3. Caracterización de la carga

3.1. Introducción

Se denomina *carga* a todas las demandas que realizan los usuarios de un sistema (programas, datos, órdenes, etc.) durante un intervalo de tiempo.

En muchas ocasiones se puede estar interesado en obtener determinados índices de prestaciones de un sistema con el fin de ver el grado de explotación a que está sometido, prever su compra, etc. Para ello es preciso tomar una serie de medidas sobre este sistema. Las herramientas necesarias para ello han sido estudiadas en el capítulo anterior. Ahora bien, para realizar estas mediciones es necesario someter al sistema a una carga determinada.

El resultado de las medidas de prestaciones de un sistema es función y, por tanto, es inseparable de la carga bajo la cual fue determinado. Por ello, cuando se da un índice de prestaciones de un sistema, debe especificarse la carga bajo la que fue obtenido dicho índice. Así, la comparación entre sistemas o configuraciones, sólo tiene sentido cuando las cargas procesadas para su comparación son las mismas.

En caso contrario, las diferencias en los índices de prestaciones reflejarán, además de las diferencias en las prestaciones reales, las diferencias en las cargas, siendo imposible separar ambas diferencias. Un ejemplo son las prestaciones que los fabricantes dan de sus equipos, que normalmente están tomadas bajo cargas que favorecen las características de sus productos y penalizan fuertemente los de sus competidores.

Otro aspecto a tener en cuenta, y que es fuente de problemas, es la variación de la carga en el tiempo. Así, la carga que procesa cualquier sistema está continuamente variando en el tiempo, lo que dificulta su modelización (figura 3.1).

Por último, otro aspecto importante es que la propia carga está interaccionando con el sistema que la procesa, estableciéndose distintas realimentaciones. Por consiguiente, se suelen tener dos realimentaciones principales en un sistema: la interna y la externa (figura 3.2).

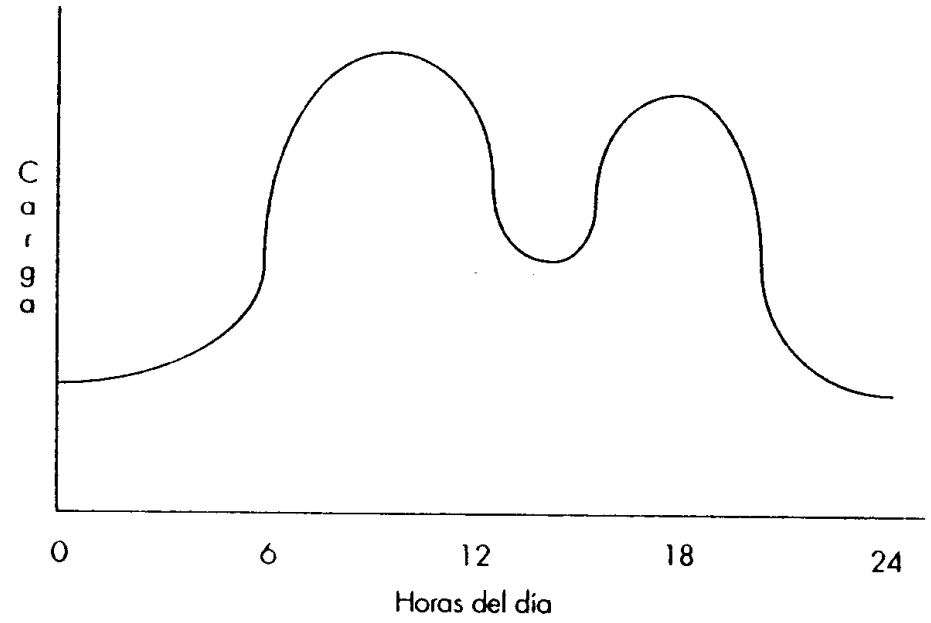

FIGURA 3.1.

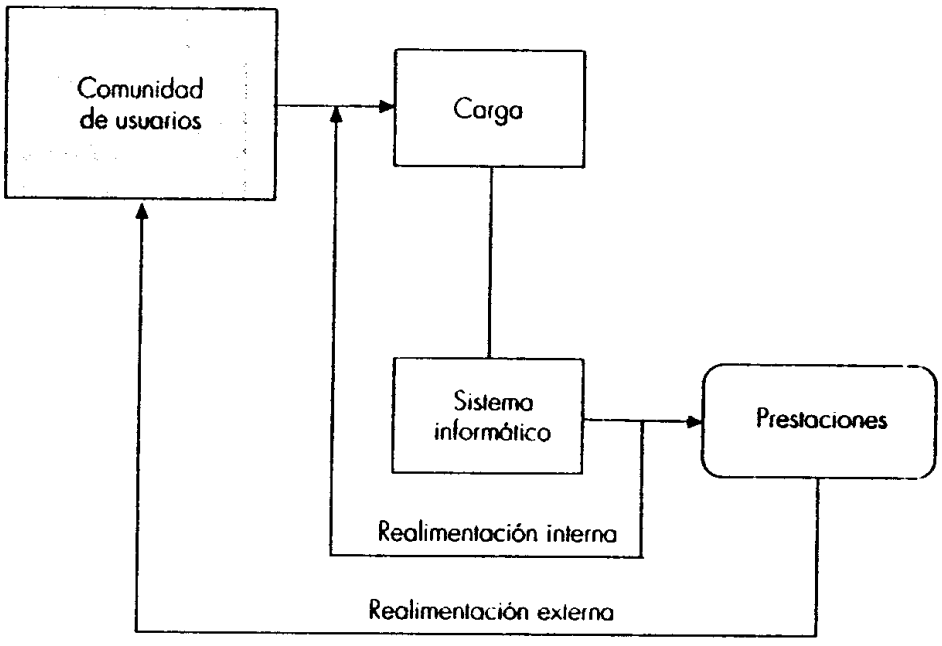

FIGURA 3.2.

El bucle interior de realimentación pone de manifiesto las variaciones que las modificaciones de parámetros del sistema operativo (del asignador dinámico de memoria, de situación de archivos en disco, etc.) inducen en la carga (y más exactamente en la coincidencia de trabajos en el sistema). Por ejemplo, el tiempo de servicio de los discos es función de la carga del sistema, pues depende del tiempo que necesita la unidad de control para resolver los conflictos; de esta forma, el consumo de recursos del sistema informático (tiempo de servicio de los discos, en este caso) puede ser función de la caga.

El bucle exterior de realimentación es el que intenta poner de manifiesto la incidencia del comportamiento del sistema (tiempos de respuesta, principalmente) sobre los hábitos y comportamientos de los usuarios, los cuales, a fin de cuentas, son los que provocan la carga.

En general, se ve, pues, que las relaciones entre el sistema, la carga y las prestaciones pueden ser muy complejas. Las principales causas de esta complejidad son:

— La carga está continuamente fluctuando.
— La carga interacciona con su entorno (el propio sistema, los usuarios, etc.).

Estas consideraciones justifican la necesidad de llegar a definir cuantitativamente la carga de un sistema, es decir, llegar a establecer un modelo de la carga. En general, esta caracterización acostumbra a efectuarse en función de aquellos parámetros que pueden afectar al comportamiento del sistema. Es decir, una carga está correctamente caracterizada si, y sólo si, su resultado es un conjunto de parámetros cuantitativos seleccionados de acuerdo con los objetivos de la operación de caracterización.

La definición de estos objetivos, es decir, del uso del modelo de la carga que se va a caracterizar, es el primer paso de cualquier estudio de evaluación del comportamiento. La especificación de estos objetivos permite determinar los parámetros que se deben usar y su representación.

Una vez establecidos los objetivos, deben determinarse las herramientas que manipularán (modelos analíticos, simulación, el sistema real, etc.) el modelo de la carga. Este hecho permitirá establecer los parámetros necesarios y su forma de representación, que, como se vio en el apartado 1.4., pueden referirse a cada componente de la carga o a su conjunto. Algunos de los parámetros que se utilizan en la caracterización de la carga son los siguientes:

— tiempo total de CPU, tiempo medio de CPU entre dos E/S y sus distribuciones
— número de operaciones de E/S
— características (o tiempo) de operaciones de E/S y sus distribuciones
— espacio en memoria
— líneas impresas
— ficheros en disco
— cintas leídas
— mezcla de instrucciones

— frecuencia de llegada o tiempo entre llegadas consecutivas y sus distribuciones
— número de usuarios
— tiempo de reflexión y su distribución
— etcétera.

Independientemente de la forma como se represente, *la carga de prueba o de test* (el modelo de la carga) se define como la carga que debe procesar un sistema mientras se está realizando un experimento de medición determinado.

Uno de los principales problemas que puede tener la carga de prueba es que no sea fácilmente reproducible. Por tanto, la carga real es difícilmente reproducible aunque se ejecuten los mismos programas en el mismo sistema.

Por consiguiente, la repetibilidad o reproductibilidad será una característica importante de las cargas de prueba, sobre todo si se están comparando distintos sistemas. Otras veces, la carga de prueba (en estudios predictivos) sólo se conoce hipotéticamente y de forma teórica.

La mayoría de las veces la carga de prueba será un modelo de la carga por los siguientes motivos:

— Satisfacer la exigencia de reproductibilidad de los experimentos, que haga significativas las comparaciones de los mismos índices de comportamiento; estas comparaciones son esenciales en muchos tipos de estudios, como, por ejemplo, cuando se evalúa la eficacia de distintas acciones de sintonización.
— Reducir sustancialmente la duración de cada sesión de medición con respecto a la que se requeriría para ejecutar la carga real total.
— Obtener una representación de la carga consistente con su uso (compatibilidad), ya que muchas veces se están utilizando simuladores o modelos analíticos, donde, evidentemente, no se puede utilizar la carga real.
— Evitar problemas de privacidad y seguridad, que a veces limitan el uso de programas y datos reales en los estudios de evaluación del comportamiento.

Por tanto, cualidades como:

— reproductibilidad
— compacidad
— compatibilidad

son características deseables en la carga de prueba. Estas características en ocasiones sólo se pueden obtener con modelos de carga que se utilizarán como carga de prueba para reemplazar la carga real.

Otras características que debe presentar el modelo de carga son:

— Representatividad, que indica los aspectos de la carga real retenidos en el modelo.

— Flexibilidad, que indica si es posible y poco costoso variar el modelo para ajustarlo a las variaciones que se produzcan en la carga real.
— Independencia del sistema, importante en los problemas de selección, que consiste en que la representatividad del modelo no varíe al ir cambiando el sistema sobre el que se procesa.

Se llama *sesión de medida* al intervalo de tiempo en que se realiza un experimento. Una sesión de medida no tiene por qué ser continua en el tiempo, sino que puede estar compuesta por varias subsesiones que se realicen en distintos días y a distintas horas.

Ejemplos de ello serían cuando interesara medir las prestaciones del sistema, cuando su carga sobrepasara cierto nivel (para estudios de saturación), o cuando se produjera la conexión a determinada red remota (para estudiar la carga que ésta produce).

3.2. Representatividad de un modelo de carga

El apartado anterior ha puesto de manifiesto la necesidad de construir cargas de prueba o modelos de la carga, como resultado del proceso de su caracterización. La precisión del modelo de la carga es claramente una característica esencial para su credibilidad y, por tanto, para su utilidad práctica. Cuando se usa un modelo para representar la carga real de un sistema, su precisión se conoce a menudo como la representatividad del modelo.

La carga puede representarse a distintos niveles, correspondientes a aquéllos en que puede describirse un sistema informático. Es decir, no hay un único criterio para evaluar la representatividad de un modelo; dependiendo del nivel de modelado adoptado y de los objetivos del estudio, se seleccionará una definición adecuada de representatividad y un criterio consistente con su evaluación.

Tres de los más importantes niveles de modelado de un sistema informático y, por lo tanto, de su carga son el físico, el virtual y el funcional. Esto significa que no hay un criterio único para evaluar la representatividad de un modelo, sino que, en función del nivel de modelado adoptado, se tendrán unas características para modelar y unos criterios de representatividad distintos.

3.2.1. Representatividad a nivel físico

A nivel físico, el modelo de la carga se basa en los consumos absolutos o unitarios de los recursos hardware y software. Por ejemplo, cada componente básico de la carga puede caracterizarse por el tiempo de CPU consumido o por el número de instrucciones máquina ejecutadas, por los espacios requeridos de memoria principal y secundaria, por el tiempo total de E/S consumido, por el número de archivos utilizados, por el número y duración de los accesos físicos de E/S a canal y a disco, etc. Este nivel de modelado, que se basa en una caracterización orientada a recursos, depende fuertemente del sistema empleado y, por

consiguiente, puede usarse significativamente en los estudios de comportamiento que no alteren sustancialmente la configuración del sistema existente. En particular, es útil cuando se pretende evaluar la eficiencia de la sintonización de un sistema o determinar la capacidad residual de un sistema.

La mayoría de los parámetros usados en la modelización a este nivel están recopilados por las rutinas de contabilidad (*logging* o *accounting*) presentes en la práctica totalidad de los sistemas operativos. Por la facilidad de acceso a los datos, este tipo de modelado es muy popular.

Como consecuencia de todo ello, se dice que un modelo de la carga *W'* representa perfectamente la carga *W* si solicita los mismos recursos físicos en las mismas proporciones que *W*. Es decir, de acuerdo con este criterio de representatividad, el modelo de la carga deberá usar los mismos recursos (CPU, E/S, etc.) en la misma proporción que en la carga real. Además, el modelo de carga será mejor en cuanto utilice los mismos recursos físicos y en las mismas proporciones, por lo que será muy fácil de construir utilizando un método de proporcionalidad.

Las características de estos modelos son:

— Orientación al consumo de recursos físicos.
— Gran dependencia del sistema.
— Relativamente fáciles de construir (los datos que se necesitan para construir estos modelos son fácilmente obtenibles de los sistemas operativos de los equipos, o pueden obtenerse fácilmente con monitores).

Las mayores aplicaciones de estos modelos son:

— Sintonización del sistema (modificación del consumo de recursos para un mejor funcionamiento del sistema).
— Planificación de la capacidad residual (qué tanto por ciento de la potencia del equipo no está siendo utilizada).
— Diseño.

3.2.2. Representatividad a nivel virtual

En este nivel se consideran los recursos del sistema a nivel lógico. Por ejemplo, cada componente básico de la carga puede caracterizarse por el número de sentencias de cada tipo del lenguaje de alto nivel, el número de accesos lógicos a cada archivo o base de datos, el número y tipo de órdenes interactivas, etc. A causa de este tipo de parámetros, los modelos de este nivel están próximos al punto de vista del programador y son menos dependientes del sistema que los modelos a nivel físico. Estos modelos son adecuados para aquellos estudios en que la modificación de la configuración o del sistema pueden hacer que alguno de los parámetros del anterior nivel de representación pierda significación. Los parámetros de este nivel de representación son bastante más difíciles de obtener que en el tipo de modelo anterior y son necesarios instrumentos especiales para medir muchos de ellos. Este tipo de modelos

acostumbra a utilizarse en estudios de ampliación, de determinación de la capacidad de un sistema, etc.

Como consecuencia de ello, se dice que un modelo de la carga *W'* representa perfectamente la carga *W* si solicita los mismos recursos físicos con la misma frecuencia que *W*. Es decir, de acuerdo con este criterio de representatividad, el modelo de la carga deberá provocar por unidad de tiempo el mismo número de ráfagas de CPU y de E/S de la misma duración que en el sistema real.

Las principales características de estos modelos son:

— Orientación a recursos lógicos.
— Menor dependencia del sistema que en el modelo anterior.
— Mayor proximidad al punto de vista del usuario que en el modelo anterior.
— Mayor dificultad para obtener los parámetros para la construcción del modelo que en el modelo anterior.

Estos modelos son adecuados para:

— Estudios de ampliación, en los que se quiere prever el funcionamiento del sistema después de añadir nuevas unidades.

3.2.3. Representatividad a nivel funcional

En este nivel la carga viene determinada por las aplicaciones que la componen. En el modelo deben aparecer programas que efectúen las mismas funciones que en la carga original. Así pues, las funciones que componen la carga deben describirse de forma independiente del sistema. Los modelos de este tipo se usan en estudios de adquisición, donde deben compararse sistemas de arquitecturas distintas o cuando las diferencias en las configuraciones, en los modelos del tipo anterior, dejan sin sentido alguno de los parámetros.

Como consecuencia de ello, se dice que un modelo de la carga *W'* representa perfectamente la carga *W* si realiza las mismas funciones con las mismas proporciones que *W*.

Por ejemplo, si un determinado sistema realiza 400 horas de compilaciones, 250 horas de pruebas de trabajos, 700 horas de explotación de trabajos y 300 horas de cálculo científico, un modelo funcional de la carga sería el que usara 40 minutos de compilaciones, 25 minutos de pruebas de trabajos, 70 minutos de explotación de trabajos y 30 minutos de cálculo científico.

Las principales características de estos modelos son:

— Orientación a aplicaciones.
— Independencia del sistema.
— Dificultad de diseñar sistemáticamente.

Y las principales aplicaciones de estos modelos son:

— Selección de un computador.
— Planificación de la capacidad.

3.2.4. Representatividad a nivel de comportamiento

Puesto que los modelos de la carga se usan normalmente para la evaluación del comportamiento del sistema, parece evidente usar, para la caracterización sobre la que se basa el modelo, parámetros cuyas relaciones con el comportamiento se conozcan completamente. Los modelos que satisfacen alguno de los criterios de representatividad precedentes pueden estar basados en parámetros cuya relación con los índices de comportamiento sea desconocida.

Si, por ejemplo, se está interesado en el tiempo de respuesta y la productividad de un sistema *batch*, puede no ser conocido, y además difícil de determinar, el impacto de parámetros, tales como el tiempo medio de CPU por programa, el número medio de operaciones de E/S, el número medio de tarjetas leídas, etc., en los valores de los índices del comportamiento considerados.

Además, normalmente se ignoran los aspectos dinámicos de la carga. Por tanto, no se intentan reproducir las mezclas de programas y su evolución en el tiempo. La carga se considera como un conjunto de programas. El criterio virtual permite la evaluación de algunos aspectos que se ignoran totalmente en el criterio físico. Por ejemplo, el cálculo de la ráfaga media de CPU requiere recoger más información dinámica acerca del comportamiento del programa que el del tiempo total de CPU consumido por un programa.

Por consiguiente, un criterio que permite evitar alguno de los inconvenientes anteriores es el de la evaluación del comportamiento, en el que se dice que un modelo W' representa perfectamente la carga W, si produce los mismos valores de los índices de comportamiento que W, cuando se ejecuta en el mismo sistema.

Es decir, se considera como criterio de la precisión del modelo la diferencia que pueda existir entre los valores de los índices de comportamiento que produzcan la carga y su modelo.

El uso de este tipo de criterio permite evaluar la validez del modelo por los efectos que produce en el sistema y no solamente por sus posibles causas. Desafortunadamente este criterio produce modelos dependientes del sistema.

3.2.5. Compacidad del modelo

Esta propiedad de los modelos de la carga tiene en cuenta el factor por el que se reduce su tiempo de ejecución respecto a la carga real. De esta forma, la construcción de modelos de la carga relativamente compactos satisfaciendo los criterios físico o virtual es relativamente directa, ya que, a grosso modo, consiste en reducir proporcionalmente el número de programas que hay que ejecutar en cada uno de los grupos homogéneos de programas que se pueden considerar en la carga real.

La obtención de un modelo compacto que tenga las mismas características funcionales que la carga original ya no es tan simple, puesto que no resulta claro lo que puede significar la reducción proporcional de varias aplicaciones. En algunos casos puede lograrse por reducción del volumen de datos que hay que procesar; en otros casos, debe reducirse el tamaño de los programas; a veces

deben seleccionarse unos datos de entrada elegidos para el caso; o también por una astuta mezcla de los métodos citados. Además, cuando se selecciona un criterio cuantitativo para la representación proporcional, se introducen dependencias no deseadas del sistema, ya que la caracterización funcional es cualitativa por naturaleza y es natural usar tiempos ligados a recursos como medidas de los factores de reducción.

El enfoque orientado al comportamiento puede producir generalmente modelos que satisfagan razonables exigencias de compacidad. Una ventaja de estos modelos es que las mezclas de programas y sus efectos en el comportamiento del sistema se toman en cuenta implícita y automáticamente por el método usado en su construcción. Sin embargo, las mezclas reales presentes en el modelo y su dinámica tienen ciertamente un impacto no despreciable en la precisión del modelo cuando sé usa para evaluar un sistema modificado. Es decir, incluso en un modelo orientado al comportamiento, el realismo en la reproducción de las mezclas y de su dinámica es una exigencia importante.

En conclusión, no hay un criterio único y absoluto para construir y evaluar modelos de la carga por lo que el nivel de caracterización y sus parámetros deberán seleccionarse de acuerdo con los objetivos del estudio y los datos disponibles.

No obstante, debe observarse que la elección de un criterio no excluye la toma en consideración de los demás, y es frecuente considerar la construcción de modelos basados en la combinación de varios de ellos, como, por ejemplo, cuando se combina el orientado al comportamiento con alguno de los demás.

3.3. Cargas de test o de prueba

La carga que procesa el sistema durante la sesión de medida, tal y como se ha comentado anteriormente, puede ser o bien la carga real de ese sistema informático o bien un modelo de esa carga.

En la siguiente tabla están resumidos los principales tipos de cargas de test:

Cargas de prueba			
	Real		
	Sintético	Natural Híbrida	
	Artificial	Ejecutable	*Mix* *Kernel* Programa sintético Secuencias transoccionales Secuencias conversacionales *Benchmarks*
		No ejecutable	Distribuciones estadísticas Trazas para simulación

3.3.1. Cargas de test reales

La carga de test que consista en la carga que se está procesando realmente en el sistema durante un sesión de medición es, al menos potencialmente, la más representativa y la más barata de implantar. Se puede considerar como un modelo de la carga ya que la duración de su ejecución es normalmente mucho más corta que la de la carga real a la que representa. La única elección que debe realizar el encargado de modelar la carga es decidir qué porción de tiempo de ejecución de la carga real es el que debe usarse en los experimentos de medición.

La duración de una sesión de medida es función de los objetivos del estudio, de la naturaleza de las aplicaciones, etc. En algunos casos, los objetivos del estudio imponen ciertas limitaciones en cuanto al instante de comienzo y a la duración. Por ejemplo, si se quiere estudiar el sistema cuando está muy cargado, cuando se produce la conexión a una red remota, etc.

Si los objetivos del estudio no imponen determinadas elecciones, la duración de la sesión debe elegirse utilizando técnicas estadísticas. En el caso de que las variables medidas (es decir, las observaciones) sean estadísticamente independientes, se puede determinar el mínimo número de observaciones (el tamaño de la muestra) que garantiza que las estimaciones están afectadas de un error máximo dado (el intervalo de confianza) con una probabilidad mínima dada (el nivel de confianza). Se trata, pues, del clásico problema de determinar el mínimo tamaño de la muestra que corresponde a una precisión dada de los resultados.

La suposición de un prerrequisito, del que todavía no se ha hecho mención, es que la secuencia de observaciones de todas las variables es estacionaria; es decir, su distribución estadística no varía a lo largo del tiempo. Esta suposición es difícil de comprobar y demostrar, aunque en general se admite como válida, a falta de mejor hipótesis.

Una de las principales dificultades de usar cargas de test reales es la de reproducir situaciones cuando se quiere repetir un experimento, debido a los imponderables (especialmente humanos) que influyen en la dinámica del sistema. Las principales razones que limitan el uso de cargas de test reales en experimentos que deban repetirse en las mismas condiciones son, pues:

— *Falta de flexibilidad,* debido a la imposibilidad de modificar los programas y sus consumos de recursos.
— *Necesidad de reutilizar los datos originales* (archivos, bases de datos, etc.) cuando se vuelven a ejecutar los programas reales; todos estos datos deberán, por consiguiente, copiarse en una memoria secundaria para su restauración posterior, con los costes económicos e interferencia subsiguientes.
— *Confidencialidad de ciertos programas y datos,* que pueden llegar a prohibir su duplicación y forzar su sustitución por programas y datos que tengan similares características de comportamiento.
— *Diferencias en la organización del hardware y el software* de los distintos sistemas o de las distintas versiones del mismo sistema que hacen uso de la carga de test real, especialmente en los estudios de selección.

3.3.2. Cargas de test sintéticas

Existen dos tipos de carga de test sintéticas (en el sentido de reducidas y no de artificiales), las cargas de test sintéticas naturales y las híbridas.

a) Cargas de prueba sintéticas naturales

Se dice que una carga de test es sintética natural, o un *benchmark* en sentido estricto (ya que hay pruebas definidas con programas artificiales que también reciben este nombre), si consta de un conjunto de programas extraídos de la carga real. Aunque el *benchmark* puede usarse en cualquier tipo de estudio de comportamiento, su principal campo de aplicación es el de los estudios de selección y ampliación.

Algunas de las dificultades, que se encuentran a menudo en la práctica del uso de estas cargas, dependen de condiciones externas a los programas considerados que, no obstante, pueden tener un notable impacto en el comportamiento medido en distintos sistemas. Por ejemplo, y sin que esta lista tenga carácter exhaustivo sino simplemente como sugerencias:

— *Prioridad de ejecución:* Cada sistema trata las prioridades de forma distinta; en algunos sistemas, la prioridad asignada a un programa por el planificador se altera según distintas políticas durante la ejecución, lo que puede afectar a la secuencia de ejecución y a la mezcla de trabajos deseadas, y, en consecuencia, al comportamiento del sistema.
— *Máximo factor de multiprogramación:* El concepto de este parámetro varía de un sistema a otro y puede influir sustancialmente en los índices de comportamiento considerados.
— *Rutinas de contabilidad:* Puesto que estas rutinas son a menudo fuentes de datos para estudios de evaluación y que cada sistema tiene sus propias rutinas, sucede, con frecuencia, que bajo un mismo nombre se esconden conceptos y variables distintas, por lo que es conveniente asegurar el significado de cada valor para poder darle el significado correcto.
— *Parámetros de generación del sistema operativo:* Cada sistema operativo tiene diversos parámetros de generación, cuyos valores pueden asignarse; los valores de estos parámentros afectan al comportamiento del sistema. Este es el caso también de otros parámetros que pueden escogerse por el responsable de la instalación o por los usuarios, por ejemplo, los métodos de acceso, las conexiones de los dispositivos a los canales, las opciones de un compilador, la situación en la jerarquía de memorias de los módulos del sistema operativo y de los archivos, etc.
— *Lenguaje de programación:* A menudo, los fabricantes amplían los lenguajes de programación con suplementos no estándar que facilitan la construccion y, en algunas situaciones, mejoran el comportamiento de los programas; no obstante, estas ampliaciones no existen en todos los fabricantes, o, aun existiendo, no tienen exactamente la misma función. Por

todo ello, si lo que se pretende es evaluar la potencia de un equipo (y no la habilidad de los programadores de un fabricante), es conveniente que los programas que constituyen el *benchmark* estén escritos en forma tal que se respete la definición estándar del lenguaje de programación.

Estos aspectos, y otros no explicitados, dan una idea de las dificultades que aparecen en la definición de un *benchmark*. Además, el cliente suministra, normalmente, con los programas las reglas que deben seguirse en su ejecución. Es decir, en un *benchmark* deben definirse descripciones detalladas de los objetivos del cliente, de las reglas operativas para su ejecución, de los resultados que deben presentarse (listados de compilaciones y resultados, listados de consola, copias en cinta de programas y datos, ...), de las mediciones que deben recogerse durante cada una de la ejecuciones, etc. En particular, una forma razonablemente eficaz de plantear el *benchmark* consiste en confiar su realización completamente a los fabricantes y solicitar una copia en soporte magnético (cinta o diskette, por ejemplo) de los componentes del *benchmark*, para, una vez efectuada la selección y con el sistema instalado, comprobar la veracidad de las pruebas realizadas, estableciendo las cláusulas de penalidad convenientes para resarcirse en caso de falsedad (compensaciones económicas o en material, devolución del sistema, etc.).

Si el problema en cuestión es el de selección de un paquete de software, el planteamiento se simplifica, puesto que las pruebas se realizan sobre el propio equipo (que se supone perfectamente conocido), y la dificultad queda circunscrita a la definición de los datos que debe manipular y a la del entorno de ejecución.

b) Cargas de prueba sintéticas híbridas

En numerosas situaciones, la carga que se pretende modelar no existe completamente (incorporación de nuevas aplicaciones todavía no desarrolladas, crecimiento vegetativo de las existentes, etc.). Entonces es preciso adoptar soluciones híbridas consistentes en representar la carga conocida por un conjunto de programas extraídos de ella y la carga no existente mediante alguno de los elementos artificiales que se describirán en el apartado siguiente (*kernels*, programas sintéticos, secuencias conversacionales, etc.), para constituir lo que se denomina una carga de test híbrida.

3.3.3. Cargas de test artificiales

Un modelo artificial de una carga dada consta de componentes diseñados con el propósito de cargar el sistema real o un modelo matemático de él. Es decir, ningún componente de la carga real del sistema forma parte de una carga de test artificial.

A) Cargas de prueba artificiales ejecutables

1. INSTRUCCIÓN DE SUMA

Históricamente fue la primera carga de test utilizada. En un principio, el procesador era la parte más importante y más cara del computador, por lo que su

velocidad determinaba las prestaciones del sistema. En este contexto, la instrucción de suma era la más utilizada y tenía sentido emplearla como carga de test para la evaluación de las prestaciones del sistema.

A medida que los sistemas fueron evolucionando, creció el número y la complejidad de las instrucciones y se tuvo que recurrir a otro tipo de cargas de test.

2. *MIX* DE INSTRUCCIONES O DE SENTENCIAS

El análisis de las frecuencias de ejecución de las instrucciones máquina en una carga real proporciona una información útil para su caracterización, ya que cualquier programa que tenga las mismas frecuencias de ejecución de instrucciones que un programa (o conjunto de programas) puede considerarse como un modelo preciso de este programa (o conjunto de ellos) con respecto al consumo de CPU. Un modelo de este tipo, un *mix* (mezcla) de instrucciones, puede constar de un único o de varios programas cuyas frecuencias de ejecución de instrucciones coincidan con las de la carga total que se está modelando.

Un ejemplo de *mix* de instrucciones sería el siguiente:

Tipo de instrucción	f_i	t_i	$f_i \cdot t_i$
R_x " R_y	0.15	15	2.25
R_x " M	0.11	10	1.10
/, *	0.05	20	1.00
+, −	0.09	18	1.62
salto condicional	0.30	15	4.50
salto incondicional	0.15	15	2.25
otras	0.15	25	3.75

La primera columna es función de la carga (las frecuencias de las instrucciones serán función de los programas que formen ésta). La segunda columna es función del procesador (sistema) que se esté estudiando. La tercera columna defineuna medida de cuánto tiempo se emplea en realizar cada uno de los diferentes tipos de instrucciones en ese sistema bajo esa carga. Si se suman estos tiempos, se obtiene el tiempo medio de ejecución de ese *mix* en el sistema. Este tiempo sería:

$$t = \sum_i fi \cdot ti$$

El valor de t puede utilizarse para comparar el poder computacional de diferentes CPUs o la eficiencia de diferentes compiladores.

Se pueden utilizar unos pesos para poder caracterizar mejor ciertas aplicaciones; por ejemplo, en una aplicación científica se pueden valorar más las operaciones aritméticas que en una aplicación de gestión.

Los *mix* de instrucciones fueron las primeras cargas de prueba que se utilizaron para evaluar la potencia de cálculo de las CPUs. Para comparar dos CPUs, deberían utilizarse *mix* que fueran lo más independientes posible del sistema, pero esto es muy difícil dado que es un programa que depende mucho de aspectos de la arquitectura de la CPU, tales como:

— Gestión de la jerarquía de memoria (gestión de memoria caché, etc.).
— Secuenciamiento de las instrucciones (prebúsqueda, segmentación, etc.).
— Manejo de direcciones (cálculo de la dirección efectiva).

En resumen, se puede decir que su misma simplicidad es su defecto, ya que, aparte de modelar únicamente los aspectos de consumo de CPU, sólo puede usarse para comparar sistemas con la misma arquitectura juego de instrucciones, que usen el mismo sistema operativo y los mismos compiladores, ya que de lo contrario se puede caer en diferencias muy notables en la representación de la carga. Es decir, por su mismo concepto, se trata de una representación extraordinarimente ligada a un sistema.

Existen numerosas definiciones de *mix* hechas por fabricantes, asociaciones de usuarios, etc., entre las cuales una de las más famosas entre las orientadas a aplicaciones es el Gibson-*mix*. Todas ellas deben considerarse como orientativas de la velocidad de la CPU de los distintos sistemas y nunca deben tomarse como elemento decisivo en la selección de un equipo o en cualquier otra comparación, ya que, como se ha dicho, no tiene en cuenta ni las diferencias de arquitectura de la CPU, ni el resto del sistema, ni el sistema operativo en sentido amplio, ni los lenguajes y sus compiladores, aspectos todos ellos que tienen un peso importante en el comportamiento de un sistema.

Una extensión del concepto de *mix* a lenguajes de alto nivel son los *mix* de sentencias, en los cuales se construyen programas que tengan la misma frecuencia de aparición de las distintas sentencias del lenguaje de alto nivel considerado que la carga que se está modelando. Lógicamente, el empleo de un *mix* de sentencias es más independiente del sistema. Desgraciadamente, este tipo de *mix* se hace dependiente del compilador, ya que según lo optimizado que éste esté, cada sentencia generará un número distinto de instrucciones máquina y, por tanto, tendrá un tiempo de ejecución distinto.

Si se mide la frecuencia de aparición de las instrucciones o de las sentencias para construir un *mix*, se debe tener en cuenta que esta frecuencia de aparición puede obtenerse de dos formas:

— *Estáticamente.* En estos casos, se mide la frecuencia de aparición de las sentencias en los listados (fuente) de los programas que componen la carga.
— *Dinámicamente.* En esta ocasión, se calcula la frecuencia con que realmente se están ejecutando las sentencias.

Evidentemente, es mucho más exacto el segundo método, aunque también es más difícil obtener los datos acerca de las frecuencias de aparición.

3. *KERNELS*

Un *kernel* es un programa o fragmento de programa que representa lo más característico de la carga. Son programas cerrados, en los que no se pueden realizar ajustes y tienen un consumo de recursos conocido y no modificable; son, pues, programas no paramétricos, que normalmente proporcionan asociaciones de usuarios o agencias gubernamentales.

Ejemplos de *kernels* que se han venido empleando son la función de Ackermann, Sieve, programas de inversión de matrices, programas de ordenación, etc.

Los *kernels* deben seleccionarse en función de su similitud con los programas que constituyen la carga del sistema. En algunos casos, estos programas, debido a que ocupan poco espacio de memoria, pueden almacenarse completamente en la memoria caché, que produce unos tiempos de ejecución muy cortos. En este caso los resultados son poco significativos y pueden inducir a incorrectas interpretaciones.

4. PROGRAMAS SINTÉTICOS

Un paso más en la tarea de adecuar programas a las características de una carga son los programas sintéticos. Estos programas no realizan ningún trabajo útil, sino que se limitan a consumir recursos del sistema. Estos recursos se consumen en cantidades determinadas por los parámetros de control de dichos programas. Los valores de los parámetros los elige el usuario del programa; normalmente determinan necesidades de recursos tales como el tiempo total de CPU, el número de operaciones de E/S, la cantidad de memoria requerida, el número de registros de entrada que se quieren leer, el número de líneas que se van a imprimir, el número y características de los archivos o bases de datos a los que hay que acceder, así como la cantidad y tipo de accesos lógicos que hay que efectuar a cada uno de ellos.

Una de las principales características de los programas sintéticos es su flexibilidad, que les permite simular un amplio espectro de programas reales desde el punto de vista del consumo de recursos. Por medio de bucles anidados, se pueden simular los ciclos CPU—E/S de un programa real especificando el número y tipo de instrucciones que debe ejecutar la CPU y controlando el número de operaciones de E/S que se han de efectuar. Un ejemplo sencillo de programa que permite modelar el consumo de CPU y de E/S es el siguiente:

```
for i:= 1 to N1 do
     "consumir CPU"
for i:= 1 to N2 do
     "consumir E/S"
```

Pueden construirse programas sintéticos con distintas estructuras para la simulación de distintos tipos de aplicaciones o bien puede recurrirse a productos comerciales que permiten una fácil adaptación a una amplia variedad de estructuras de programas.

En muchos casos, los programas sintéticos además de tener el conjunto inicial de parámetros de control ya mencionado, tienen uno o dos conjuntos suplementarios de parámetros denominados de corrección y de calibración. Los primeros se usan para eliminar o para reducir las distorsiones no deseadas de algunos consumos de recursos debido a los cambios en los parámetros de control que, idealmente, no deberían influir en otros consumos. Por ejemplo, cada vez que se modifica el número de operaciones de E/S requeridas por un programa, cambia también el tiempo total de CPU solicitado; este tiempo deberá, por consiguiente, corregirse para obtener así el valor del tiempo de CPU especificado por el correspondiente parámetro de control.

A diferencia de los parámetros de corrección, que están incluidos y por lo tanto son difíciles de modificar, los parámetros de calibración pueden usarse para un ajuste fino del programa sintético durante la fase de calibración. Esta fase es necesaria puesto que una combinación de valores de los parámetros de control, a los que corresponderá un consumo de recursos deseado, puede lograrse sólo raramente. Después de haber determinado la combinación de valores de entrada correspondiente a valores de los consumos de recursos muy próximos a los deseados, se modifican los parámetros de calibración para aumentar la precisión del programa sintético.

5. SECUENCIAS TRANSACCIONALES

Hasta aquí, de una forma más o menos implícita, se han estado considerando *benchmarks* naturales o artificiales de entornos de trabajo *batch*. Ahora bien, si el entorno del cual se pretende construir un modelo de la carga es de tipo transaccional o conversacional, es decir, de entornos donde un conjunto de usuarios remotos cree tener, cada uno de ellos, el sistema informático a su exclusiva disposición, se plantean varias dificultades suplementarias. Entre ellas, y tal vez en primer lugar, está la de conseguir movilizar un conjunto importante de personas, las cuales, además, difícilmente actuarán de forma idéntica en cada uno de los experimentos que se repitan y ello por errores debidos a imponderables personales (que, tal vez por azar, pudiera considerarse que se compensan) y por el efecto de aprendizaje que haría que probablemente en cada repetición se redujeran los tiempos de reflexión.

Por estas razones es conveniente evitar, en cuanto se pueda, estos efectos, para lo cual resulta preferible disponer de un esquema como el de la figura 3.3, en el que el sistema informático, cuyo comportamiento se quiere evaluar, contiene todo el software de sistema (sistema operativo, monitor de comunicaciones, etc.), de ayuda a la programación (monitor de transacciones o conversacional, sistema de gestión de base de datos, etc.) y de aplicación, mientras que el segun-

do simula el comportamiento de la red y de los usuarios de los terminales (transaccionales o conversacionales) conectados a ella. La simulación de cada uno de estos usuarios puede hacerse mediante un archivo que contenga los mensajes que enviaría sucesivamente al sistema de proceso con los tiempos de reflexión asociados a cada uno de los mensajes. La conexión entre ambos sistemas se efectúa con tantas líneas como las que tendría realmente el sistema cuyo comportamiento se está evaluando.

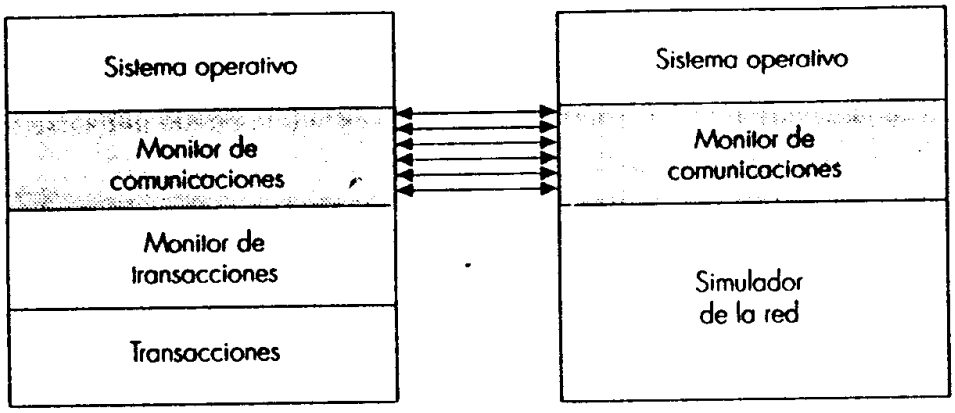

FIGURA 3.3.

Ahora bien, este montaje puede considerarse, cuanto menos, prolijo y caro, y, en muchos casos, imposible por no disponer del hardware suficiente, por lo que con frecuencia se recurre a la simplificación representada en la figura 3.4. En ella el propio sistema se encarga de la generación y del tratamiento de los mensajes, admitiendo, de forma gratuita, que el consumo de recursos en la simulación de los usuarios de la red será equivalente al del monitor de comunicaciones. A pesar del defecto mencionado, este esquema de trabajo resulta notablemente más fácil de implantar.

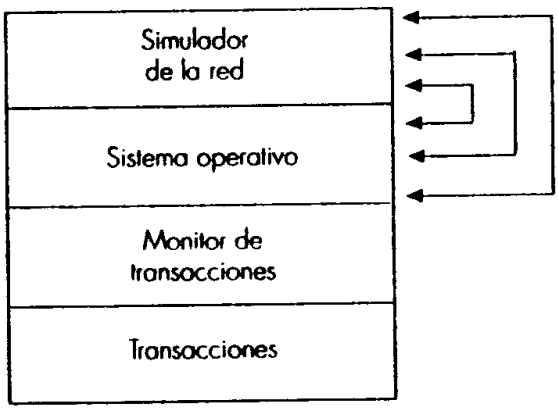

FIGURA 3.4.

Cuando el entorno de trabajo es de tipo transaccional, la simulación del comportamiento de los usuarios puede hacerse conservando la traza de mensajes que se envía desde cada terminal con su temporización. Dicha traza sería la que inyecta el simulador de la red al sistema para su tratamiento. En muchos casos, con el fin de simplicar el simulador, se procede a guardar una única traza con la totalidad de los mensajes que llegan al sistema sin detallar de qué terminal proceden. Es decir, se trabaja como si hubiera un solo terminal que enviara mensajes sin esperar su respuesta; variando de forma proporcional los intervalos entre mensajes se consigue simular con facilidad los aumentos de carga del sistema.

El monitor de transacciones puede obtener la traza de lo que se envía desde cada terminal con su tiempo asociado. Esta traza es la que puede utilizarse en el simulador de la red.

6. SECUENCIAS CONVERSACIONALES

Cuando se trata de establecer el modelo de un entorno conversacional de trabajo, normalmente orientado a la programación, no puede usarse el método de la traza que se acaba de exponer para los sistemas transaccionales, puesto que los mensajes, que incluyen órdenes de control dirigidas al sistema operativo, no son directamente transportables, en general, de un sistema informático a otro. Para evitar este problema, es preciso describir a nivel funcional el guión de trabajo de cada uno de los usuarios, denominado secuencia o guión conversacional. Cada uno de ellos debe adaptarse a cada uno de los entornos de trabajo ofrecidos por los distintos sistemas operativos, determinando la secuencia de mensajes que deberán enviarse desde cada terminal, con la temporización correspondiente.

Un ejemplo de lo que podría ser un guión o secuencia conversacional es la siguiente:

1. Conectarse al sistema.
2. Copiar el programa P desde un archivo permanente a uno de trabajo.
3. Editar el programa P desde el archivo temporal añadiendo 25 líneas de código, en las cuales habrá un cierto número de errores.
4. Renumerar la líneas de código.
5. Compilar el programa P.
6. Editar nuevamente el programa P para corregir los errores, que habrá detectado el compilador (que serán los que se han introducido en el paso 3).
7. Compilar de nuevo el programa P, dejando el módulo objeto en el mismo archivo temporal.
8. Montar el programa P usando módulos que existen en una librería permanente.
9. Ejecutar el programa P usando los datos que se hallan en los archivos correspondientes y dejando el listado de resultados en algún archivo accesible.
10. Editar el listado para analizar la bondad de los resultados.

11. Enviar a impresión el listado generado.
12. Copiar los módulos fuente, objeto y ejecutable del programa P en los archivos permanentes correspondientes.
13. Desconectarse del sistema.

Este guión debería organizarse de forma adecuada en cada entorno concreto, traduciendo a órdenes del sistema operativo las distintas funciones especificadas. Con el correspondiente programa que lance de forma controlada estos guiones, se podrán simular ambientes conversacionales y, como en el caso anterior, variando los tiempos entre epígrafes del guión y el número de guiones activos, se podrán simular todo tipo de situaciones.

7. *BENCHMARKS*

En la actualidad están siendo muy utilizados como cargas de prueba para la comparación de sistemas informáticos, unos programas estándar, construidos con alguno de los métodos anteriormente descritos, que simulan una carga genérica (a diferencia de lo que se ha hecho hasta ahora que era intentar modelar la carga de un sistema determinado). A estos programas o conjuntos de programas se les conoce con el nombre de *benchmarks*.

B) Cargas artificiales no ejecutables

1. DISTRIBUCIONES ESTADÍSTICAS

Cuando en un estudio de evaluación se adopta una técnica de modelado, la carga de prueba deberá ser compatible con esta técnica. Es decir, por ejemplo, si el sistema se modela como una red de colas, la carga debe describirse por parámetros tales como:

— la distribución de los tiempos entre llegadas de los programas
— el número de visitas de cada programa a cada recurso
— la distribución de los tiempos de servicio en cada recurso
— etcétera.

En estos casos, la construcción de un modelo de la carga parece sencilla, puesto que se trata sólo de asignar valores numéricos a los parámetros característicos. En realidad, estimar correctamente estos valores, sin introducir errores apreciables en los resultados, es una tarea muy difícil. La representatividad de modelos de la carga para modelos de redes de colas es, a menudo, bastante limitada, principalmente a causa de las hipótesis que deben introducirse en el modelo para hacerlo matemáticamente tratable.

En una carga de test probabilística, los parámetros se consideran usualmente como variables aleatorias independientes cuyas distribuciones pueden estimarse a partir de las distribuciones medidas en la carga real, cuando están disponibles, o aproximadas mediante funciones analíticas. Este tipo de datos también se utiliza en los modelos de simulación gobernada por las distribuciones.

2. TRAZAS PARA SIMULACIÓN

Otro tipo de carga de test que se utiliza a menudo para gobernar modelos de simulación es la traza. Una traza consiste en una secuencia cronológica, registrada en un soporte adecuado y representando la información referente a determinados tipos de acontecimientos que han acaecido en un sistema durante una sesión de medida.

Un acontecimiento es un cambio de estado de algún componente del sistema (como se vio en el capítulo 2 dedicado a los monitores). Ejemplos de acontecimientos orientados a la conducción de una simulación son los siguientes:

— la conexión de un usuario
— la llegada de un programa
— la iniciación de un programa
— la asignación de un dispositivo
— las peticiones de operaciones de E/S
— etcétera.

Variando proporcionalmente los intervalos entre acontecimientos consecutivos puede simularse fácilmente la variación de la carga.

La representatividad de una traza es bastante alta. Sin embargo, la recogida de datos para una traza puede provocar un *overhead* importante que distorsione su calidad, ya que los instantes e, incluso, el orden de aparición de los acontecimientos puede resultar alterado por la existencia del monitor encargado de obtener la traza.

3. COMPARACIÓN

La tabla siguiente resume la comparación entre las principales características de los modelos de carga reales y artificiales.

Característica	Carga real	Carga artificial
Representatividad	Potencialmente alta	Usualmente baja
Coste de realización	Bajo	Alto
Coste de uso	Bajo	Alto
Flexibilidad	Baja	Alta
Reproductibilidad	Baja	Alta
Compacidad	Baja	Alta
Independencia del sistema	Baja	Alta
Coste de recoger y reducir datos	Alto	Bajo
Aplicaciones	Sintonización	Diseño Sintonización Adquisición

3.4. Técnicas de implantación de los modelos de la carga

Como en la implantación de cualquier modelo, las operaciones para implantar modelos de la carga pueden agruparse en tres fases:

— formulación
— construcción
— validación

Los modelos de la carga pueden consistir, como se ha visto, en un conjunto de componentes (por ejemplo, programas) o pueden ser monolíticos (*mix* de instrucciones, trazas, ...). Los modelos del primer tipo son más populares y se aplican en la mayoría de estudios del comportamiento, mientras que los del segundo tipo tienen su aplicación limitada a algún problema de diseño de hardware o estudios de selección de CPU. Por ello, sólo se considerarán las técnicas para construir modelos del primer tipo.

La figura 3.5 muestra las principales operaciones que hay que realizar para la implantación de un modelo de la carga.

3.4.1. Fase de especificación

Esta fase consiste en una secuencia de decisiones que se pretenden tomar, evidentemente influidas por los objetivos del estudio. La definición de los componentes básicos de la carga, es decir, la identificación del menor nivel de detalle que debe modelarse y del conjunto de parámetros que debe usarse en su descripción, se encuentran entre las primeras operaciones que se deben realizar. El componente básico de la carga es la menor unidad de trabajo que se considerará y, como tal, pueden seleccionarse aplicaciones, trabajos, programas, guiones, órdenes, instrucciones de lenguaje de alto nivel, instrucciones máquina, etc. Evidentemente, cuanto más alto es el nivel del componente básico, tanto menor es el detalle con el que se describirá la carga. El análisis del uso previsto del modelo debe permitir definir el componente básico y, por consiguiente, influirá en la selección de los parámetros que deben usarse en su descripción cuantitativa.

Otro factor que influye en la elección de los parámetros es su disponibilidad. Muy frecuentemente en la implantación de modelos de la carga se usan los valores de los parámetros registrados por las rutinas de contabilidad del sistema, y sucede, a veces, que no están disponibles los del período deseado, o que los parámetros deseados no quedan registrados.

Así pues, los parámetros que deben usarse para caracterizar la carga se seleccionan basándose en el nivel de detalle del modelo, después de estudiar sus objetivos y verificar su disponibilidad. Si el modelo se construye a nivel de recursos físicos, pueden usarse parámetros tales como el tiempo de CPU, la memoria ocupada, el número de operaciones de E/S, el número de archivos de trabajo, el número de cintas, etc.; si, por el contrario, el modelo debe implantarse a nivel funcional, sus parámetros serán, por ejemplo, el número de compila-

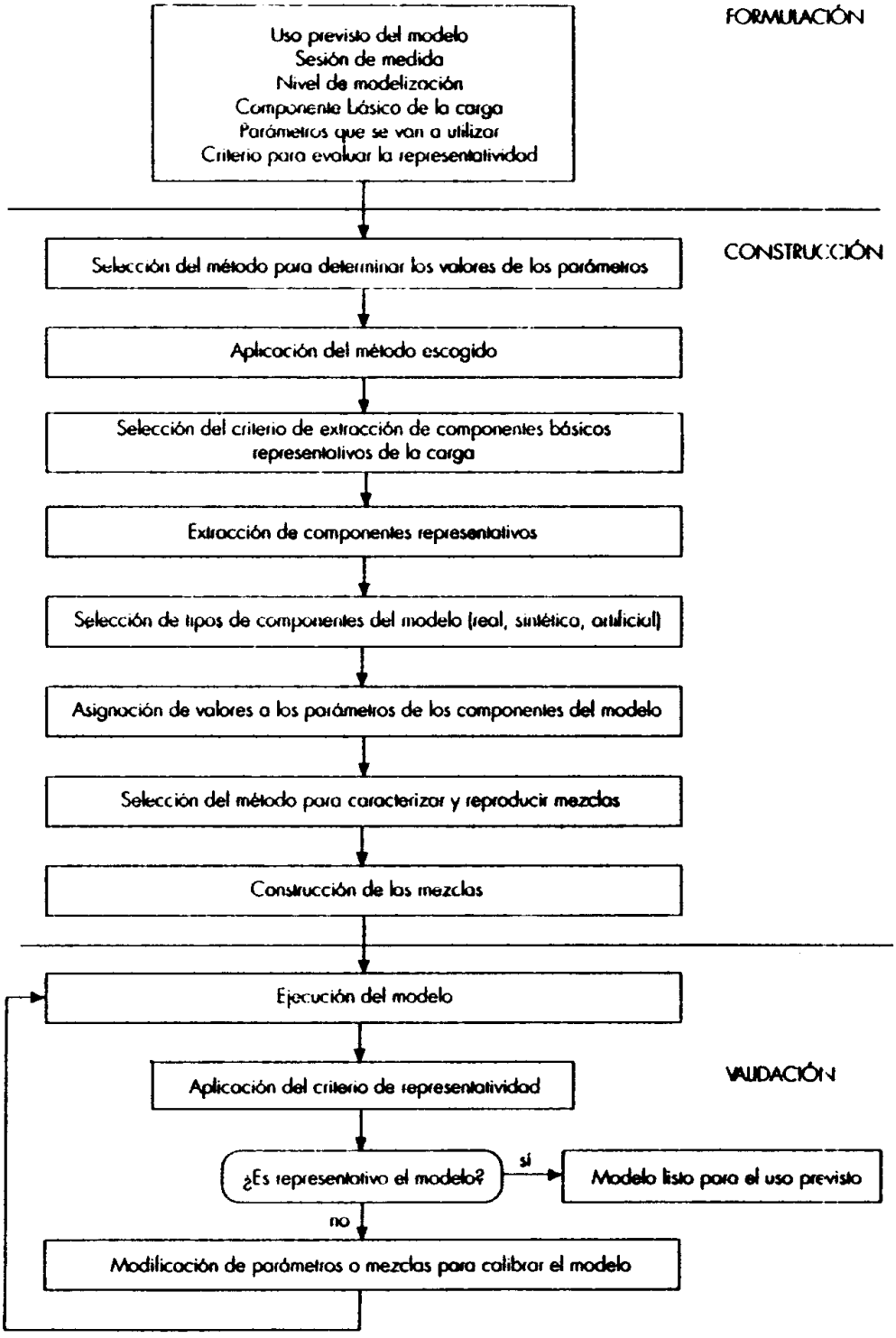
FORMULACIÓN
Uso previsto del modelo
Sesión de medida
Nivel de modelización
Componente básico de la carga
Parámetros que se van a utilizar
Criterio para evaluar la representatividad
CONSTRUCCIÓN
Selección del método para determinar los valores de los parámetros
Aplicación del método escogido
Selección del criterio de extracción de componentes básicos representativos de la carga
Extracción de componentes representativos
Selección de tipos de componentes del modelo (real, sintético, artificial)
Asignación de valores a los parámetros de los componentes del modelo
Selección del método para caracterizar y reproducir mezclas
Construcción de las mezclas
Ejecución del modelo
Aplicación del criterio de representatividad
VALIDACIÓN
¿Es representativo el modelo?
sí
Modelo listo para el uso previsto
no
Modificación de parámetros o mezclas para calibrar el modelo

FIGURA 3.5.

ciones y el de ejecuciones de programas FORTRAN y de programas COBOL, el número de ejecuciones de programas de servicio del sistema, el número de ejecuciones de rutinas de clasificación (SORT), etc. Puesto que estos valores se analizarán con el objetivo de identificar afinidades y agrupar componentes similares, es necesario prestar particular atención a la elección del número de parámetros que se van a usar y a la homogeneidad de sus valores. Si el número de parámetros es demasiado pequeño, puede que resulte difícil distinguir las distintas clases, mientras que si su número es demasiado elevado puede aparecer el problema contrario. Es evidente, además, que deben seleccionarse parámetros de tipo homogéneo, aunque, a veces, su falta de homogeneidad no es evidente y puede producir errores no deseados. Por ejemplo, en un enfoque orientado a recursos, si no se dispone del número de operaciones de E/S efectuadas, se puede usar en su lugar el valor del tiempo total de E/S, que incluye el tiempo de espera gastado en las colas de E/S en cada dispositivo, el cual depende del conjunto de programas en ejecución concurrente. Es decir, ciertos componentes pueden tener valores de sus parámetros que varíen con el instante en que se han ejecutado y esta situación puede provocar errores de agrupación.

3.4.2. Fase de construcción

Esta fase consta de las cuatro operaciones básicas siguientes:

— *Análisis de los parámetros:* Determinados, en la fase de especificaciones, los parámetros que caracterizarán la carga y el modelo, se comienza realizando la operación de medida sobre el sistema. Dicha operación se realiza con cualquiera de las técnicas de medida expuestas en el capítulo precedente y da como resultado un conjunto enorme de datos que puede considerarse como un conjunto de vectores en un espacio con un número de dimensiones igual al número de parámetros de la carga. Como la cantidad de datos es generalmente considerable, su análisis con el objetivo de identificar grupos de componentes con características similares deberá llevarse a cabo mediante técnicas estadísticas de análisis de datos para manipular muestras multidimensionales.

— *Extracción de valores representativos:* Para la extracción de valores significativos de entre una nube de datos, se utilizan técnicas estadísticas, de entre las que cabe destacar:
  - Muestreo de las distribuciones de probabilidad de cada parámetro.
  - Muestreo de la distribución de probabilidad conjunta.
  - Muestreo de las componentes de la carga.
  - Análisis de los componentes principales.
  - Algoritmos de agrupamiento.
  - Modelos markovianos.

  Estos métodos se analizarán más adelante dentro de este apartado, una vez se haya completado el estudio de la fase de construcción del modelo. No obstante hay que mencionar que el resultado de la aplicación de algu-

no de estos métodos será la determinación de las características de las clases componentes del modelo.

— *Asignación de valores a los componentes del modelo:* Trata de la transformación de los valores representativos en componentes ejecutables. Si se van a utilizar los componentes de la carga real o programas *kernel*, se tendrán que buscar aquéllos cuyos parámetros sean próximos a los valores de los componentes representativos. Si, en vez de ello, se usan programas sintéticos, se asignarán a sus parámetros de control los valores correspondientes de los componentes representativos y se deberá calibrar el modelo modificando los parámetros de calibración de los programas sintéticos. El número de componentes que constituirán el modelo influye directamente en su representatividad e, indirectamente, en su compacidad.

— *Reconstrucción de mezclas de componentes significativos:* El objetivo consiste en reproducir en el modelo situaciones similares a las que se producen en la carga real. La última operación en la construcción de un modelo de la carga consiste en la reproducción de las mezclas de la carga real mediante los componentes significativos. Por ejemplo, si nuestro objetivo es reproducir el comportamiento del sistema en las horas de punta, es preciso lograr, mediante el modelo de la carga, mezclas de programas en ejecución similares a las que se dan en esas horas. Por consiguiente, será deseable reproducir en el modelo una secuencia de componentes representativos similar a las que se encuentran en la carga real. Esto aumentará la verosimilitud de que se producirán las mismas situaciones de contención en los recursos críticos cuando el sistema trabaje con el modelo de la carga.

  Para reproducir las mezclas, se tendrán que analizar las secuencias de componentes de la carga como series temporales y estudiar las secuencias de valores que contienen. Un posible método consiste en clasificar los componentes en grupos con características similares y determinar los grupos a los cuales pertenecen los componentes en ejecución en determinados instantes de muestreo. De esta forma se obtienen n-tuplos de valores, cuyo análisis permitirá identificar las mezclas más frecuentemente ejecutadas.

Antes de entrar en la fase de validación, se van a estudiar los métodos que se pueden utilizar en la definición de las clases.

a) Muestreo de distribuciones de cada parámetro

Este método consiste en deducir los valores de los parámetros del modelo muestreando sus distribuciones separadamente. Las distribuciones de probabilidad de los parámetros se suponen conocidas por medida directa o por estimación.

Para fijar la composición del modelo de la carga, se empieza fijando el tamaño de la muestra que se desea construir, N, que debe elegirse de acuerdo con consideraciones estadísticas que permitan admitir la distribución de los paráme-

tros del modelo de la carga como una buena aproximación de la distribución de la carga original. A continuación, se seleccionan mediante valores pseudoaleatorios los valores de los parámetros a partir de las distribuciones de cada parámetro (figura 3.6). Cada número pseudoaleatorio se asigna a uno de los *N* componentes del modelo de la carga y sirve para extraer el conjunto de parámetros que caracterizarán el componente del modelo.

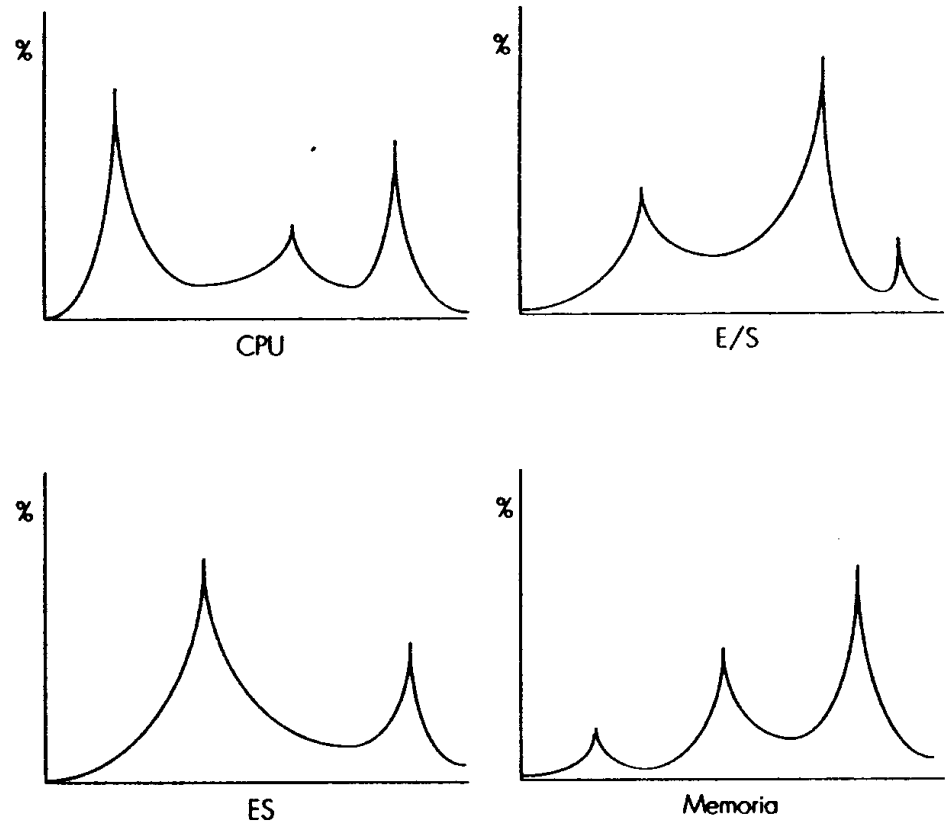

FIGURA 3.6.

Puesto que las distribuciones de los parámetros se han establecido separadamente, este método no tiene en cuenta las correlaciones que habitualmente existen entre los parámetros. En otras palabras, este método supone que el comportamiento del sistema no depende de las correlaciones entre los parámetros de los componentes considerados. Esta suposición normalmente no es válida y puede resultar que los modelos construidos den un comportamiento que esté bastante lejos del que produce la carga original. No considerar la correlación entre los parámetros de los componentes conduce a modelos que reproducen la utilización global de los recursos, pero, al trabajar con mezclas de programas

bastante distintos de los que se dan en la carga real, conducirán a valores distintos de los restantes índices de comportamiento.

Otra dificultad práctica aparece cuando se asignan a los componentes del modelo los valores obtenidos por muestreo de las distribuciones de los parámetros. Hay relaciones que deben respetarse entre los parámetros de un componente dado, como, por ejemplo, el número de operaciones de E/S impone un valor mínimo del tiempo de CPU, correspondiente al requerido para la preparación y gestión de aquellas operaciones.

La principal ventaja de este método es su sencillez de construcción (siempre y cuando se disponga del software estadístico adecuado) y además permite la fácil verificación del modelo, construido por este u otro método, por cuanto se conoce la distribución de los parámetros que se debe obtener.

b) Muestreo de la distribución de probabilidad conjunta

Este método permite eliminar la principal crítica del método del muestreo de cada parámetro, descrito en el apartado anterior. Aquí se muestrea la distribución conjunta de todos los parámetros de los componentes de la carga. Consta de los siguientes pasos:

1. Construcción de la distribución conjunta de todos los parámetros de la carga.
2. Muestreo de la distribución conjunta de los parámetros de la carga.
3. Uso de los componentes definidos por los parámetros obtenidos para construir el modelo de la carga.

El principal inconveniente del método es tener que trabajar con una distribución multidimensional. En algunos casos puede considerarse como un paso previo al uso de un método de agrupamiento.

c) Muestreo de los componentes de la carga real

El muestreo directo de la carga real elimina los problemas e inconvenientes que tienen los métodos expuestos en los apartados anteriores. Este método consiste en muestrear la carga real y asignar a cada componente del modelo los valores de los parámetros del elemento muestreado. De esta forma se conservan las correlaciones entre los parámetros que existen en la carga real. Otro aspecto importante de este método es la facilidad con que pueden construirse con componentes artificiales. Cuando se extrae un componente que no puede usarse directamente en el modelo, es relativamente fácil sustituirlo por un programa sintético o un *kernel* que tenga aproximadamente los mismos parámetros del componente extraído. Los aspectos críticos del método son los que hacen referencia al procedimiento de muestreo y al tamaño de la muestra.

El criterio del instante de muestreo consiste en extraer de la carga real aquellos componentes que se hallan en ejecución en el instante de muestreo. En este caso, la probabilidad de extraer un componente es proporcional a su tiempo de ejecución.

El criterio de muestrear uno de cada *n* consiste en extraer de la carga real uno de cada *n* componentes ejecutados en el sistema. Cada componente tiene, en este caso, la misma probabilidad de ser extraído, independientemente de las características que tenga.

El primero de estos criterios tiende a atribuir un peso excesivo en el modelo a los componentes con tiempos de ejecución largos. Por otro lado, el segundo de ellos da a estos componentes demasiado poco peso. Para verificar la validez de una muestra, pueden aplicarse desde sencillos cálculos estadísticos de la media y la varianza de cada parámetro hasta tests de la bondad del ajuste realizado.

El problema de la representatividad del modelo está claramente relacionado con el número de componentes extraídos. Este número no puede ser demasiado pequeño puesto que la carga que se debe modelar se compone normalmente de miles o decenas de miles de componentes. El tamaño de la población inicial y las varianzas de sus parámetros imponen la cota inferior del tamaño de la muestra. La realización de un modelo compacto, por el contrario, tiende a elegir valores relativamente grandes de los intervalos de muestreo o del valor de *n*. La selección correcta deberá constituir un equilibrio entre estas dos condiciones contrapuestas.

d) Análisis de las componentes principales

El término análisis factorial hace referencia normalmente a las técnicas estadísticas que describen conjuntos de datos multidimensionales mediante representaciones geométricas. El objetivo de este análisis consiste en elegir un subespacio del espacio de las variables de forma que la proyección del conjunto de datos sobre este subespacio preserve el máximo de información posible del conjunto original. Es decir, la propiedad importante del análisis factorial es su capacidad de reducción de la dimensión con poca pérdida de información.

Entre los métodos de análisis factorial, el análisis de las componentes principales es una técnica que transforma un conjunto de variables en otro conjunto (el conjunto de componentes principales) caracterizado por la dependencia lineal respecto de las variables del conjunto original y por la ortogonalidad entre sus propias variables. El criterio para elegir la primera de las componentes principales es el de maximizar la varianza de la función lineal que expresa la dependencia de las variables transformadas respecto de las variables originales. La segunda se elige para maximizar la varianza residual con la restricción que debe ser ortogonal a la primera y así sucesivamente.

En la práctica, esta técnica consiste en buscar los valores propios (en orden de módulo decreciente) y los correspondientes vectores propios asociados de la matriz de correlación de un conjunto dado de variables. El módulo relativo de cada valor propio da la varianza de la componente principal (es decir, del vector propio). El detalle de estos métodos puede hallarse en textos estadísticos convenientes, y para su cálculo pueden usarse los paquetes que lo desarrollan.

Suponiendo que la mayoría de los puntos se distribuyen en el espacio de las variables como una hiperelipse, los factores o componentes principales pueden considerarse como sus correspondientes ejes. En particular, el primer factor corresponde al eje más largo, el segundo al siguiente eje más largo y así sucesivamente. La importancia de un factor puede expresarse en función de la cantidad de varianza total de los datos que aporta. La varianza aportada por cada factor es la varianza de las distancias entre las proyecciones de los datos sobre el eje correspondiente al factor y el centroide de la hiperelipse. Puesto que cada factor se ha elegido para minimizar la varianza residual en los datos después de extraer los factores precedentes, sólo unos pocos de los primeros factores explican normalmente la mayor parte de la varianza de los datos. Por lo tanto, es muy importante determinar la fracción de varianza aportada por cada factor, tanto para tomar una correcta decisión del número de factores que se considerarán, como para evaluar el error debido a la reducción del número de factores.

La carga de una variable en un factor es la fracción de la varianza de la variable que aporta dicho factor. Si se representa por a_{ji} la carga de la variable j sobre el factor i, la varianza de la variable j aportada por el factor i viene dada por a_{ji}^2 y la fracción de la varianza total aportada por dicho factor es

$$\sum_{j=1}^{n} \frac{a_{ji}^2}{n}, \text{ para } i = 1, 2, \ldots, n$$

Debe observarse que la varianza de cada variable es 1 puesto que se han normalizado los valores de todos los parámetros; es decir, la varianza total coincide con el número n de variables considerado.

Los principales pasos requeridos por la aplicación de esta técnica y que siguen la mayor parte de los paquetes disponibles (nuestra única tarea es la de preparar los datos) son los siguientes:

1. Cálculo de las correlaciones entre las variables, es decir, entre los parámetros de los componentes de la carga.
2. Cálculo de los factores de carga iniciales de las variables en cada factor.
3. Optativamente, rotación (ortogonal u oblicua) de los factores iniciales para obtener los factores terminales. Este paso tiene por objeto producir mayores diferencias entre los factores de carga de forma que sea más fácil interpretar el significado de los distintos factores.

El análisis en componentes principales es particularmente útil en la validación de los resultados obtenidos por otras técnicas estadísticas y en el estudio de las principales causas de la variabilidad de las demandas de cada uno de los componentes de la carga, lo cual permite reducir la dimensión de cada componente de la carga, reduciendo las necesidades de obtención y almacenaje de datos y, posteriormente, el esfuerzo en su tratamiento para la definición de clases.

e) Métodos de agrupamiento

El objetivo de los métodos de agrupamiento es subdividir un conjunto de puntos en grupos o clases (*clusters*), de forma que las diferencias entre los miembros de un grupo sean inferiores (de acuerdo con un determinado criterio) que las que existen entre los miembros de grupos distintos. Estos métodos intentan representar la afinidad entre los puntos en función de estructuras tales como particiones o árboles.

El primer problema que se plantea en los métodos de agrupamiento consiste en la definición del criterio de similitud de los componentes de la carga. La forma más habitual es por la distancia que los separa.

Como paso previo a la aplicación de uno cualquiera de estos métodos, hay que realizar una normalización o reescalado. Como ejemplo de ello, la siguiente tabla refleja los resultados de una medición:

Programa a_i	Memoria ocupada	Cintas asignadas
1	54	1
2	88	0
3	120	2
4	110	2
5	64	0
6	68	1
7	112	1
8	56	0
9	64	1
10	90	4
11	60	1
12	110	1
13	60	0
14	116	1
15	118	2
16	150	2

Si se representa gráficamente esta nube de puntos, se obtiene la representación de la figura 3.7, utilizando la misma escala en ambos ejes. De esta gráfica se deduciría que existen cuatro grupos formados por los programas que poco más o menos ocupan la misma memoria. En este caso, la memoria determina el agrupamiento.

Si en vez de las escalas utilizadas se hubiera construido el gráfico con una escala de ordenadas diez veces mayor que la de abscisas, se obtendría el gráfico de la figura 3.8. De la observación de este gráfico se deduce que hay siete clases constituidas, respectivamente, por:

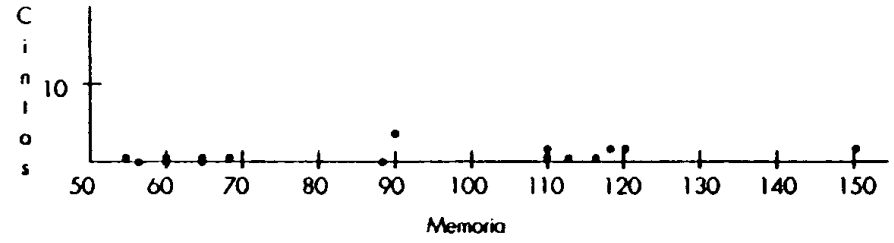

FIGURA 3.7.

— Clase 1: memoria pequeña y ninguna cinta (5, 8, 13)
— Clase 2: memoria pequeña y una cinta (1, 6, 9, 11)
— Clase 3: memoria intermedia y ninguna cinta (2)
— Clase 4: memoria intermedia y 4 cintas (10)
— Clase 5: memoria grande y 1 cinta (7, 12, 14)
— Clase 6: memoria grande y 2 cintas (3, 4, 15)
— Clase 7: memoria muy grande y 2 cintas (16)

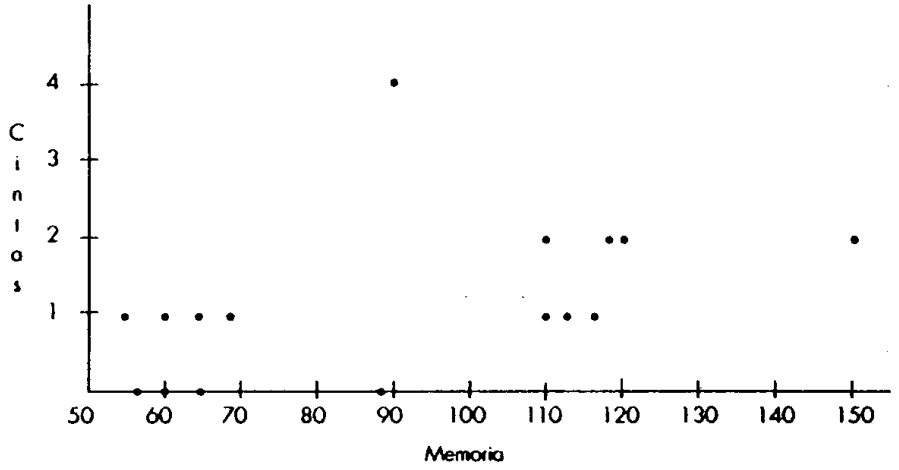

FIGURA 3.8.

Si en vez de las escalas utilizadas en el gráfico de la figura 3.7, se hubiera construido el gráfico con una escala de ordenadas cien veces mayor que la de abscisas, se obtendría el gráfico de la figura 3.9. De la observación de este gráfico se deduce que hay cuatro clases caracterizadas por el número de cintas.

Por consiguiente imprescindible una normalización o escalado uniforme para una correcta identificación de los grupos, es decir, que todas las variables se midan en unidades homogéneas. Existen diferentes métodos de realizar este reescalado, entre los que cabe citar los siguientes:

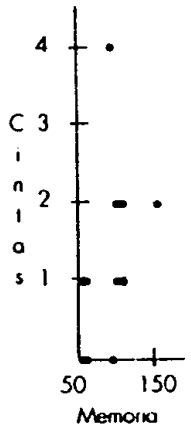

FIGURA 3.9.

— Las nuevas coordenadas se obtienen a partir de las antiguas en función de la media y la desviación típica. El valor de la nueva coordenada se obtiene a partir de la ecuación:

$$p'_{ij} = \frac{p_{ij} - m_j}{\sigma_j}$$

Donde $i = 1, \ldots, m$ (m es el número de valores), y $j = 1, \ldots, n$ (n es el número de parámetros), m_j es la media de los valores obtenidos en ese parámetro y σ_j la desviación típica.

$$m_j = \frac{\sum_{i=1}^{m} p_{ij}}{m}$$

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{m} p_{ij}^2}{m} - \frac{\left(\sum_{i=1}^{m} p_{ij}\right)^2}{m(m-1)}}$$

— Las nuevas coordenadas se obtienen a partir de las antiguas a partir de los índices de variación de cada coordenada. Este método da variaciones en las nuevas coordenadas entre 0 y 1. El valor de la nueva coordenada se obtiene a partir de la expresión:

$$p'_{ij} = \frac{p_{ij} - min(p_j)}{max(p_j) - min(p_j)}$$

donde $min(p_j)$ y $max(p_j)$ son el mínimo y el máximo valor obtenidos para ese parámetro.

— Las nuevas coordenadas se obtienen a partir de las antiguas a partir de unos pesos o ponderaciones. Dicha ponderación permite, además de escalar correctamente, resaltar la importancia de cierto parámetro frente a los demás. Dada su importancia, este método se puede utilizar en combinación con cualquiera de los anteriores. El valor de la nueva coordenada se obtendría a partir de la expresión:

$$p_{ij}' = p_{ij} * w_j$$

donde w_j es el peso asignado a ese parámetro. Criterios para seleccionar los pesos pueden ser:

- El peso de cada parámetro se basa en consideraciones de representatividad del modelo y, por tanto, en aspectos externos a los datos que hay que analizar.
- El peso de un parámetro es inversamente proporcional a su desviación típica de forma que se obtengan contribuciones al cálculo de las similitudes entre los puntos de cada clase que tengan el mismo orden de magnitud para cada componente.

— En casos extremos hay que utilizar transformaciones más drásticas, tales como logarítmicas u otras, para poner de manifiesto las características de los parámetros.

Una vez normalizados los valores obtenidos, se puede pasar a realizar el agrupamiento. Para ello, en primer lugar se debe seleccionar la distancia que representa la similitud entre los puntos (programas) que se quieren agrupar.

La más popular, y especialmente adecuada a magnitudes de tipo global (tiempo total de ejecución, memoria ocupada, etc.) en la descripción del componente, es la *distancia euclídea*. No obstante, en muchos casos se usan otros tipos de distancias tales como:

— la *distancia euclídea con ponderación* en los parámetros (la inversa de la desviación tipo, u otras),

— la *distancia de Mahalanobis* que esla distancia euclídea ponderada por una función de la varianza,

— la *distancia de ji-cuadrado,* adecuada a las variables que definen el perfil de un componente (tiempo de CPU en la ejecución del código del usuario, tiempo de CPU para las llamadas al supervisor, tiempo de E/S en cada tipo de dispositivos, etc.), es decir, buscando la similitud de componentes que tengan parecidos usos unitarios de recursos (figura 3.10). En esta figura, de acuerdo con la distancia euclídea, son similares el programa 1 y el 2 y de acuerdo con la distancia de ji-cuadrado lo son el 1 y el 3.

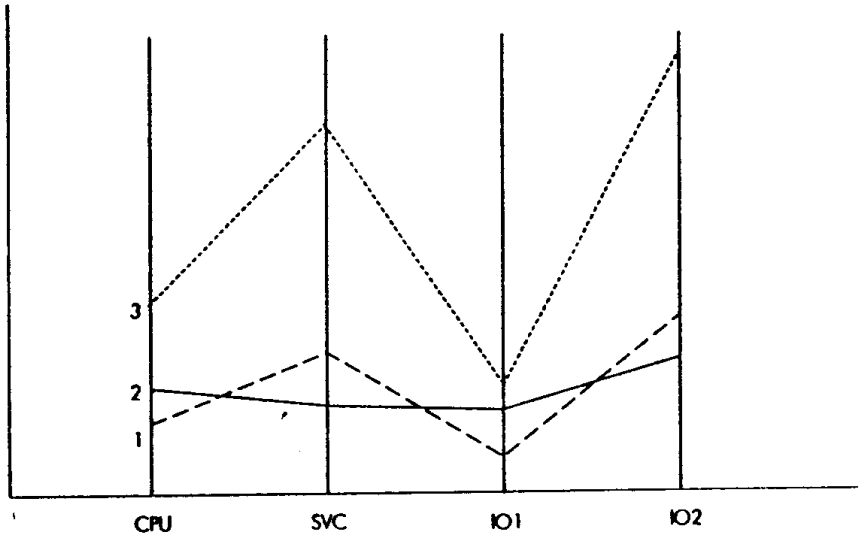

FIGURA 3.10.

No obstante, en muchos caso casos, la selección de la técnica (tipos de normalización y de distancia) no se conoce a priori, por lo que se deben efectuar distintos ensayos con diferentes técnicas hasta encontrar aquella que se adecua a las características de los datos que se están manipulando.

Incluso antes de seleccionar la técnica de normalización es conveniente eliminar los puntos singulares (*outliers*) que corresponden a componentes de la carga con un valor disparado de algunos parámetros, los cuales en la realidad precisan de una planificación singular en su ejecución y hay que analizar con cuidado si deben intervenir en el modelo de la carga, atendiendo a la situación que se desea representar. El motivo de su eliminación es la posible distorsión que pueden provocar en el estudio de las distribuciones estadísticas de los parámetros.

Otros aspectos a tener en cuenta son el tiempo de cálculo y la memoria necesarios para la determinación de las clases a causa del enorme número de componentes (millares o decenas de millares) que pueden encontrarse en una instalación y que son los datos que deben suministrarse al algoritmo de clasificación. Para reducir la cantidad de datos que hay que manipular se pueden utilizar dos técnicas. La primera consiste en escoger sólo una muestra de los datos seleccionados por un procedimiento de muestreo aleatorio, para garantizar una representación proporcional de los distintos componentes de las clases del conjunto original: la segunda se aplica cuando puede admitirse que el comportamiento de los distintos componentes de la carga es estable (o parecido) en sus distintas ejecuciones y consiste en tomar como características de cada componente el valor promedio de sus distintas ejecuciones; una vez constituidas las clases agrupando componentes de comportamiento promedio parecido, se calculan las caracterís-

ticas de las clases ponderando cada componente por su frecuencia de ejecución. En ambas técnicas, la asignación posterior de la totalidad de los puntos a las clases definidas puede dar una medida de la calidad de la clasificación obtenida tomando como índice, por ejemplo, el número de componentes que no tienen una asignación clara o cuyos parámetros caen fuera del ámbito de variación de los parámetros de cada clase.

Otro aspecto relacionado con los anteriores en cuanto al tiempo de cálculo y la memoria necesarios para realizar la clasificación es el que se refiere a los parámetros que se consideran en ella. Además debe tenerse en cuenta que el número de parámetros repercute directamente en el coste de la recopilación de los datos y que un número elevado de parámetros puede hacer difícil sino imposible la obtención de la partición en clases deseada. La principal técnica utilizada para efectuar esta selección es la ya expuesta del análisis de las componentes principales (apartado 3.4.2.d).

Una vez fijados los componentes y su forma de definición, por un lado, y el criterio de similitud (distancia), por otro, sólo queda la aplicación del algoritmo propiamente dicho de clasificación. El objetivo es conseguir una separación en clases de forma que las varianzas dentro de cada clase sean las menores posibles y que las varianzas entre clases sean las mayores posibles. El algoritmo que permitiría determinar, para un determinado número de clases, cuál es la mejor partición posible es de tipo combinatorio (comprobar todas las posibles agrupaciones de los componentes en las clases); este procedimiento es absolutamente inaplicable desde el momento en que la población que se pretende clasificar rebasa alguna decena. Por ello se han desarrollado un conjunto de algoritmos heurísticos que, partiendo de conjeturas aceptables, proporcionan particiones en clases razonablemente buenas, pero sin tener nunca la certeza de haber alcanzado el óptimo, debido a que la variación de los parámetros de los componentes no permite, en general, particiones claras, lo cual hace que los límites de las clases sean bastante difusos. Los distintos métodos que se pueden utilizar para la obtención de las clases se pueden agrupar en globales, que, a su vez, se clasifican en jerárquicos y no jerárquicos, y en paso a paso.

Los métodos globales no jerárquicos se basan en realizar una subdivisión inicial del espacio en k clases e ir mejorando iterativamente el agrupamiento de los m elementos de n componentes desplazándolos de una clase a otra. El agrupamiento óptimo se alcanza cuando ya no se pueden desplazar elementos de una clase a otra. En definitiva lo que se intenta mejorar es el índice, en este caso definido con una distancia euclídea:

$$\sum_{c=1}^{k} \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} - y_{cj})^2 \cdot x_{ic}$$

donde y_{cj} es el centro de masas de una clase, definido por la expresión:

$$y_{cj} = \frac{\sum_{i \in c} p_{ij}}{m_c}$$

para $j = 1, \ldots, n$, y siendo m_c el número de elementos de la clase c, y donde x_{ic} es una variable booleana que vale 1 si el elemento i pertenece a la clase c y 0 en caso contrario.

Un algoritmo de agrupamiento no jerárquico, denominado de las nubes dinámicas, es el siguiente:

1. Se seleccionan inicialmente un cierto número reducido (por ejemplo, cuatro) de puntos al azar como centros iniciales de clase. Evidentemente la mejor o peor fortuna en la elección de los centros iniciales de clase puede condicionar el resto de la clasificación.
2. Se asigna cada uno de los puntos a la clase cuyo centro esté más próximo.
3. Se recalculan los centros de clase de acuerdo con los nuevos grupos formados.
4. Ir de nuevo a la fase 2 hasta obtener una clasificación estable.
5. Repetir el agrupamiento varias veces partiendo de nuevos centros de clase elegidos al azar.
6. Efectuar la intersección de las distintas clasificaciones obtenidas.

Otro popular grupo de métodos no jerárquicos lo constituyen los métodos iterativos que consisten en mejorar una partición inicial en clases por desplazamiento de los componentes de una clase a otra, de acuerdo con el criterio de calidad de la partición, y recalculando, después de cada cambio o de cada pasada sobre todos los componentes, los centros de cada una de las clases. La configuración óptima se obtiene cuando no se puede realizar ningún cambio o cuando la mejora del criterio de calidad de una iteración a otra es inferior a un umbral determinado. Este método se utiliza, con frecuencia, en la mejora de una clasificación previamente establecida por otro método.

Otro método no jerárquico es el que, partiendo de la matriz de distancias entre todos los puntos que se pretenden clasificar, procede al cambio de filas y columnas con el objetivo de agrupar en torno de la diagonal principal los valores inferiores de las distancias, quedando las clases constituidas por los bloques de valores pequeños que aparecen en la diagonal principal.

Entre los métodos jerárquicos se encuentra el método del árbol de distancia mínima, que es un método jerárquico descendente de agrupamiento basado en la construcción del árbol mínimo que conecta todos los puntos del conjunto. El peso asociado a cada arista es proporcional a la distancia considerada entre los dos puntos representados en los nodos que la limitan. La construcción del árbol de distancia mínima se realiza eliminando las aristas de mayor peso evitando el aislamiento de ningún punto. Una vez construido el árbol de distancia mínima, se puede proceder a la reducción del número de nodos por agregación.

Otro método jerárquico ascendente de agrupamiento es el que se logra partiendo de todos los puntos con un peso unitario, buscando los dos puntos más próximos y sustituyéndolos por otro con una masa que sea la suma de las de los componentes. Este algoritmo de agrupamiento se va repitiendo hasta dejar reducido a un punto el conjunto de todos ellos. La determinación del número de clases se logra determinando aquellos pares de puntos tales que la distancia que los separa es notablemente superior a las de las agrupaciones anteriores (figura 3.11). Esquemáticamente, este algoritmo ejecuta los siguientes pasos:

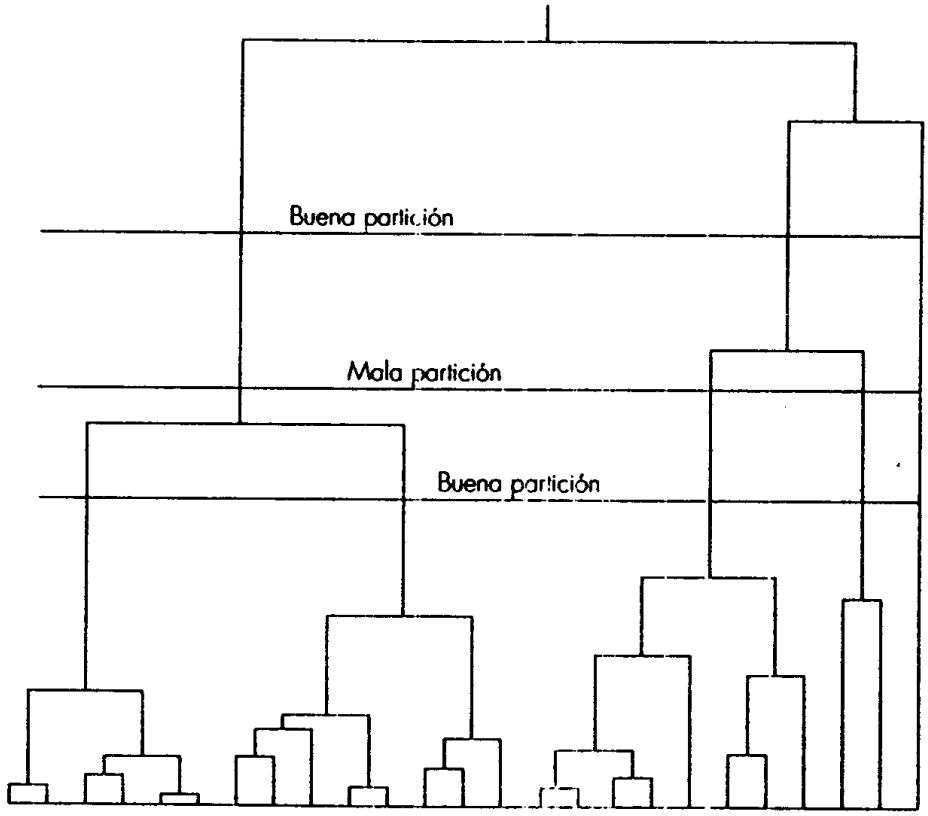

FIGURA 3.11.

1. Obtener la matriz de distancias.
2. Determinar la distancia mínima.
3. Fundir los elementos que determinan distancia mínima en uno solo.
4. Generar la nueva matriz de distancias (a partir de la anterior eliminando las filas y columnas de los elementos agrupados y añadiendo las correspondientes al grupo formado.
5. Ir al paso 2 hasta que:
   - Sólo exista un único grupo.

- Exista un determinado número de grupos.
- La distancia de agrupamiento sea superior a un determinado valor.

El problema es cómo reducir dos puntos en uno solo, teniendo en cuenta que uno o los dos anteriores pueden ser ya un punto equivalente a un conjunto de ellos agrupados con anterioridad (en una o varias agrupaciones anteriores), y posteriormente, calcular la distancia de este nuevo punto ficticio al resto, teniendo en cuenta la masa de cada uno de los puntos agrupados. Para ello se han propuesto diferentes técnicas, de entre las que cabe destacar:

— Encadenamiento simple

$$d_{c,(ab)} = min(d_{c,a}, d_{c,b})$$

— Encadenamiento completo

$$d_{c,(ab)} = max(d_{c,a}, d_{c,b})$$

— Grupo centroide

$$d_{c,(ab)}^2 = \frac{n_a}{n_a + n_b} d_{c,a}^2 + \frac{n_b}{n_a + n_b} d_{c,b}^2 - \frac{n_a n_b}{(n_a + n_b)^2} d_{a,b}^2$$

— Grupo medio

$$d_{c,(ab)}^2 = \frac{n_a}{n_a + n_b} d_{c,a}^2 + \frac{n_b}{n_a + n_b} d_{c,b}^2$$

donde a y b son los dos componentes agrupados en la iteración actual y c el componente para el cual queremos calcular la distancia al nuevo grupo formado por a y b. Las distancias $d_{c,a}$ y $d_{c,b}$ se puede obtener de la anterior matriz de distancias.

En el caso de las expresiones del grupo centroide y del grupo medio, n_a y n_b expresan el número de componentes de a y b, respectivamente. Lógicamente si se trata de la agrupación de dos componentes individiuales, tanto n_a como n_b valdrán uno. No ocurre lo mismo, en cambio, si alguno de ellos (o los dos) son fruto de un agrupamiento anterior.

Los métodos paso a paso tienen la ventaja de no requerir la totalidad de los datos simultáneamente. La definición de las clases y sus características puede hacerse de forma estática, es decir, a partir de un conjunto de clases de características definidas se van asignando los puntos a ellas a medida que se van presentando, o de forma dinámica, es decir, que las características de las clases se van modificando en función de los puntos que van llegando y también pueden crearse nuevas clases cuando el nuevo punto llegado queda demasiado alejado de las clases existentes en aquel momento. El principal inconveniente de este método es su inestabilidad, que le hace depender en gran medida del orden de llegada de los puntos.

La elección del algoritmo de agrupamiento más adecuado a un problema dado no es ni sencilla ni directa, sino que depende de los objetivos del estudio y de las características de los datos. Algunos algoritmos están adecuados a identificar y aislar clases de tipo elipsoidal, mientras que otros tienden a hallar clases más bien aplanadas. Por consiguiente, es recomendable tantear cuál de ellos conduce a la clasificación más adecuada a un estudiar determinado. En muchos casos, se usan en cascada varios métodos, ya sea para mejorar con el segundo la clasificación obtenida con el primero, ya sea para utilizar distintas distancias, etc. En otros casos se procede a la intersección de distintas clasificaciones obtenidas bien sea con distintas distancias, bien sea partiendo de centroides distintos.

Después de partir los componentes de la carga en clases homogéneas, se deben seleccionar los representantes de cada clase. Algunos de los criterios, que pueden adoptarse para la elección del número de representantes, consisiten en hacer este número:

— Proporcional al número de componentes (ejecuciones) de la clase.
— Una función del número y de la ponderación de los componentes de la clase.
— Proporcional a la influencia de la clase en el comportamiento.

Cada uno de estos criterios tiene aspectos positivos y negativos y es más o menos adecuado a la implantación del modelo de la carga dependiendo de los objetivos del estudio en el que debe usarse el modelo.

El primer criterio es claramente el más sencillo, pero puede eliminar los casos singulares presentes en la muestra, provocando, tal vez, un modelo insuficientemente representativo.

El segundo criterio parece más razonable, pero el problema aparece en el momento de asignar la ponderación a cada clase.

El tercer criterio es el más conveniente cuando el modelo se base en una caracterización orientada al comportamiento. La influencia de una clase sobre el comportamiento puede definirse de varias formas, dependiendo de los objetivos del estudio.

f) Métodos markovianos

Estos métodos pretenden determinar la frecuencia con que se encadena la ejecución de los componentes de la carga. Para ello se pretende la construcción de la matriz de probabilidades de transición entre las clases definidas por alguno de los métodos anteriores, para que sustituya a la que existe entre los componentes de la carga real.

3.4.3. Fase de validación

La evaluación del criterio de representatividad del modelo, basado en los objetivos del estudio definidos en la fase de formulación, debe efectuarse en la fase de validación para establecer la validez del modelo implantado. En esta fase no sólo se evaluará la representatividad del modelo con el conjunto de

parámetros determinado que se considera, sino que se deberán realizar experimentos para determinar el dominio de validez del modelo. La validación del modelo deberá efectuarse comparando su comportamiento con el de la carga real en aquellos puntos en que se conozca por haberse efectuado algún tipo de medición. La determinación del dominio de validez es, en general, extremadamente difícil, pero lo que puede afirmarse es que los modelos estructuralmente correctos y realizados cuidadosamente tienden a ser más válidos que los que no lo han sido.

Un modelo puede calibrarse modificando algunos valores de sus parámetros que influyen en su comportamiento, procediendo, iterativamente, a comparar los índices de comportamiento que produce el modelo con los de la realidad y a modificar los paráinetros de calibración hasta lograr el ajuste requerido.

3.5. *Benchmarks*

Aunque este tema se tratará con mayor detalle en el capítulo 6, algunos ejemplos de *benchmarks* estándar son los siguientes:

— *Benchmarks* de PCWORLD, publicados en esta revista. Son un conjunto de programas muy elementales que se pueden utilizar para comparar PCs o compiladores.
— *Benchmarks* de BYTE, publicados también en esta revista. Son unos programas más conocidos, compuestos de los siguientes:

1. Serie de Fibonacci
2. *Float test*
3. Criba de Eratóstenes
4. *Quicksort*
5. *Savage*
6. *Dhrystone*

— *Whetstone, benchmark* desarrollado para la comparación de prestaciones con operaciones en coma flotante.
— *Dhrystone,* quizás uno de los más conocidos, se emplea para ver las prestaciones de los sistemas en programación de aplicaciones.
— *Benchmarks* para estaciones de trabajo, compuesto por una serie de programas característicos del tipo de trabajo de estos sistemas.
— *Linpack, benchmark* que compara las prestaciones de los sistemas ejecutando las rutinas de una biblioteca de cálculo matricial.
— *The perfect club benchmark*, conjunto de programas reales que se emplean, preferentemente, para comparar prestaciones de supercomputadores.