Entornos de programación para sistemas distribuidos tolerantes de fallos

ERNESTINA DE LOS ÁNGELES MARTEL JORDÁN

RESUMEN

Las nuevas tecnologías de multiprocesadores y redes de comunicación han permitido el auge de los sistemas distribuidos. Estos proporcionan transparencia, flexibilidad, fiabilidad, una gran potencia a bajo coste, así como escalabilidad. Sin embargo, el grado de complejidad de los sistemas distribuidos hace que los lenguajes de programación distribuida no sean suficientes para desarrollar aplicaciones distribuidas; se requiere un entorno que facilite la programación distribuida. Se analizan las características de varios entornos y se diseña uno para desarrollar aplicaciones distribuidas escritas con Drago, basado en el paradigma de grupos, una extensión del paradigma cliente-servidor.

ABSTRACT

Configuring toolkits for distributed systems capable of reacting to errors

The technological advances in the fields of multiprocessors and communicating networks have enormously improved the development of distributed systems. Distributed systems provide transparency, flexibility, reliability, performance and scalability. However, due to the complexity of distributed systems, distributed programming languages are not enough to develop distributed applications; it is required a toolkit to facilitate programming distributed aplications. In this paper, we show the current researchs in such subject and design a tool to program distributed aplications with Drago. Drago is based on groups, an extention to the client-server model of communication.

Kermarrec establece diferentes enfoques para programar sistemas distribuidos [KNP96]:

- Mediante los **servicios del sistema operativo** que utiliza los mecanismos de comunicación entre procesos distribuidos que a su vez proporciona el sistema operativo (por ejemplo, *sockets* de UNIX). El inconveniente de este enfoque es que las aplicaciones son muy dependientes del sistema operativo, con lo cual se reduce la portabilidad de una plataforma a otra. Una alternativa es la utilización de estándares como POSIX (*Portable Operating System Interface*), sin embargo, la programación es a bajo nivel y no se libera al programador de errores complejos.

- Mediante una **capa entre la aplicación y el sistema destino** que utiliza bibliotecas de servicios que permiten la programación de sistemas distribuidos. Por ejemplo, la biblioteca PVM (*Parallel Virtual Machine*) proporciona un conjunto de servicios que permiten considerar un conjunto de máquinas heterogéneas como una única máquina virtual. Aunque PVM parece solucionar el problema de portabilidad, el programador se encuentra con otros problemas relacionados con el paso de mensajes y manejo de buffers. Otras bibliotecas de servicios como DCE (*Distributed Computing Environment*) facilita la programación de aplicaciones distribuidas proporcionando seguridad, servicio de nombres, sincronización y llamada remota a procedimiento independientemente de la plataforma en la que se encuentre. La desventaja es que estos servicios se proporcionan mediante una API (*Application Programming Interface*) muy compleja.

- El enfoque que utiliza **lenguajes de programación distribuida** ha de contemplar la configuración del sistema distribuido en fragmentos y la programación de esos fragmentos. Muchos autores ([BWD+93], [KMS89]) coinciden en que para construir grandes sistemas es necesario dividir el sistema en componentes, los cuales se programan de forma independiente. La estructura del sistema es el resultado de una configuración de esos componentes y el comportamiento del mismo lo determina la programación de dichos componentes. Una de las ventajas de utilizar lenguajes de programación distribuida es que el propio lenguaje tiene la sintaxis y semántica necesaria para ejecutar la aplicación en un sistema distribuido, soportando comunicación/sincronización entre los fragmentos de la aplicación. Las características principales a considerar en un lenguaje de programación distribuida (lenguaje de configuración más lenguaje de programación de fragmentos) son:

  - **Determinación de fragmentos.** Una aplicación distribuida se compone de fragmentos que se ejecutan en diferentes nodos de la red. El lenguaje de configuración distribuida debe determinar cuáles son los fragmentos de la aplicación distribuida, es decir, las unidades de distribución de la aplicación.

  - **Ubicación física de los fragmentos.** Para cada fragmento o unidad de distribución se genera un ejecutable. El lenguaje de configuración debe incluir primitivas que permitan mapear esos ejecutables en los nodos de la red. Este mapeo de unidades de distribución en nodos no es trivial, ya que las máquinas pueden ser heterogéneas. Por tanto, se ha de establecer un mecanismo que asocie el ejecutable adecuado a cada una de las máquinas del sistema distribuido.

  - **Balanceo de la carga.** Cuando un fragmento comienza su ejecución lo hace en la máquina que disponga de recursos suficientes (CPU, memoria, etc.). Sin embargo, durante la ejecución puede variar la carga de la máquina y, como consecuencia, requerirse una migración del fragmento a otra máquina. El lenguaje de programación distribuida debe considerar cómo llevar a cabo tales migraciones sin que se al-

cance un estado inconsistente en el sistema.

- **Tolerancia a fallos y reconfiguración.** El lenguaje de configuración ha de permitir especificar los cambios que pueden tener lugar de forma dinámica (durante la ejecución) en la estructura del programa. La reconfiguración dinámica soporta modificaciones en tiempo de ejecución (sustitución de fragmentos, reconfiguraciones necesarias para tolerar fallos en el sistema), y está íntimamente relacionada con el mantenimiento de la consistencia e integridad de la aplicación.

- **Entornos heterogéneos.** En un sistema distribuido pueden existir máquinas de diferentes plataformas. El lenguaje de configuración debe proporcionar mecanismos que permitan la interacción en entornos heterogéneos tanto a nivel hardware como software.

- **Seguridad.** En un sistema distribuido se comparten recursos. El lenguaje de configuración puede incorporar mecanismos de autentificación.

Drago es un lenguaje de programación distribuida en el que las características de configuración no están integradas en el propio lenguaje. Actualmente la configuración en Drago se realiza mediante unos ficheros que deben estar en determinados directorios del sistema.

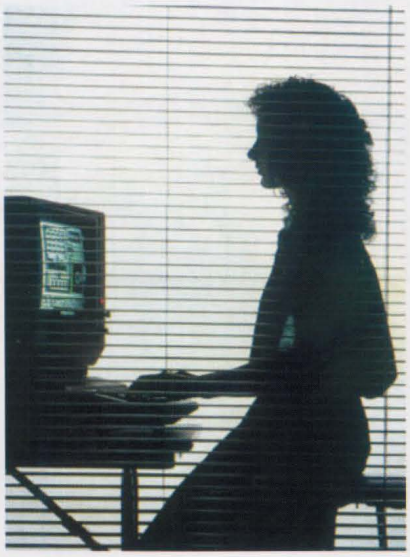
Por tanto, resulta imprescindible diseñar un entorno de configuración para Drago, mediante el cual se pueden configurar los grupos replicados y cooperativos. Los grupos de Drago son estáticos, aunque se puede pensar en una configuración dinámica de los grupos de la aplicación. Los sistemas distribuidos son propensos a particiones y, por tanto, el control de éstas debe tenerse en cuenta en el entorno de configuración.

ENTORNO DE TRABAJO

A continuación se presentan el entorno para el cual se propone la herramienta de configuración de aplicaciones distribuidas: el lenguaje de programación Drago y los protocolos subyacentes que proporcionan tolerancia a fallos.

El lenguaje de programación Drago

Drago [M94] es un lenguaje de programación distribuida que proporciona el soporte para desarrollar aplicaciones distribuidas estáticas y tolerantes a fallos. En el modelo de programación que utiliza Drago no se utiliza memoria compartida (por tanto, la única posibilidad de comunicación entre nodos de la red es mediante el paso de mensajes) y se considera que los procesadores del sistema son del tipo fallo silencioso (cuando un procesador falla, simplemente deja de funcionar sin avisar al resto).



El modelo de comunicación en Drago se basa en el paradigma cliente-servidor: un cliente solicita servicios a un servidor, el cual puede estar localizado en un nodo remoto. Concretamente en Drago se utiliza una extensión del modelo anterior: el paradigma de comunicación con grupos. Un grupo es un conjunto de unidades de distribución (denominadas agentes) que comparten la misma semántica de la aplicación. Los agentes pueden residir en distintos nodos de la red, aunque varios agentes pueden residir en el mismo nodo. La interacción entre los agentes se realiza mediante una extensión del mecanismo de cita del lenguaje Ada: la cita remota con grupo. El grupo servidor proporciona servicio al cliente sin que éste se entere del funcionamiento interno de dicho grupo, el cual no es visible al exterior. Drago soporta dos abstracciones de grupo:

- ***Grupos replicados,*** compuesto por réplicas idénticas que toleran fallos mediante redundancia modular: las réplicas se sitúan en distintos nodos del sistema distribuido y todas reciben los mismos mensajes y en el mismo orden, asegurando de esta forma el comportamiento de la aplicación como si de un autómata finito determinista se tratase. Todas las réplicas pasan

por los mismos estados y producen las mismas salidas.

- ***Grupos cooperativos,*** los cuales se utilizan para distribuir los datos y la carga del trabajo entre los miembros del grupo en el sistema distribuido. En este caso, los miembros del grupo cooperan para conseguir un objetivo común, sin necesidad de pasar por los mismos estados como en el caso anterior.

Los grupos en Drago son estáticos, es decir, una vez creado el grupo no se permite la incorporación de nuevos miembros. Cuando un miembro del grupo falla abandona para siempre el sistema.

Protocolos de comunicación tolerantes a fallos

Drago utiliza unos protocolos [G95] que proporcionan una co-municación con las siguientes características:

Atomicidad, todos los miembros vivos de un grupo observan los mismos eventos y en el mismo orden.

Causalidad, cuando un evento es causal a otro es observado en todo el sistema antes que aquel al que precede.

Fiabilidad, la notificación de eventos se realiza a todos los miembros del grupo o a ninguno.

Uniformidad, si a algún proceso se le entrega un mensaje, entonces a todos los procesos correctos (que no han fallado) se les entrega dicho mensaje.

Estos protocolos son de radiado y están íntimamente relacionados con la tolerancia a fallos en los sistemas distribuidos.

ENTORNOS DE CONFIGURACIÓN DISTRIBUIDOS EXISTENTES

En este apartado se presentan diferentes entornos de configuración distribuida de los cuales se han extraído algunas características para nuestra herramienta de configuración. Una de las características que diferencian a nuestra herramienta de configuración de las ya existentes es el entorno en el que trabajamos, el cual soporta comunicación con grupos y tolerancia a fallos.

Configuración de sistemas distribuidos en Ada 95

Ada 95 es un lenguaje de programación orientado a objetos que proporciona un modelo para programación distribuida. En dicho modelo ([BW95], [BW96]) la herramienta de configuración la ha de desarrollar el programador. Actualmente existen dos herramientas de configuración basadas en el modelo de programación distribuida de Ada 95: ADEPT y GLADE.

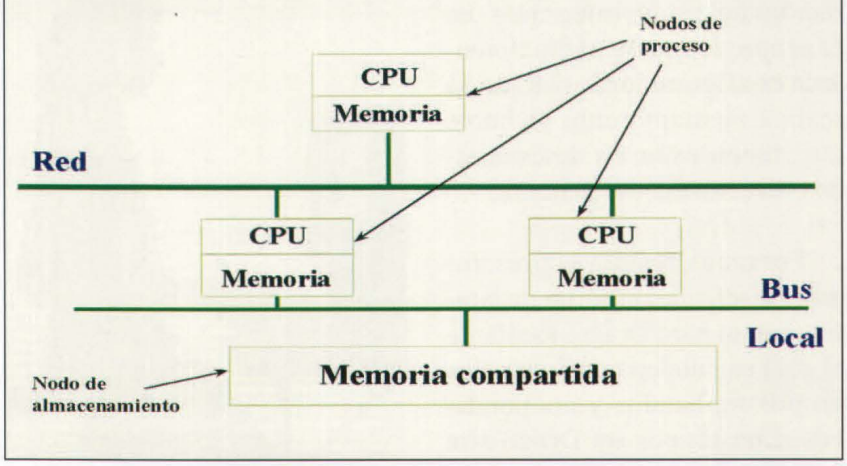
A continuación se describe el modelo de programación distribuida de Ada 95 y las herramientas de configuración ADEPT y GLADE.

Modelo de programación distribuida de Ada 95

Como acabamos de mencionar, este modelo abarca un mecanismo de distribución y una especificación de la semántica de comunicación. Sin embargo, el modelo no especifica ninguna herramienta de configuración, aunque considera que es necesaria. Por tanto, el diseñador de la aplicación distribuida debe proporcionar una implementación para el subsistema de comunicaciones y para la herramienta de configuración.

El **modelo de distribución** (ver figura 2) de Ada 95 considera una distribución a nivel software y otra a nivel hardware:

FIGURA 1 Nodos y particiones



- **Distribución software**

Un programa distribuido se compone de uno o más fragmentos, llamados *particiones*, (ver figuras 1 y 2) que se ejecutan de forma independiente, salvo cuando se necesita comunicación entre ellas. Las particiones se clasifican en:

- ***Activas***, son aquellas que tienen flujos de ejecución propios.

- ***Pasivas*** *o memoria compartida*, no tienen flujos de ejecución.

- **Distribución hardware**

Las unidades de distribución hardware son los nodos (ver figura 2) que se dividen en:

- ***Nodos de proceso***, con capacidad de ejecución y de almacenamiento. Las particiones activas se ubican en los nodos de proceso.

- ***Nodos de almacenamiento***, sólo con capacidad de almacén. Las particiones pasivas se ubican en los nodos de almacenamiento y son directamente accesibles por las particiones activas.

En este modelo la aplicación se puede ejecutar de forma local, localizando todas las particiones en una única máquina, o de forma distribuida, situando las particiones en diferentes máquinas del sistema distribuido.

Las particiones activas no pueden acceder de forma directa a los datos de otras particiones activas. Por tanto, la **comunicación** entre particiones activas se realiza mediante:

- Memoria compartida o nodos de almacenamiento. Los nodos de almacenamiento proporcionan datos y subprogramas a las particiones activas, es decir, su espacio de direcciones es directamente accesible por todas las particiones activas.

- Llamada remota a procedimiento (RPC, *Remote Procedure Call*) (ver figura 3). Una partición activa puede realizar una llamada a un procedimiento situado en un nodo remoto de la misma forma que hace una llamada a un procedimiento local. El compilador se encarga de generar la llamada remota mediante unos suplentes de envío y recepción. Estos suplentes se encargan de enviar la llamada con sus parámetros correspondientes por la red (suplente de envío o cliente) y una vez en el nodo remoto, tomar esos parámetros y realizar la llamada al procedimiento en cuestión (suplente de recepción o servidor). Una vez que se tienen los resultados, el suplente servidor los envía por la red hasta llegar al cliente que hizo la llamada.

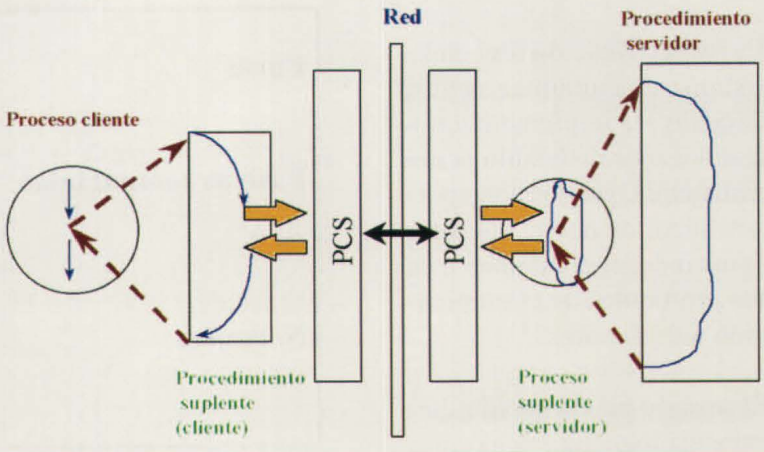
El modelo de programación distribuida de Ada 95 no especifica qué subsistema de comunicaciones debe utilizarse. Sin

FIGURA 2 **Modelo de distribución de Ada 95**

Hardware	Software
Nodo	Partición
- Proceso - Almacenamiento	- Activa - Pasiva

Una o más Particiones por Nodo

FIGURA 3 **Subsistema de comunicación**



embargo, propone una interfaz o especificación a partir de la cual el diseñador de la aplicación distribuida puede implementar el subsistema de comunicaciones adaptado a sus necesidades. Esta especificación es lo que se llama PCS (*Partition Communication Subsystem*). Los suplentes proporcionados por el compilador para realizar la llamada remota a procedimiento realizarán una llamada al PCS (ver figura 3) con el fin de proporcionar la semántica de comunicación adecuada (en nuestro entorno de trabajo, el PCS proporciona comunicación con grupos de forma atómica, causal, fiable y uniforme para garantizar la tolerancia a fallos).

En definitiva, según el modelo de programación distribuida de Ada 95, un sistema distribuido es una interconexión de uno o más nodos de proceso y cero o más nodos de almacenamiento (ver figura 1). Los nodos de proceso se comunican entre sí mediante memoria compartida (nodos de almacenamiento) o mediante llamada remota a procedimiento. Las ventajas de este modelo de programación son:

- La existencia de nodos de proceso y de almacenamiento permite que el modelo funcione tanto en multiprocesadores como en sistemas con memoria distribuida.
- La especificación del subsistema de comunicaciones, dejando su implementación al diseñador de la aplicación distribuida, proporciona portabilidad, es decir, que funcione independientemente de los protocolos de comunicación subyacentes.
- El usuario puede decidir ejecutar el programa en una única máquina o en varias máquinas sin tener que realizar ninguna modificación sobre el código.

En Ada 95 una partición es un conjunto de bibliotecas (procedimientos o paquetes). Para distinguir entre particiones activas y pasivas se establece una categorización de estas particiones mediante *pragmas* (sentencias que indican al compilador la categoría de la partición). De esta forma, el programador designa el papel que juega la partición en el programa distribuido. En la figura 4, se presentan las diferentes categorías de particiones en Ada 95 y su significado.

Herramientas de configuración: GLADE y ADEPT

GLADE [KNP96] es una herramienta de configuración de libre distribución que utiliza el modelo de distribución de Ada 95 y está compuesta de:

- **GARLIC:** subsistema de comunicaciones.
- **GNATDIST:** herramienta que permite dividir la aplicación distribuida en particiones y ubicarlas en las máquinas físicas partiendo de un fichero de configuración (ver figura 5). Algunas de las características de esta herramienta son:
  - Por defecto, la herramienta pregunta el nombre de las máquinas en las que se desea ubicar las particiones. En el fichero de configuración se pueden especificar las máquinas en las que se desea situar las particiones o bien que la localización de las particiones sea dinámica, es decir, varíe durante la ejecución del programa. La localización dinámica de particiones se puede utilizar para reconfigurar dinámicamente la aplicación, bien porque se ha producido un error o porque ha cambiado la carga de la máquina en la que se encuentra la partición.
  - En el fichero de configuración se especifican los directorios en los que se encuentran los ejecutables en cada máquina. Esta característica resulta de especial interés cuando se trabaja en entornos heterogéneos.

FIGURA 4 **Categorización de particiones**

Puras	Sin estado, se replican en todas las particiones que las referencien. Contienen tipos. (Pasivas)
Pasivas compartidas	Nodos de almacenamiento. (Pasivas)
RCI	Contienen procedimientos que se llaman de forma remota. (Activas)
Normales	Sin categorización. Si se incluyen en más de una partición su estado evoluciona de forma diferente

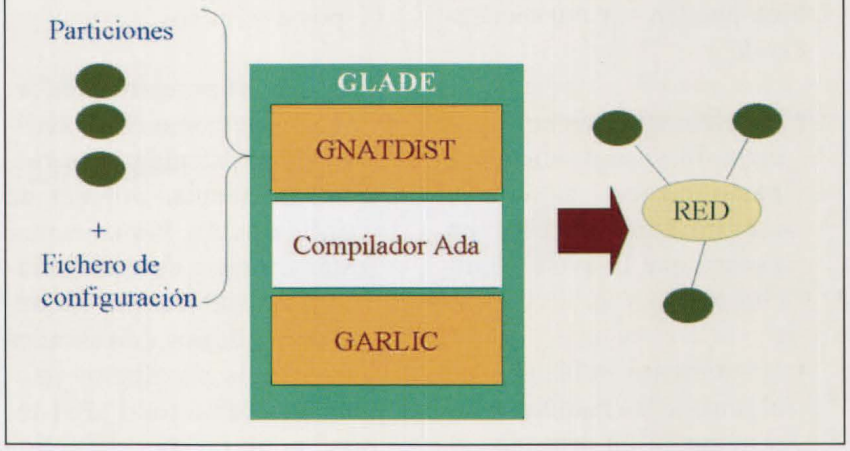
- Modo traza que ayuda a depurar la aplicación distribuida.

ADEPT (*Ada 95 Distributed Execution and Partitioning Toolset*) [GKN+97] es una herramienta de configuración gráfica, a diferencia de la anterior. Sin embargo, el que la herramienta sea gráfica disminuye la eficiencia y portabilidad ya que presenta una fuerte dependencia con respecto al entorno gráfico (MOTIF), además de código adicional no relacionado con la funcionalidad de la herramienta.

Esta herramienta está formada por:

- Una herramienta gráfica, denominada PCT (*Partition Configuration Tool*), mediante la cual se configura la aplicación basada en el modelo de programación distribuida de Ada 95. La configuración se realiza en los siguientes pasos:
  - Creación de particiones.
  - Asignación de bibliotecas (procedimientos y paquetes) a las diferentes particiones.
  - Generación de los ejecutables de cada partición.
  - Creación de nodos de proceso y almacenamiento.
  - Asignación de las particiones a los nodos de proceso y almacenamiento.
  - Carga y ejecución del programa distribuido.
- Un subsistema de comunicaciones, implementación del PCS que proporciona Ada 95.

FIGURA 5 Herramienta GLADE



HERRAMIENTA DE CONFIGURACIÓN PARA APLICACIONES DISTRIBUIDAS TOLERANTES A FALLOS

En este apartado se presentan las características deseables para nuestro entorno de configuración. Antes de comentar tales características mostramos los módulos de la aplicación distribuida con los cuales interactúa la herramienta de configuración (ver figura 6):

- El *sistema* de la figura 6 representa el conjunto de máquinas, la red que las interconecta, los recursos existentes en cada una (memoria, CPU, disco, etc.), así como el software y procesos ejecutándose en cada una de esas máquinas.
- La *herramienta de configuración* configura el sistema distribuido de forma estática o dinámica. La configuración estática se lleva a cabo inicialmente, mientras que la dinámica se realiza durante la ejecución.
- El *monitorizador* observa el sistema y avisa a la herra-

FIGURA 6 Herramienta de configuración

mienta de configuración de los cambios que puedan ocurrir en el mismo. Los cambios pueden ser consecuencia de:

- Un error en el sistema, por ejemplo, si se produce una partición en la red, el monitorizador recibe un evento que le avisa de dicho error.

- Un aumento en la carga de alguna de las máquinas del sistema: si un proceso comenzó a ejecutarse en una máquina porque ésta contaba inicialmente con recursos suficientes, puede que en un instante determinado la carga en esa máquina sea excesiva y disminuya considerablemente el rendimiento de ese proceso. Tal situación debe comunicarse al monitorizador para que avise de ello a la herramienta de configuración.

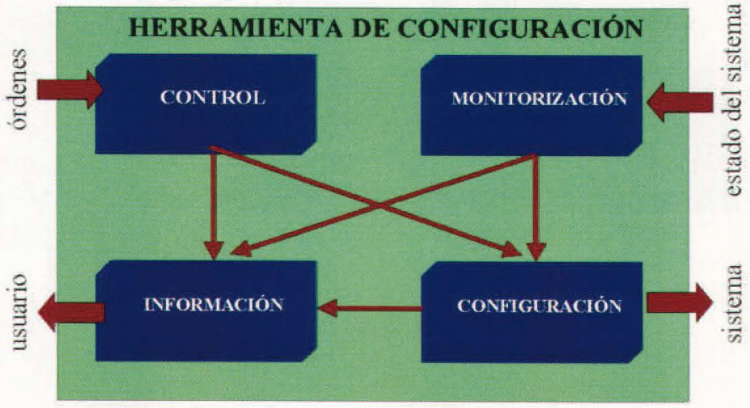
- Modificación del número de miembros que forman el grupo. Aunque los grupos en nuestro entorno son estáticos, es decir, durante la vida del grupo no se pueden añadir miembros al grupo, se está trabajando en la incorporación de grupos dinámicos.

- Los *usuarios* reciben información de cómo está configurado el sistema y, a su vez, pueden ejecutar órdenes de configuración. Para ejecutar estas órdenes de configuración se establece un esquema de privilegios. Los usuarios con ciertos privilegios pueden acceder a todo el sistema, mientras que aquellos con privilegios limitados sólo pueden hacerlo a los procesos y máquinas que ellos hayan configurado. En cada instante, puede existir más de un usuario observando la configuración del sistema. Con el fin de evitar inconsistencias en el sistema, cuando existen varios usuarios, todos pueden visualizar la configuración; sin embargo, para la ejecución de órdenes de configuración se debe garantizar exclusión mutua.

Con el fin de facilitar el diseño de la herramienta de configuración se ha dividido ésta en los siguientes módulos (ver figura 7):

FIGURA 7

Módulos de la herramienta de configuración



- *Módulo de configuración*, encargado de realizar todas las funciones que cambien la configuración del sistema. La configuración puede ser estática o dinámica. La primera se establece en el arranque de la aplicación mientras que la segunda, también llamada reconfiguración, tiene lugar durante su ejecución. Los cambios en la configuración pueden ser debidos a órdenes del usuario (módulo de control) o bien a un cambio de estado del sistema (módulo de monitorización).

- *Módulo de monitorización*, éste, a partir del estado del sistema, obtenido a partir de eventos o bien mediante la observación del mismo, proporciona datos a:

  - El módulo de configuración, el cual actúa en consecuencia.

  - El módulo de información que a su vez los proporciona a los usuarios.

- *Módulo de información*, responsable de proporcionar en cualquier instante la configuración actual, a partir de los datos que recibe de:

  - Un cambio en la configuración del sistema por el módulo de configuración.

  - Una monitorización del estado del sistema.

  - Procesamiento de las órdenes de usuario por el módulo de control.

- *Módulo de control*, recibe órdenes de usuario y a partir de las mismas genera ór-

denes para el módulo de configuración y datos para el módulo de información.

Los módulos citados son los responsables de las principales funciones de la herramienta de configuración. Esas funciones se detallan a continuación, indicándose, en cada caso el módulo o módulos implicados:

- *Ubicación física.* De la misma forma que en el modelo de programación distribuida de Ada 95 necesitábamos localizar las particiones en los nodos, aquí debemos preocuparnos de la ubicación de los procesos en las máquinas y de ciertos recursos que éstos necesitan (por ejemplo, el software que proporciona tolerancia a fallos en la comunicación con grupos). En este apartado puede ser interesante considerar los detalles siguientes:

  - El usuario puede especificar el tipo de grupo: cooperativo o replicado.

  - Para el caso de grupos replicados se especifica el número mínimo de réplicas necesarias para que el grupo pueda funcionar.

  - Opcionalmente la herramienta de configuración puede realizar una ubicación transparente al usuario: el usuario proporciona los ejecutables necesarios junto con ciertas especificaciones (por ejemplo, arquitectura necesaria, recursos necesarios, etc.) y la herramienta de configuración los sitúa en las máquinas adecuadas; por ejemplo, aquellas que disponen de los recursos necesarios o, en determinadas aplicaciones, en máquinas muy próximas entre sí.

El módulo de configuración de la herramienta es el responsable de esta función, así como de los detalles mencionados.

- *Localización y monitorización de recursos.* Una de las funciones del módulo de monitorización es supervisar la carga de los recursos de las distintas máquinas del sistema. Si un proceso está ejecutándose en una máquina y en un momento dado ésta no tiene recursos suficientes para continuar ejecutándolo, la herramienta de configuración debe proceder a la migración del proceso a otra máquina. La solución no es trivial ya que la migración implica una transferencia de estado. En herramientas como *Piranha* [M97] se arranca un proceso nuevo en otra máquina, se le realiza la transferencia de estado y a continuación se termina con el proceso anterior. En otros casos como en Conic [KM90] no se realiza la transferencia de estado hasta que se alcance un estado de reposo en todos los procesos implicados. Las migraciones, además, no deben ser frecuentes en un mismo proceso.

- *Control de acceso a usuarios.* El módulo de control cuando recibe órdenes de usuario comprueba si tienen privilegios suficientes. Se puede incluir una autentificación de los usuarios (mediante clave de acceso a la herramienta de configuración u otra técnica similar) y, basándose en la misma, comprobar que los usuarios pertenecen a determinados grupos con ciertos privilegios previamente definidos.

- *Presentación de la configuración actual del sistema.* El módulo de información de la herramienta de configuración es el responsable de esta función. La información presentada al usuario se obtiene a partir de los datos proporcionados por el resto de los módulos de la herramienta. En un momento dado pueden existir varias herramientas de configuración ejecutándose en diferentes máquinas del sistema y ambas presentándole la configuración al usuario.

- *Control de versiones software.* El módulo de configuración controla que todos los miembros de un grupo utilicen las mismas versiones software (bibliotecas, por ejemplo) .

- *Interoperabilidad.* Este término se refiere a la interacción entre plataformas heterogéneas, tanto a nivel software (comunicación entre sistemas operativos diferentes) como hardware (interacción entre arquitecturas distintas). El módulo de configuración de la herramienta controla la interoperabilidad.

- *Especificación de la entrada/salida estándar del grupo.* Un grupo, en un momento dado, puede desear generar una salida a pantalla o bien a un fichero. El módulo de configuración es el responsable de determinar la entrada/salida estándar de un grupo. Esta solución resulta sencilla para los grupos replicados, ya que éstos siem-

pre pasan por los mismos estados, reciben las mismas entradas y generan salidas idénticas. Los grupos cooperativos, en cambio, pueden necesitar diferentes entradas o generar salidas diferentes. La salida es especialmente problemática ya que usuarios diferentes pueden realizar composiciones distintas de las salidas de los miembros del grupo cooperativo y generar resultados inconsistentes. Para solucionar este problema se diseña un componedor. Un componedor es un proceso que unifica las salidas de todos los miembros del grupo cooperativo. El módulo de configuración define como salida estándar del grupo cooperativo la salida estándar del componedor. La entrada estándar de un grupo cooperativo puede ser única, o bien, especificar una entrada estándar diferente para cada uno de los miembros del grupo cooperativo.

- *Configuración de parámetros de red.* Algunos parámetros de comunicación se configuran inicialmente. Sin embargo, la monitorización del comportamiento del sistema puede dar lugar a una reconfiguración de esos parámetros. La herramienta de configuración permite que el usuario modifique esos parámetros, o bien que el cambio lo realice el módulo de configuración.

- *Depuración y registro de la evolución del grupo.* El módulo de monitorización se encarga de realizar esta depuración distribuida. Una implementación consiste en que uno de los miembros del grupo se dedique exclusivamente a observar el comportamiento del grupo y genere información, por ejemplo, al nivel de mensajes que permita seguir la ejecución de la aplicación distribuida.

- *Servicios de entorno.* Algunos de estos servicios que inicia el módulo de configuración son:

  - Utilidad que visualiza los servicios proporcionados por cada grupo. Si deseamos hacer una teleconferencia hemos de conocer a priori el grupo que proporciona este servicio para realizar la petición a dicho grupo. La utilidad de consulta de servicios asociados a grupos puede, a su vez, registrar las versiones software de cada grupo.

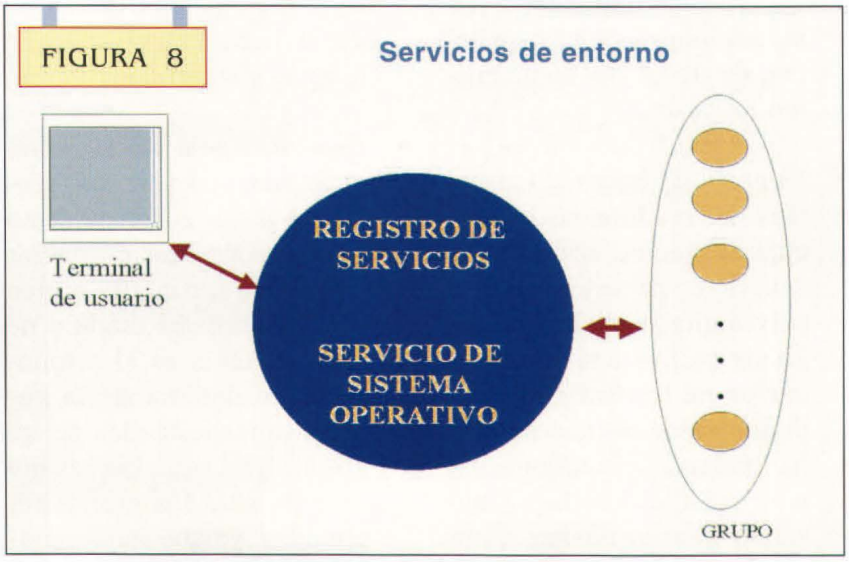
  - Servicio de sistema operativo. Este servicio debe existir en cada máquina que interaccione con grupos. Este servicio (ver figura 8) es responsable de gestionar comandos y señales sobre grupos. Por ejemplo, cuando se manda una señal CTRL-C a un grupo, este servicio convierte esa señal a mensajes que entienda el grupo y que sean equivalentes a dicha señal. Los comandos de usuario que actúan sobre grupos se definen en una biblioteca de programas (API) que puede contener comandos para conocer el estado de un grupo, para redireccionar la entrada/salida estándar del grupo, etc.

- *Recepción de notificaciones de fallos.* El módulo de monitorización, el de control y el de configuración intervienen en esta función para reaccionar a los fallos del sistema. Algunos detalles a considerar en esta función son:

  - Determinar la política que se ha de adoptar cuando una máquina deja de funcionar y en la misma se encontraban miembros de un grupo.

  - Especificar cómo actuar cuando se mata a un proceso el cual resultaba imprescindible para completar el número mínimo de réplicas del grupo replicado.

FIGURA 8



CONCLUSIONES

La programación distribuida mediante lenguajes de programación distribuida no es sencilla. Esa es la razón por la que en muchos entornos distribuidos actuales se han dividido los lenguajes de programación distribuida: en lenguajes de configuración que definan la estructura de la aplicación, y en lenguajes de programación que definan el comportamiento de la misma.

En nuestro grupo de trabajo se planteó la necesidad de realizar una herramienta de configuración para nuestro entorno: programación distribuida tolerante a fallos mediante el paradigma de comunicación con grupos. Esta herramienta no sólo incluye un lenguaje de configuración sino que es capaz de reaccionar a eventos que se produzcan en nuestro sistema. La implementación de la herramienta puede verse respaldada por una base de datos relacional en la que se vayan registrando las diferentes configuraciones por las que pasa en el sistema. Esto implicaría decidir entre desarrollar la gestión de la base de datos o utilizar sistemas de gestión de bases de datos ya existentes.

Otra posibilidad a considerar es la de ejecutar la herramienta desde una página Web. Actualmente existen lenguajes y mecanismos que lo permiten. Sin embargo, la elección de cuál utilizar no es trivial.

La posibilidad de trabajar en entornos heterogéneos a nivel software y hardware es otra de las características en estudio.

GLOSARIO

Biblioteca: Unidad software cuya funcionalidad puede ser utilizada por otros programas.

Cita de Ada: El paralelismo en el lenguaje Ada se consigue mediante las tareas. La cita es el mecanismo de sincronización entre tareas.

Exclusión mutua: Situación en la que se permite un único acceso a un recurso. Por ejemplo, en caso de escritura a disco, sólo uno de los procesos puede escribir.

Grupo: Conjunto de procesos que comparten la misma semántica de la aplicación. Estos procesos pueden situarse en la misma o diferentes máquinas.

Paquete: Unidad de programación del lenguaje Ada que consta de una especificación y un cuerpo.

Proceso: Programa en ejecución.

Radiado: Envío de mensajes a destinos. El radiado puede ser a todos los destinos (broadcast) o a un número limitado de los mismos (multicast).

Redundancia modular con votación: Mecanismo de detección de fallos en el que varias copias de un proceso operan concurrentemente. Las salidas se comparan y aquellas que no coincidan se consideren erróneas.

Subprograma: Procedimiento o función que a partir de unas entradas genera unas salidas.

BIBLIOGRAFÍA

[KM85] J. Kramer, J. Magee: "*Dynamic Configuration for Distributed Systems*", *IEEE Transactions on Software Engineering.* Núm. 4. Abril 1985. Págs. 424-436.

[SKM86] M. Sloman, J. Kramer, J. Magee: "*Flexible communication structure for distributed embedded systems*". *IEEE Proceedings.* Vol. 133, Núm. 4. Julio 1986. Págs. 201-211.

[D89] N. Dulay: "*The Conic Language: an introduction*". Dept. of Computing Imperial College. London. Septiembre 1989. Transparencias 1-17.

[KMF89] J. Kramer, J. Magee, A. Finkelstein: "*A Constructive Approach to the Design of Distributed Systems*". *Proceedings of the 3rd Workshop on Large Grain Paralelism.* Pittsburgh. SEI/CMU. Octubre 1989. Págs. 1-15.

[KMS89] J. Kramer, J. Magee, M. Sloman: "*Constructing Distributed Systems in Conic*". *IEEE Transactions on Software Engineering.* Vol 15, Nº 6. Junio 1989. Págs. 663-675.

[KM90] J. Kramer, J. Magee: "*The Evolving Philosophers Problem; Dynamic Change Management*". *IEEE TSE.* Vol.

BIBLIOGRAFÍA

16. Núm 11. Noviembre 1990. Págs. 1-24.

[BWD+93] M. R. Barbacci, C. B. Weinstock, D. L. Doubleday, M. J. Gardner, R. W. Lichota: "*Durra: a structure description language for developing distributed applications*". *Software Engineering Journal.* Marzo 1993. Vol. 8 (2). Págs. 83-94.

[C93] F. Cristian: "*Automatic reconfiguration in the presence of failures*". *Software Engineering Journal.* Marzo 1993. Págs. 53-60.

[HWP+93] C. Hofmeister, E. White, J. Purtilo: "*Surgeon: a packager for dinamically reconfigurable distributed applications*". *Software Engineering Journal.* Marzo 1993. Págs. 95-101.

[M94] F. J. Miranda González: "*Drago: un lenguaje para programar aplicaciones distribuidas y tolerantes a fallos y cooperativas*". Tesis doctoral, 1994.

[BW95] A. Burns, A. Wellings: "*Concurrency in Ada*". Cambridge University Press. 1995.

[CRV95] F.J.N. Cosquer, L. Rodrigues, P. Veríssimo: "*Using Tailored Failure Suspectors to Support Distributed Cooperative Applications*". *Proceedings of the 7th International Conference on Parallel and Distributed Computing and Systems.* Washington. Oct. 1995. Págs. 1-10.

[CV95] F.J.N. Cosquer, P. Veríssimo: INESC-IST, Lisboa. "*The Impact of Group Communication Paradigms on Groupware Support*". *Proceedings of the 5th IEEE CS Workshop on Future Trends of Distributed Computing Systems.* Agosto 1995. Págs. 1-13.

[G95] F. J. Guerra Santana: "*Protocolos eficientes de consenso para sistemas distribuidos*". Tesis doctoral, 1995.

[CAV96] F.J.N. Cosquer, P. Antunes, P. Veríssimo: "*Enhancing Dependability of Cooperative Applications in Partitionable Enviroments*". IST-INESC. Lisboa 1996. Págs. 1-23.

[MAA+96] F. J. Miranda, A. Álvarez, S. Arévalo, F. J. Guerra: "*Drago: An Ada Extension to Program Fault-Tolerant Distributed Applications*". 1996 Ada-Europe International Conference on Reliable Software Technologies. Suiza. Junio 1996. Págs. 235-246.

[KNP96] Y. Kermarrec, L. Nana, L. Pautet: "*GNATDIST: a configuration language for distributed Ada95*", *Proceedings Tri-Ada'96.* Pensylvania. ACM. Diciembre 1997. Págs. 63-72.

[BW96] A. Burns, A. Welligs: "*Programming replicated systems in Ada 95*". *Computer Journal.* Vol. 39, Núm 5. 1996. Págs. 361-373.

[GKN+97] Gargaro, Kermarrec, Nana, Pautet, Smith, Tardieu, Theriault, Volz & Waldrop: "*ADEPT (Ada 95 Distributed Execution and Partitioning Toolset*". Abril. 1997.

[M97] S. Maffeis: "Piranha: A CORBA Tool for High Availability", *IEEE Computer*, Abril 1997. Págs. 59-66.

BIOGRAFÍA

Ernestina de los Ángeles Martel Jordán

Es Diplomada y Licenciada en Informática por la Universidad de Las Palmas de Gran Canaria. Desde Febrero de 1994 trabaja como profesora ayudante en el Departamento de Electrónica, Telemática y Automática de la ULPGC. Actualmente se encuentra realizando su tesis doctoral en el Grupo de Sistemas Operativos Distribuidos de la ULPGC.

Dirección:
Dpto. Electrónica, Telemática y Automática
Universidad de Las Palmas de Gran Canaria
Campus Universitario de Tafira
35017- Las Palmas
Teléfono: 928 45 28 76 - Fax: 928 45 18 43
e-mail: emartel @cic.teleco.ulpgc.es

Este trabajo ha sido patrocinado por:

MAYA, S.A.