algoritmo de distribución que implementa el comportamiento descrito.

La dimensión de cada nivel determina el grado de paralelización. El grado de paralelización máximo del sistema viene dado por el nivel con mayor dimensión. Por ejemplo, en el caso anterior obtenemos la siguiente tabla:

Nivel	Dimensión	Tareas
1	4	1 2 3 4
2	2	5 6
3	1	7

```
Algoritmo_Distribuido(plan P)
  //nivel_1:tareas_iniciales
  Para i de 1 a tamaño(P)
    Si P(i,2)=0 entonces N(1, j)←P(i,1) y j++ fin_SI
  fin_Para_i
  k←1
  Mientrasexistan tareas ejecutar
    r←1 y i←1
    Si P(i,2)>0 entonces
      Para j de 3 a tamaño(P(i))+3
        Para n de 1 a tamaño(N(k))
          Si P(i, j)=N(k,n) entonces Eliminaelemento(P(i, j) fin_SI
        fin_Para_n
      fin_Para_j
      Si tamaño(P(i))=0 entonces N(k+1,r)←P(i,1) y r++ fin_SI
    fin_Si
    Si i=numerotareas entonces
      i←1 y k++ y r←1
      sino i++
    fin_Si
  fin_Mientras
fin_Algoritmo
```

Figura 10: Algoritmo de distribución

A partir de esta tabla podemos concluir que el número deseado de agentes que pueden intervenir en el sistema son 4 (dimensión máxima), pudiéndose repartir las tareas entre estos de la siguiente forma:

Agente 1: {1, 5, 7}
Agente 2: {2, 6}
Agente 3: {3}
Agente 4: {4}

Se pueden dar casos usuales en los que el grado de distribución del sistema no se corresponda con el caso real, es decir, que existan menos agentes (máquinas) disponibles que nivel de paralelización. En el caso anterior, si el sistema esta limitado a 2 agentes, habrá que determinar algún tipo de algoritmo que decida la forma de distribución. Se podría pensar en asignar tareas de forma equitativa, según aparezcan en la lista. Por ejemplo:

Agente 1: {1, 2, 5, 7} y Agente 2: {3, 4, 6}

4 Ejemplo de Aplicación

En esta sección presentamos un ejemplo de aplicación de la metodología propuesta para el desarrollo de algunos módulos del proyecto DAMOCIA. Para ello, describiremos brevemente el entorno de trabajo de dicho proyecto; no obstante, el lector podrá encontrar numerosas referencias en la literatura (Bienvenido, Marin *et al.*, 1997; Bienvenido, Iribarne *et al.*, 1997; García *et al.*, 1997; Iribarne, Guirado, *et al.*, 1997).

4.1 Marco del Ejemplo de Aplicación

Como mencionamos en la introducción del presente trabajo, DAMOCIA (Diseño Asistido Mediante Ordenador para la Construcción de Invernaderos Automatizados) fue el resultado de un proyecto de investigación financiado por la Unión Europea dentro del marco de proyectos ESPRIT (Acción especial P7510 PACE) y el Ministerio de Industria Español (PATI PC191).

En este trabajo colaboró el Departamento de Lenguajes y Computación de la Universidad de Almería (España) y el centro de investigación FIAPA (Fundación para la Investigación Agraria en la Provincia de Almería), entre otros. Extensiones del mismo se están aplicando al proyecto CAMED, financiado por el Ministerio de Industria Español (CDTI/CICYT 970068). DAMOCIA tenía como objetivo facilitar la adaptación de nuevas estructuras de invernaderos en territorio de la UE tras la aplicación de la norma europea CEN-TC284 GREENHOUSE.

Veamos ahora las razones que motivaron el planteamiento y posterior desarrollo del entorno de trabajo anteriormente citado. Aunque nos centraremos tan sólo en una parte del mismo (Damocia-Sim), el lector podrá encontrar parte del contenido siguiente en la literatura (Iribarne *et al.*, 1997).

4.2 Introducción al Entorno de Trabajo

La mejora de las prestaciones y la reducción de costes de los sistemas informáticos han hecho que cada vez más se utilicen en la simulación de sistemas y procesos complejos. En muchos casos estos procesos de simulación requieren la aplicación de diversos submodelos distintos, relativos a los diversos subsistemas, partes físicas, etapas o subprocesos; submodelos que se basan en principios matemáticos, físicos, químicos, etc., perfectamente diferenciados. A menudo ocurre también que se pueden aplicar unos u otros submodelos en un proceso concreto de simulación debido a la evolución de los conocimientos correspondientes, la mejora de las técnicas o la capacidad de cálculo disponible, los requisitos de exactitud o focalización de los resultados o la disponibilidad de recursos.

Ante esta situación, que se acentúa en los sistemas complejos, pensamos que una solución adecuada puede ser la utilización de arquitecturas de simulación multimodelo distribuidas, en las que los distintos submodelos se implementan de forma independiente en una arquitectura general con un diseño especialmente cuidadoso con las estructuras de datos correspondientes a los resultados parciales de los distintos submodelos.

Dicha filosofía de simulación la hemos aplicado en nuestra herramienta DAMOCIA-Sim. El objetivo fue estudiar el comportamiento de las estructuras de invernadero como

captadoras de energía, sin necesidad de construirlas. DAMOCIA-Sim se complementa con otra herramienta, DAMOCIA-Design, que define las características estructurales del invernadero a modelizar a través de un lenguaje declarativo (García, Tallón *et al.*, 1997).

El método de evaluación del comportamiento de una estructura de invernadero se basa, principalmente, en la simulación de la radiación solar global. A partir de esta información, se evalúa el comportamiento del invernadero frente a la incidencia de la radiación, considerando las características físicas de la cubierta. El objetivo final de este simulador es facilitar la renovación y tecnificación de las estructuras de invernadero, mediante la mejora de sus condiciones microclimáticas.

DAMOCIA-Sim está compuesto por un conjunto de modelos caracterizados por existir entre ellos lo que llamamos 'independencia conectiva'. Cada módulo del modelizador es 'visto' por el resto como una entidad cerrada, totalmente independiente y capaz de realizar la tarea que se le tenga asignada. Sin embargo, esta independencia no es del todo ajena a la secuencia de ejecución del resto de los módulos; existe un 'paso de mensajes' entre módulos, facilitando la comunicación entre los mismos con rutinas para solicitar/conceder información.

Por tanto, DAMOCIA-Sim (en la Figura 11, se muestran algunas pantallas de su interfaz) es un software que implementa un conjunto de modelos caracterizados por definir en su conjunto un sistema multimodelo, distribuido, e incremental.

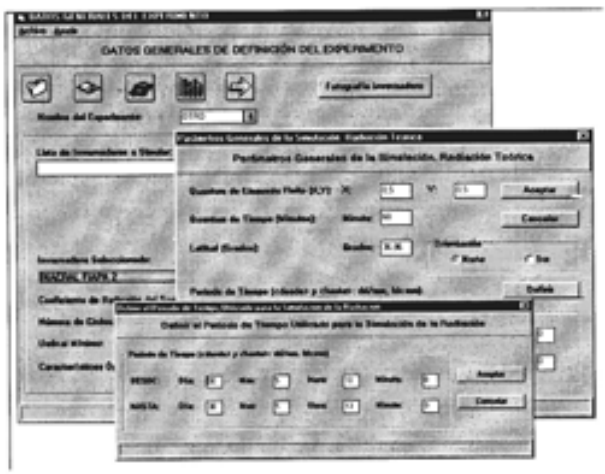

Figura 11: Ejemplos de la interfaz de DAMOCIA-SIM

El propósito es calcular la cantidad de energía que capta un invernadero mediante mapas de radiación en el interior del mismo. Esto se consigue trabajando con elementos finitos, discretizando las cubiertas del invernadero y el interior del mismo (elementos de superficie y elementos volumétricos, respectivamente) y el periodo de tiempo de simulación al que está sometido.

La simulación se lleva a cabo estableciendo los parámetros desde la interfaz de la aplicación (Figura 11). Algunos de estos son: la definición del invernadero, las características ópticas del plástico (índice de refracción, coeficiente de absorción y grosor), tamaño de los elementos finitos en los que se divide el invernadero, periodo de tiempo, intervalo de discretización del tiempo, modelo de absorción del invernadero. Internamente se generan mapas de iluminación de superficies (Figura 12) que en su conjunto determinarán mapas de radiación externos (plástico) e internos al invernadero y que se utilizarán en la aplicación para explotar los resultados.

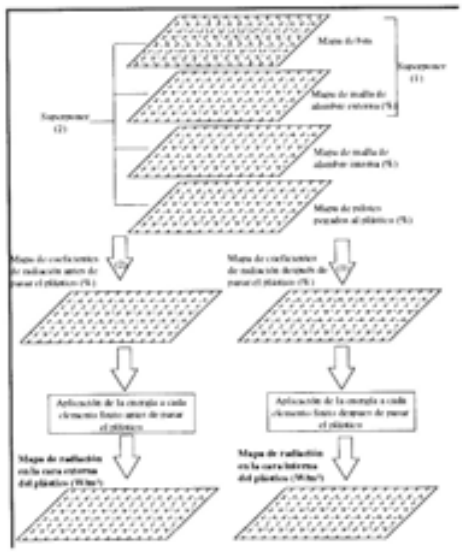

Figura 12: Mapas de iluminación

4.3 El Ejemplo de Aplicación

La arquitectura general de distribución de la aplicación está controlada por un gestor que obtiene un plan de ejecución de las tareas del experimento de simulación definido desde la interfaz (Figura 11). Este gestor prepara los datos de entrada para cada una de las tareas y decide a qué nodos (máquinas) de la red enviarlos. De esta forma, en función del número de nodos disponibles, el gestor reparte las tareas entre éstos, lanzando en cada uno de ellos unos procesos que se ejecutan en segundo plano, y que controlan de forma independiente el

orden de ejecución de las tareas. El plan general de ejecución se irá 'resolviendo' en tiempo real, dependiendo de la disponibilidad de cada nodo de la red.

Para interpretar esto, nos centraremos, a su vez, en una parte de la herramienta DAMOCIA-Sim, la correspondiente a la modelización de un invernadero en su cara externa. En la figura 12 se muestra un esquema de su funcionamiento.

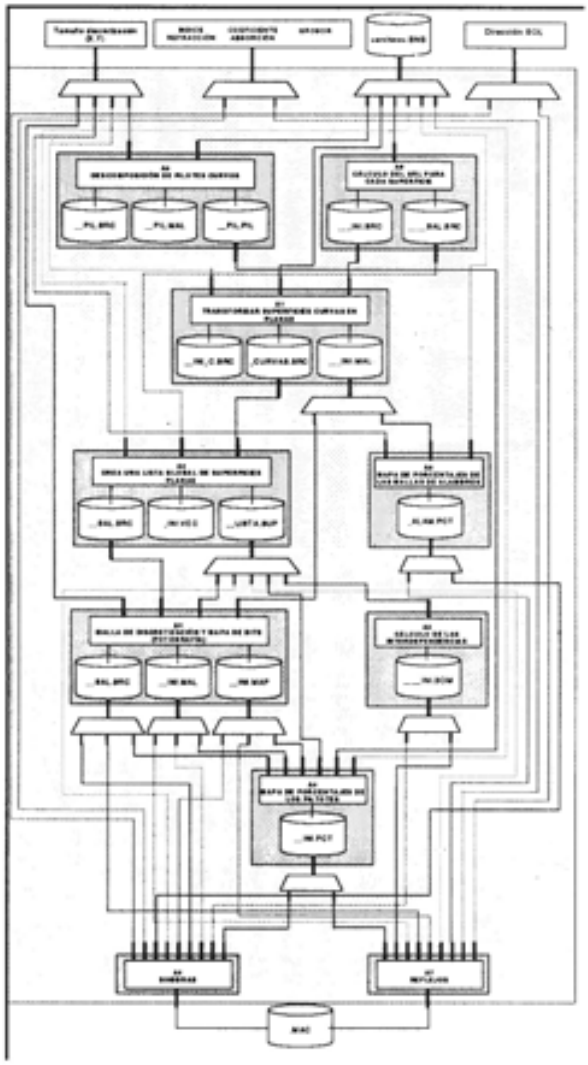

Figura 12: Modelización externa de un invernadero

Del esquema anterior podemos obtener el conjunto de tareas global (Tabla 1) que intervienen en la fase externa de la modelización. También podemos determinar el plan de ejecución inicial que aceptará el distribuidor del sistema (este se muestra en la Figura 13).

Tarea	*Descripción*
A0	*Cálculo Sistema de Referencia a cada superficie*
A1	*Transformar superficies curvas en planas*
A2	*Crear una lista global de superficies planas*
A3	*Cálculo malla de discretización y mapa de bits*
A4	*Cálculo del mapa de porcentajes de los pilotes*
A5	*Dependencias espaciales entre superficies*
A6	*Sombras entre superficies, en su fase externa*
A7	*Reflejos entre superficies, en su fase externa*
A8	*Mapa de porcentajes de las mallas de alambres*
A9	*Descomponer pilotes curvos en tramos rectos*

Tabla 1: Tareas de la fase externa de la simulación

El grado de distribución de estas tareas dependerá, sobre todo, de las características de los parámetros que se introduzcan desde la interfaz. Así, por ejemplo, si los tamaños de los elementos finitos de las superficies son de 0'5 x 0'5 metros, la distribución será menor, e incluso se podrán ejecutar todos los procesos en un mismo nodo antes que asignárselos a otro (distribuirlos). Las tareas A6 y A7 son las que presentan mayor capacidad de distribución al depender de factores tales como el periodo de tiempo de la simulación (por ejemplo, estudiar la radiación que capta un invernadero desde las 10:00 hasta las 19:00) y el nivel de análisis de reflejos entre las cubiertas del invernadero (ciclos de reflexión), tanto en sus caras externas como internas.

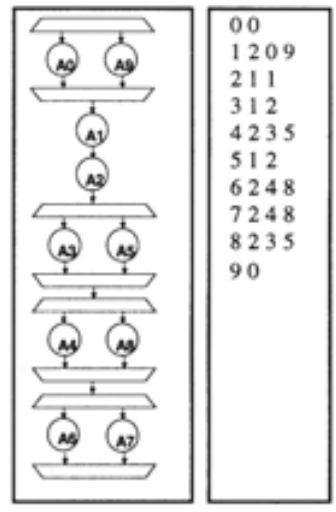

Figura 13: Plan (Ap)

El proyecto se realizó utilizando como plataforma de desarrollo un conjunto de PCs en red. La ejecución de los procesos de simulación puede llevarse a cabo sobre cualquier máquina con C estándar. La interfaz de usuario se ha de ejecutar en un entorno Windows (Windows 3.1, Windows NT o Windows95-98).

La metodología de desarrollo es mixta, la visión general sigue nuestra propuesta TE y los distintos subsistemas se pueden desarrollar siguiendo metodologías clásicas estructuradas u orientadas a objetos (según los elementos y destrezas disponibles en cada momento por parte de los implementadores). Los planes se modelan antes de su implementación utilizando CommonKADS.

Conclusiones y Futuros Trabajos

Del trabajo desarrollado podemos concluir básicamente dos cosas. En *primer lugar*, es posible realizar el análisis y diseño de herramientas de simulación que se ejecuten de forma distribuida mediante un análisis genérico de las tareas, al realizar la descomposición del objeto de estudio en sus elementos constituyentes. La composición en procesos concretos se realiza en tiempo de ejecución, combinando los correspondientes grafos de dependencia, siendo general e independiente del problema.

En *segundo lugar*, la arquitectura DACAS se puede utilizar, de forma general, como esquema de implementación en problemas de simulación (y/o diseño), en los que podamos diferenciar un esquema de diseño con las características señaladas en el punto anterior (un claro plan de tareas y un esquema de descomposición del objeto de la simulación).
Las principales *aportaciones del trabajo* son: *(a)* Un método para modelar la capacidad de distribución del procesamiento de los sistemas de simulación complejos. *(b)* La construcción de una plataforma distribuida flexible (el sistema de gestión de procesos es general), a partir de la descripción mediante CommonKADS de los niveles más altos de un problema (Iglesias-Fernández, 1997). *(c)* Un lenguaje gráfico para representar la capacidad de distribución de sistemas complejos en virtud de las etapas de tratamiento y de las partes constituyentes. *(d)* Una herramienta de simulación que estudia el comportamiento microclimático de invernaderos y que pone de manifiesto la efectividad del método propuesto.

Entre los *futuros trabajos* tenemos: *(a)* Mejorar el proceso de actualización del plan de ejecución de procesos incorporando reactividad ante la evolución del sistema global y preferencias sobre la ejecución de los procesos. *(b)* Incluir la evaluación del costo de los procesos, condicionando la ejecución distribuida a un rendimiento mínimo. Este análisis se realizará en la fase de construcción del plan de ejecución inicial a partir de las características propias de las tareas y elementos analizados, reduciendo el nivel inicial de distribución y tomando un límite de rendimiento mínimo. *(c)* Analizar la representación de procesos de simulación de tipo iterativo mediante el análisis de las distintas etapas de los bucles de simulación y sus consecuencias en la capacidad de distribución potencial de los procesos. *(d)* Ampliar el ámbito de aplicación de la técnica a la simulación de otros sistemas, como pueden ser los sistemas biológicos.

Referencias

Bienvenido J.F., Marin R. et al., "DACAS: A Distributed Architecture for Changeable and Adaptable Simulation". *EIS'97*. 1997.

Bienvenido J.F., Iribarne L.F. et al., "DAMOCIA-SIM, a generic tool for radiation simulation into mild winter region greenhouses". *First European Conference for Information Technology in Agriculture*. EFITA'97. Copenhagen. 1997.

Durfee E.H. and **Rosenshein J.S.**, "Distributed problem solving and multi-agent systems: Comparisons and examples". *In Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop*. 94-104, 1994. ftp://ftp.eecs.umich.edu/pub/people/durfee/}.

Erman, L., Hayes-Roth, F. Lesser, et al. "The HearsayII Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty". 12 (2) /213-253 *Comp. Surv*. 1980.

Hybinette M. and **Fujimoto R.**, "Dynamic Virtual Logical Processes". *Proc. of the 12th workshop on parallel and distributed simulation*, PADS98, 100-107, 1998.

García J.R., Tallón M. et al., "A Design Methodology Based on Multilevel Declarative Definitions and Distributed Software Architecture. Application to the Design of Greenhouses for Mild Winte Regions: DAMOCIA-Design". *2nd Annual International Conference on Industrial Engineering Applications and Practice*. Vol I. San Diego, USA. 1997.

Iglesias-Fernández C.A., "Definición de una metodología para el desarrollo de sistemas multiagente". *Tesis doctoral*, Universidad Politécnica de Madrid. 1997.

Iribarne L.F., Guirado R., et al., "Simulación Multimodelo mediante una Arquitectura Distribuida DACAS: DAMOCIA-Sim". *INFONOR97*, Chile, 1997.

Kirihara S., Aoyagi S. and **Onai R.**, "Adaptative Selection of Reactive/Deliverate Planning for the Dynamic Environment". *In Multi-Agent Rationality*, Boman and Van de Welde (eds), Springer-Verlag, 1997.

Lalis, I. and **Menhart P.**, "Object Oriented Toolset for Sequential and Distributed Simulation". *European Simulation Multiconference ESM95*, 1995.

Shi Z., Cao H., et al. "A Building Tool for Multiagent Systems: AOSDE". *Proceeding of the XV. IFIP World Computer Congress* ed. José Cuena, ITKnows Informations Technologies and Knowledge Systems, Viena, 1998.

So Y. and **Durfee E. H.**, "Designing Tree-Structured Organizations for Computational Agents". *Computational and Math. Organization Theory*, 2/3:219-246, Fall 1996.

Tseng M.M. and **Jiao J.**, "A Framework of Design for Mass Customization". *Annual International Conference on Industrial Engineering Applications and Practice*. Vol I. San Diego, USA, 1997.

Wooldridge M., "A Knowledge Theoretic Approach to Distributed Problem Solving". *13th European Conference on Artificial Intelligence*. Henri Prade eds. John Wiley and Sons pubs, 1998.

Yang, H. and **Zhang, C.,** "Definition and Application of a Comprehensive Framework for Distributed Artificial Intelligence". Springer, *LNAI 1087*, 1996.

Zhang C. and **Li Y.**, "An Algorithm for Plan Verification in Multiple Agent Systems". W. Wobcke, M. Pagnucco, C. Zhang eds. of *Agents and Multi-Agent Systems. Formalisms, Methodologies, and Applications*. LNAI, 1441/149-161, 1998.

***Luis-Fernando Iribarne Martínez**. Licenciado y Diplomado en Informática por la Universidad de Granada, España (1991). Profesor del Dpto de Lenguajes y Sistemas Informáticos de la Universidad de Granada desde 1991 hasta 1994 y profesor asociado por el Dpto de Lenguajes y Computación en la Escuela Politécnica Superior de la Universidad de Almería desde 1994 hasta la fecha. Investigador colaborador con el centro de investigación FIAPA (Fundación de Investigación Agraria de la Provincia de Almería, 1994-1996). Investigador del proyecto DAMOCIA, financiado por la UE y el Ministerio de Industria Español (1993-1996). Pertenece al grupo de investigación Plasticultura e informática aplicada a la agricultura y medioambiente (AGR-072). E-mail: liribarn@ualm.es*

***Isabel M. Flores Parra**. Licenciada y Diplomada en Informática por la Universidad de Granada, España (1990). Profesora del Dpto de Lenguajes y Sistemas Informáticos de la Universidad de Granada desde 1990 hasta 1994 y profesora asociada por el Dpto de Lenguajes y Computación en la Escuela Politécnica Superior de la Universidad de Almería desde 1994 hasta la fecha. Investigadora del proyecto DAMOCIA, financiado por la UE y el Ministerio de Industria Español (1993-1994). Pertenece al grupo de investigación Plasticultura e informática aplicada a la agricultura y medioambiente (AGR-072). E-mail: imflores@ualm.es*

***J. Fernando Bienvenido Bárcena**, Licenciado en Física por la Universidad de Sevilla, España (1982). Análista-programador y analista de sistemas en la empresa privada entre 1982 y 1984. Analista de sistemas en la Oficina Técnica Informática del CEMA, división de investigación de Michelin. Profesor del Dpto de Lenguajes y Sistemas Informáticos de la Universidad de Granada desde 1989 hasta 1994 y profesor por el Dpto de Lenguajes y Computación en la Escuela Politécnica Superior de la Universidad de Almería desde 1994 hasta la fecha. Investigador del proyecto DAMOCIA, financiado por la UE y el Ministerio de Industria Español (1993-1996). Pertenece al grupo de investigación Plasticultura e informática aplicada a la agricultura y medioambiente (AGR-072). E-mail: fbienve@ualm.es*