Ambiente para especificación de requerimientos de sistemas de tiempo real.

Director:
DE GIUSTI, Armando E.

Alumno:
ROCCA, Paulo L.

PROYECTO FINAL
Junio de 1993

Indice.

Indice de figuras.

El contenido de este informe.

Una caracterización de los sistemas de tiempo real es presentada en el capítulo 1 poniendo de manifiesto la complejidad de estos sistemas y de su proceso de desarrollo. Se plantea entonces la necesidad de la especificación formal para controlar dicha complejidad.

El capítulo 2 define el alcance de este proyecto: la especificación de requerimientos del sistema, y su propósito: el desarrollo de un ambiente de especificación de requerimientos de sistemas de tiempo real.

En los capítulos que siguen se describen herramientas y metodologías de especificación de requerimientos, en base a las cuales se caracterizará y construirá el ambiente.

El capítulo 3 describe las herramientas usadas para especificar: las componentes de la especificación de requerimientos; el capítulo 4, distintas metodologías de especificación de requerimientos.

En el capítulo 5 se caracteriza un ambiente de especificación de requerimientos de sistemas de tiempo real, y en el capítulo 6 se describe un primer prototipo del ambiente implementado en Smalltalk/V.

Finalmente, en el capítulo 7 se enuncian las etapas que completan la especificación de un sistema de tiempo real.

Capítulo 1: Conceptos de sistemas de tiempo real.

1.1 Definición.

Un sistema de tiempo real puede definirse como un sistema controlado (por software o firmware) que desarrolla todas sus actividades (funciones) dentro de restricciones de tiempo especificadas [NIE90].

1.2 Clasificación.

Pueden distinguirse dos clases de sistemas de tiempo real:

- CLASE A: formada por los sistemas de tiempo real que no operan bajo restricciones de tiempo estrictas (sistemas blandos o "soft"). Entre ellos se encuentran los sistemas para reserva de pasajes, atención bancaria automatizada, etc. Proveen comunicación en línea entre el usuario y el sistema a través de terminales interactivas. Aun cuando se espera que los sistemas de esta clase respondan rápidamente, se tolera que el tiempo de respuesta varíe en función de la carga del sistema [MAG86, PET84].

- CLASE B: formada por los sistemas de tiempo real que operan bajo restricciones de tiempo estrictas (sistemas duros o "hard"). Entre ellos se encuentran los sistemas de control y supervisión de procesos. Cuando son estimulados por eventos externos deben responder en un tiempo finito y especificado [MAG86].

La funcionalidad de un sistema de tiempo real, cualquiera sea su clase, podría ser implementada totalmente en hardware, totalmente en software, o parte en hardware y parte en software. La implementación en software ,total o parcial, requiere del hardware sobre el que dicho software podría ser ejecutado [HAT88]. La tendencia actual hacia la implementación total en software obliga a caracterizar principalmente este tipo de sistemas de tiempo real.

1.3 Características.

1.3.1 Componentes: hardware y software.

Un sistema de tiempo real incluye usualmente un conjunto de dispositivos de hardware independientes que operan a diferentes velocidades. Estos dispositivos deben ser controlados (por software o firmware) de modo tal que el tiempo de respuesta del sistema no dependa de la velocidad del dispositivo más lento, esto es, que los dispositivos más rápidos no estén forzados a esperar ni a operar en sincronización con los más lentos.

Para satisfacer los requerimientos de performance establecidos y optimizar la capacidad de los dispositivos de hardware involucrados, se introduce en el diseño y en la implementación de los sistemas de tiempo real el concepto de operación concurrente.

1.3.2 Aspectos del diseño y de la implementación.

Deben considerarse las interfaces con los dispositivos de hardware y sus diferentes velocidades. Esto implica la construcción de manejadores de dispositivos ("device drivers") y de manejadores de interrupciones ("interrupt handlers").

La funcionalidad del sistema de tiempo real puede descomponerse en un conjunto de procesos secuenciales que se comuniquen, que serán implementados como módulos de software sobre una arquitectura de hardware específica. En general, los manejadores se diseñan como parte de dicho conjunto.

En consecuencia, el diseño de un sistema de tiempo real consistirá en un conjunto de procesos que operan simultaneamente en forma asincrónica (completamente independiente un proceso de otro) o bien en forma acoplada a través del acceso a datos compartidos y/o del pasaje de mensajes (datos o control).

Estos procesos lógicos deben ser traducidos a componentes de software que serán asignadas a uno o más procesadores. Por lo tanto, un sistema de tiempo real puede ser implementado como un sistema de monoprocesamiento o como un sistema de procesamiento distribuido.

En un sistema de monoprocesamiento, todas las funciones de la aplicación son ejecutadas en el mismo procesador y los elementos concurrentes compiten por el uso del mismo. Se dice que la concurrencia es aparente.

En un sistema de procesamiento distribuido, porciones de la aplicación están dispersadas en dos o más procesadores que tienen memoria local y/o compartida. Si estos procesadores no comparten memoria se requiere un mecanismo de comunicación eficiente. Se dice que la concurrencia es real.

Se procurará construir sistemas de tiempo real con un alto grado de correspondencia y simplicidad. La correspondencia (entre la especificación del problema y su solución) asegura que la solución se ajusta a los requerimientos, y ayuda en la determinación de la correctitud del programa. La simplicidad contribuye en hacer la solución más fácil de comprender y de mantener, e incrementa la confiabilidad.

1.3.3 Complejidad.

La complejidad de los sistemas de tiempo real se pone de manifiesto durante su diseño e implementación al tratar:

1.- La reducción de la complejidad por medio de la transformación de los requerimientos (descomposición funcional) en unidades (subfunciones) de fácil tratamiento.

2.- La abstracción de procesos: identificación del conjunto apropiado de procesos concurrentes que representa al sistema.

3.- El control de los dispositivos de hardware.

4.- La selección de la arquitectura de hardware apropiada que pueda soportar adecuada mente el diseño de software.

5.- La asignación de procesos concurrentes a procesadores.

6.- El manejo de la comunicación y la sincronización entre procesos.

7.- El procesamiento de mensajes.

8.- El manejo de colas y espacios para almacenar mensajes y datos ("buffers").

9.- La protección de datos compartidos.

10.- La selección de un proceso concurrente, entre aquellos que comparten un mismo procesador, para restablecer su contexto y cederle el control del procesador ("scheduling" y "dispatching").

11.- La interface con relojes de tiempo real.

12.- El manejo de los requerimientos de tiempo críticos.

13.- La detección de condiciones de falla.

14.- El diseño de simuladores en software de los dispositivos de hardware no disponibles durante el testeo.

15.- El testeo de procesos concurrentes, y la detección y corrección de errores de programa ("debugging").

Adicionalmente, un sistema de tiempo real debe ser diseñado e implementado teniendo en cuenta ciertos criterios asociados al modo en que se espera que opere.

1.3.4 Modo de operación.

Un sistema de tiempo real debe correr continuamente, en forma automática y con una confiabilidad extremadamente alta (con facilidades para "debugging"). Además, debe procesar rápidamente para satisfacer los requerimientos de performance.

La operación continua implica que estos sistemas estén disponibles y sean confiables durante largos períodos de tiempo con características de performance aceptables.

La disponibilidad puede ser definida como la probabilidad de que un sistema esté operando en un determinado instante de tiempo. La confiabilidad, como la probabilidad de que opere correctamente durante un período de tiempo especificado.

La performance de un sistema de tiempo real depende de:

- un alto grado de correspondencia entre el diseño de software y la especificación de la aplicación,
- una arquitectura de hardware apropiada para soportar el diseño de software, y
- a distribución de las componentes de software entre los procesadores del sistema.

Para satisfacer las características de performance de estos sistemas es necesario incluir un cierto grado de tolerancia a fallas.

La tolerancia a fallas puede ser definida como la habilidad de un sistema para continuar operando ante la presencia de fallas de hardware y errores de software, y aun así satisfacer, al menos en parte, los requerimientos.

Un sistema tolerante a fallas incluye mecanismos para:

- detectar las fallas de hardware y los errores de programa, y
- corregir los problemas empleando procedimientos de recuperación:

  1.- Reconfiguración física de la arquitectura de hardware: reemplazo o eliminación de las componentes de hardware falladas.

  2.- Reconfiguración lógica: redistribución de los procesos en los procesadores disponibles.

Hoy en día, la posibilidad de contar con procesadores de bajo costo hace particularmente viable la construcción de sistemas de tiempo real distribuidos.

1.4 Sistemas de tiempo real distribuido.

Las arquitecturas de procesamiento paralelo (supercomputadoras y sistemas distribuidos) son usadas actualmente para solucionar problemas de gran complejidad. Dichos problemas son divididos en tareas más pequeñas que son distribuidas entre los procesadores (por el sistema operativo o por el diseñador/programador). La velocidad de cómputo se ve incrementada, siempre que el mecanismo de comunicación sea eficiente.

En particular, los sistemas de múltiples microprocesadores interconectados son ideales para soportar las funciones de la mayoría de los sistemas de tiempo real, a causa de su relativamente bajo costo y su gran versatilidad.
Uno de los mecanismos más comunes para interconectar microprocesadores en un sistema de tiempo real es el uso de enlaces físicos o cables ("buses").

Se obtienen mejoras en la performance expandiendo la arquitectura con microprocesadores adicionales, usando microprocesadores más nuevos, o mejorando la eficiencia de los enlaces de comunicación.

Sin embargo, en algunos sistemas de tiempo real distribuidos es necesario dedicar una supercomputadora para soportar funciones de cómputo intensivo (procesamiento de señales digitales o procesamiento de imágenes).

Microprocesadores cada vez más poderosos, versátiles y baratos están siendo empleados en el control de complejas aplicaciones de tiempo real. Los sistemas de múltiples microprocesadores son usados más efectivamente en aplicaciones donde se pueden identificar (para procesamiento concurrente) areas funcionales con requerimientos de comunicación mínimos. Estas areas serán transformadas en nodos virtuales (elementos concurrentes de software compuestos por: uno o más procesos y software de soporte asociado) que son asignados a nodos físicos (microprocesadores o microcomputadoras conectados a través de un bus).

Los sistemas de múltiples microprocesadores son usados como:

- sistemas de adquisición de datos
- sistemas de control automático de procesos
- sistemas de control de robots
- sistemas de procesamiento intensivo de entrada y salida
- sistemas altamente confiables

puesto que ofrecen bajo costo, flexibilidad para programar, y una relativamente fácil implementación de los requerimientos de tolerancia a fallas (agregando microprocesadores redundantes).

Es necesario disponer de un adecuado soporte lógico (software) que permita desarrollar y ejecutar los sistemas de tiempo real hasta aquí caracterizados.

1.5 Soporte lógico de sistemas de tiempo real.

Como hemos visto, un sistema de tiempo real consiste en un conjunto de procesos concurrentes. En líneas generales, el soporte lógico para el desarrollo y la ejecución de sistemas de tiempo real debe proporcionar herramientas para especificar dichos procesos, para asignarlos a los procesadores, y para sincronizarlos y comunicarlos eficientemente.

Los requerimientos básicos a ser satisfechos por el soporte de sistemas de tiempo real son:

1.- Capacidad para especificar:

- módulos de software y sus interfaces,
- nodos virtuales (que consisten usualmente en procesos fuertemente acoplados), y
- nodos físicos (procesadores).

2.- Capacidad para asignar nodos virtuales a nodos físicos.

3.- Capacidad para generar y enlazar código, y para "debugging" distribuido.

4.- Capacidad para realizar:

- priorización de procesos,
- manejo de interrupciones,
- "scheduling" y "dispatching" de procesos,
- sincronización y comunicación de procesos (inter o intraprocesadores),
- manejo de excepciones, y
- acceso a relojes de tiempo real.

Estos requerimientos podrían ser satisfechos usando un lenguaje adecuado de programación de alto nivel, siendo el soporte para la ejecución proporcionado por el mismo lenguaje o por un ejecutivo (núcleo de tiempo real o "kernel") [NIE90].

Particularmente serán estudiadas aquellas herramientas que permitan la especificación de los requerimientos de los sistemas de tiempo real.

1.6 Especificación.

Las características de los sistemas de tiempo real obligan a considerar especialmente la primera etapa del desarrollo de un sistema: el análisis y la descripción de los requerimientos.

La complejidad estructural y funcional creciente de los sistemas de tiempo real, destacada en los párrafos anteriores, obligó a transformar lo que inicialmente era una descripción informal en lenguaje natural, en una expresión rigurosa: la especificación de requerimientos del sistema.

La especificación de requerimientos del sistema es la expresión en un lenguaje formal, o mediante alguna herramienta gráfica, de las funciones que el sistema debe cumplir y de las restricciones de contexto.

Los objetivos de la especificación son:

- proporcionar mayor claridad y documentación, y
- posibilitar la verificación y/o validación parcial o total (permitiendo comprobar que el sistema desarrollado satisface los requerimientos especificados).

Todo ello contribuye a una mayor confiabilidad y a un menor costo global del sistema (facilitando el mantenimiento al proporcionar información y conocimiento integral de la solución) [DEG91].

Los capítulos siguientes tratan el proceso y la metodología de especificación de requerimientos, y se orientan a la caracterización y el desarrollo de un primer prototipo de ambiente de especificación de requerimientos de sistemas de tiempo real.

1.7 Referencias bibliográficas.

[DEG91] De Giusti, Armando E., "Descripción y Validación de Hardware. Aplicaciones de Tiempo Real", Rio de Janeiro, V EBAI, 1991.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[MAG86] Magalhäes, Maurício Ferreira, "Software para tempo real", Campinas, UNICAMP, 1986.

[NIE90] Nielsen, Kjell W., "Ada in Distributed Real-Time Systems", New York, McGraw-Hill, 1990.

[PET84] Peterson, James Lyle y Silverschatz, Abraham, "Operating System Concepts", Massachusetts, Addison- Wesley, 1984.

Capítulo 2: Alcance del proyecto.

El proceso de desarrollo de sistemas comprende un conjunto de fases que se deben completar para desarrollar un sistema. Se emplean distintas metodologías (herramientas y técnicas) para completar cada una de esas fases y representar sus productos (especificar). Dichos productos constituyen la especificación del sistema [HAT88, SHU92]. Este proyecto se circunscribe a la primera etapa en la especificación de un sistema de tiempo real, que es llevada a cabo durante la fase de análisis de requerimientos del sistema: la especificación de requerimientos del sistema.

2.1 Proceso de desarrollo de sistemas de tiempo real.

En el capítulo anterior se presentó una caracterización de los sistemas de tiempo real fuertemente influenciada por la metodología de desarrollo de sistemas de tiempo real distribuidos presentada en [NIE90].

Es posible descubrir ciertas limitaciones en dicha metodología. En primer lugar, el desarrollo del hardware es omitido pues la funcionalidad del sistema es implementada únicamente en software y las componentes de la arquitectura son microprocesadores sobre los que el software es ejecutado. Por último, el desarrollo del software es orientado al uso de Ada como lenguaje de implementación; en consecuencia, el empleo de la metodología de especificación está limitado, en cierta medida, a aquellos que se encuentran familiarizados con proyectos de diseño e implementación en Ada.

En razón de estas limitaciones resulta conveniente adoptar un enfoque más general según el cual la funcionalidad de un sistema de tiempo real pueda ser implementada totalmente en hardware, totalmente en software, o parte en hardware y parte en software; en cuyo caso, la construcción del sistema involucra el desarrollo concurrente de hardware y software, y su posterior integración.

El proceso de desarollo de sistemas de tiempo real entonces considerado (figura 2.1 [DEG91, SHU92]) incluye las siguientes fases:

- el análisis de requerimientos del sistema,
- el diseño del sistema,
- el análisis de requerimientos del hardware y del software, y
- el diseño preliminar y detallado del hardware y del software

que son llevadas a cabo en forma interactiva e iterativa [HAT88, SHU92].

2.2 Primera etapa.

Centraremos nuestra atención en la primera fase del proceso de desarrollo: el análisis de requerimientos del sistema (para determinar y especificar qué hará el sistema), presentando métodos para identificar y documentar cuáles son los requerimientos del sistema.

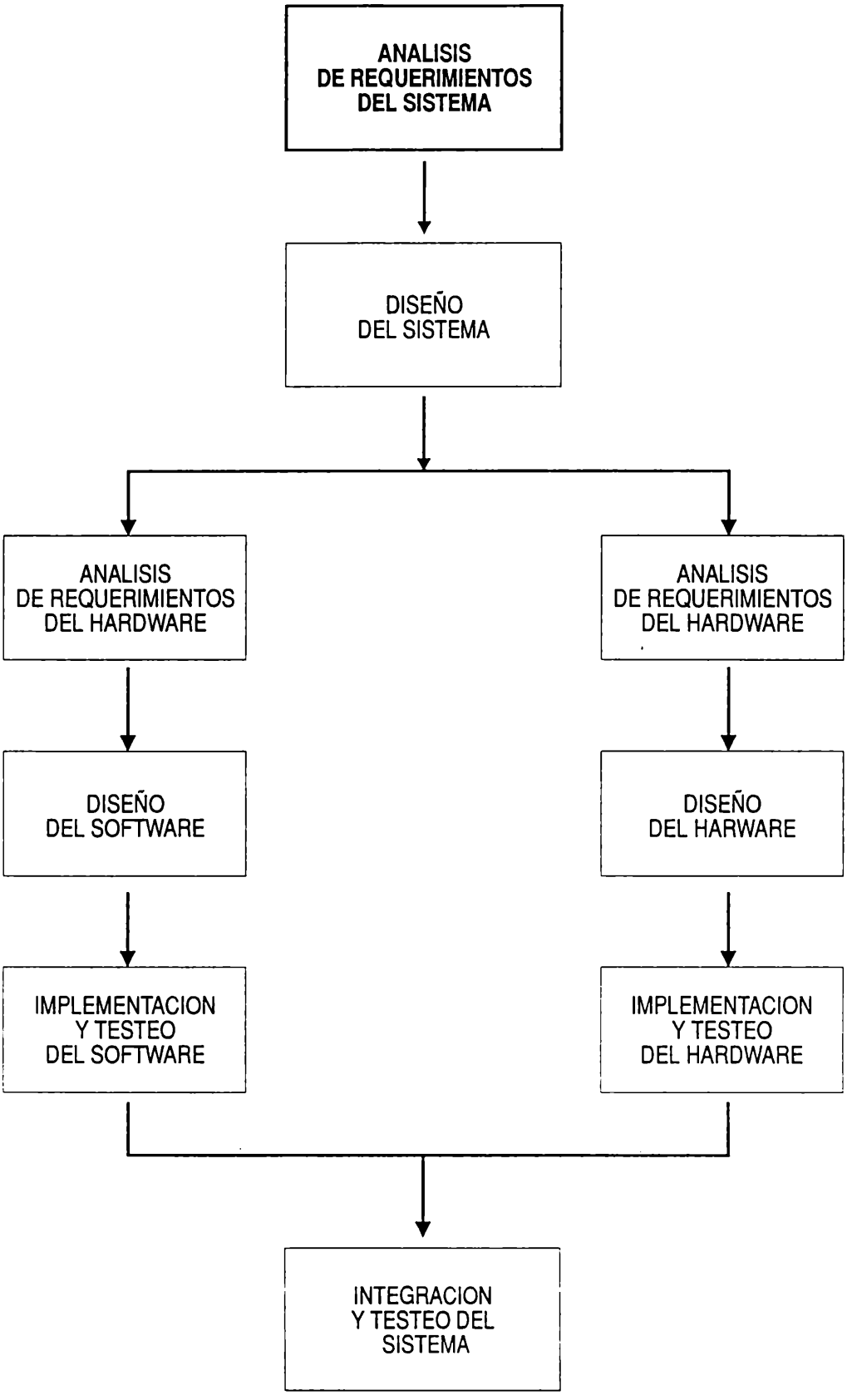


Figura 2.1 Proceso de desarrollo de sistemas de tiempo real

La especificación de requerimientos del sistema juega un rol muy importante en el desarrollo del mismo puesto que de ella se derivan el diseño del sistema y los requerimientos de hardware y de software. Posibilita el testeo de la correspondencia existente entre el sistema construido y los requerimientos especificados. Además, disponer de una documentación clara facilita la comprensión y el mantenimiento del sistema, disminuyendo su costo.

La metodología de análisis de requerimientos a la que haremos referencia es presentada en [SHU92] como Análisis Estruturado de Tiempo Real (AETR). El AETR comprende métodos tradicionales de análisis estructurado [DEM78] a los que se incorporan elementos para tratar las restricciones de tiempo y el control [HAT88].

El estudio de las herramientas y metodologías para especificar requerimientos servirá de base para desarrollar el soporte lógico adecuado que permita representar las componentes relevantes de la especificación de requerimientos.

A continuación, se presentará en términos generales qué tarea se lleva a cabo durante la fase de análisis de requerimientos, cuál es la documentación resultante, y qué subconjunto mínimo de dicha documentación debe permitir representar el soporte lógico a ser desarrollado (el ambiente de especificación de requerimientos).

2.3 Análisis de requerimientos del sistema.

Durante esta fase se definen los requerimientos del sistema, aquello que el sistema debe hacer para satisfacer las necesidades del cliente (requerimientos operacionales).

Documentación.

El producto (o documentación) resultante de esta fase es la especificación de requerimientos del sistema (modelo de requerimientos) que incluye:

- Diagramas de contexto de datos (DCDs): Muestran las interfaces entre el sistema que se está especificando y el mundo exterior.
- Diagramas de flujo de datos (DFDs): Muestran los requerimientos del sistema en términos de las actividades (funciones o procesos) que debe realizar el mismo, primero en general y después en detalle usando descomposición jerárquica. También muestran los flujos y los bancos de datos.
- Especificaciones de proceso (EPs): Definen el procesamiento realizado por los procesos de más bajo nivel de los DFDs (procesos primitivos).

Las componentes de la especificación hasta aquí mencionadas permiten expresar "qué" actividades ocurren en el sistema. Resulta igualmente importante expresar "cuándo" ocurren esas actividades, para lo cual se hace uso de las siguientes componentes:

- **Diagramas de contexto de control (DCCs).**
- **Diagramas de flujo de control (DFCs).**
- **Especificaciones de control (ECs).**

La especificación de requerimientos del sistema incluye además:

- **Diccionario de requerimientos (DR):** Define los elementos de los DFDs y DFCs.
- **Especificación de tiempos de respuesta (ETR):** Define los tiempos de respuesta del sistema a los estímulos del mundo exterior.

Soporte lógico para especificar requerimientos.

El soporte lógico a desarrollar debe permitir representar al menos:

- el ambiente en el cual el sistema debe operar,
- la funcionalidad del sistema (aunque más no sea en un primer nivel de abstracción) en términos de funciones y flujos de información entre funciones,
- la lógica que explique cuando se activan y desactivan dichas funciones, y
- el tiempo que el sistema debe tardar en responder a los estímulos del ambiente en el que opera.

En otras palabras, se deben poder representar el DCD, los DFDs (incluyendo flujos de control), las ECs y la ETR.

En los capítulos que siguen se describen las componentes de la especificación de requerimientos (capítulo 3), y se muestra de que manera llevar a cabo la representación de dicha especificación (capítulo 4), sentando de este modo las bases para la caracterización y construcción de un ambiente de especificación de requerimientos.

2.4 Referencias bibliográficas.

[DEG91] De Giusti, Armando E., "Descripción y Validación de Hardware. Aplicaciones de Tiempo Real", Rio de Janeiro, V EBAI, 1991.

[DEM78] DeMarco, Tom, "Structured Analysis and System Specification", New York, Yourdon, 1978.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[NIE90] Nielsen, Kjell W., "Ada in Distributed Real-Time Systems", New York, McGraw-Hill, 1990.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.

Capítulo 3: Especificación de requerimientos del sistema.

Los requerimientos del sistema se determinan y representan usando herramientas adecuadas durante la fase de análisis de requerimientos del sistema.

3.1 Fase de análisis de requerimientos del sistema.

En términos generales, durante esta fase se desarrollan dos actividades (figura 3.1):

- el análisis del problema, y
- la descripción del producto.

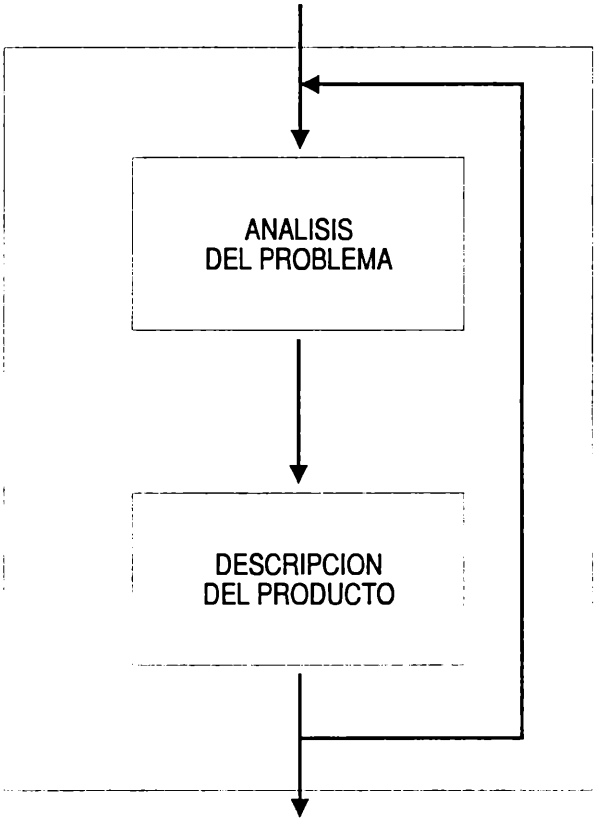


Figura 3.1 Fase de análisis de requerimientos del sistema.

El análisis del problema involucra:

- el establecimiento de los objetivos del sistema,
- la identificación de las funciones y sus interfaces,
- a especificación de la performance de cada función, y
- la descomposición en subfunciones de cada función que por su complejidad requiera un análisis más detallado.

La descripción del producto involucra la preparación de un documento (la especificación de requerimientos del sistema) que describe el comportamiento externo esperado del sistema. Dicho sistema será construido para resolver un problema que, recién después del análisis, es cabalmente comprendido. Las componentes del documento serán descriptas detalladamente más adelante.

Ambas actividades se desarrollan iterativamente, esto es, se realiza un poco de análisis, un poco de descripción; a continuación un poco más de análisis, un poco más de descripción, y así sucesivamente hasta obtener el producto.

Un ambiente para especificar sistemas de tiempo real debe acompañar el proceso de desarrollo de los mismos. En lo que concierne a la fase de especificación de requerimientos, eso significa que el ambiente debe permitir la descripción del producto parcial de cada iteración y no ser usado simplemente para "pasar en limpio" el producto final de la fase. En consecuencia, debe facilitar la modificación (ampliación o corrección) de las componentes de la especificación desarrolladas a lo largo de iteraciones previas y, al mismo tiempo, mantener la consistencia entre las mismas. Se dice que las componentes de la especificación son consistentes entre sí cuando representan al mismo sistema.

Suposiciones.

Para facilitar la tarea de análisis resulta conveniente suponer que los requerimientos son "esenciales" e "independientes de la tecnología".

Los objetivos del sistema son descriptos inicialmente, casi siempre de manera informal, como resultado de un primer contacto con el cliente. Dichos objetivos son considerados "esenciales", significando con ello que reflejan fielmente las únicas necesidades que deben ser satisfechas.

Además, los requerimientos son considerados "independientes de la tecnología" que será empleada para implementarlos. Los aspectos tecnológicos deberán ser tratados durante la fase de diseño. Si la tecnología entonces disponible no puede satisfacer alguno de los requerimientos, será necesario modificarlos. Esta retroalimentación ("feedback") en el proceso de desarrollo refleja la interacción existente entre el análisis y el diseño.

Ambas suposiciones permiten clarificar los aspectos relevantes a ser tratados durante al análisis simplificando dicha tarea.

3.2 Componentes de la especificación de requerimientos del sistema.

Las diferentes herramientas (componentes gráficas, textuales y tabulares) usadas para representar los requerimientos del sistema se detallan a continuación.

3.2.1 Diagrama de contexto de datos (DCD).

El DCD identifica las entidades externas con las que se debe comunicar el sistema, y establece el propósito principal del mismo representándolo como un simple proceso con nombre. Muestra los datos que fluyen entre el sistema que está siendo especificado y las entidades externas con las que éste se debe comunicar.

Componentes.

Un DCD (figura 3.2) consiste en:

- un proceso,
- terminadores, y
- flujos de datos.

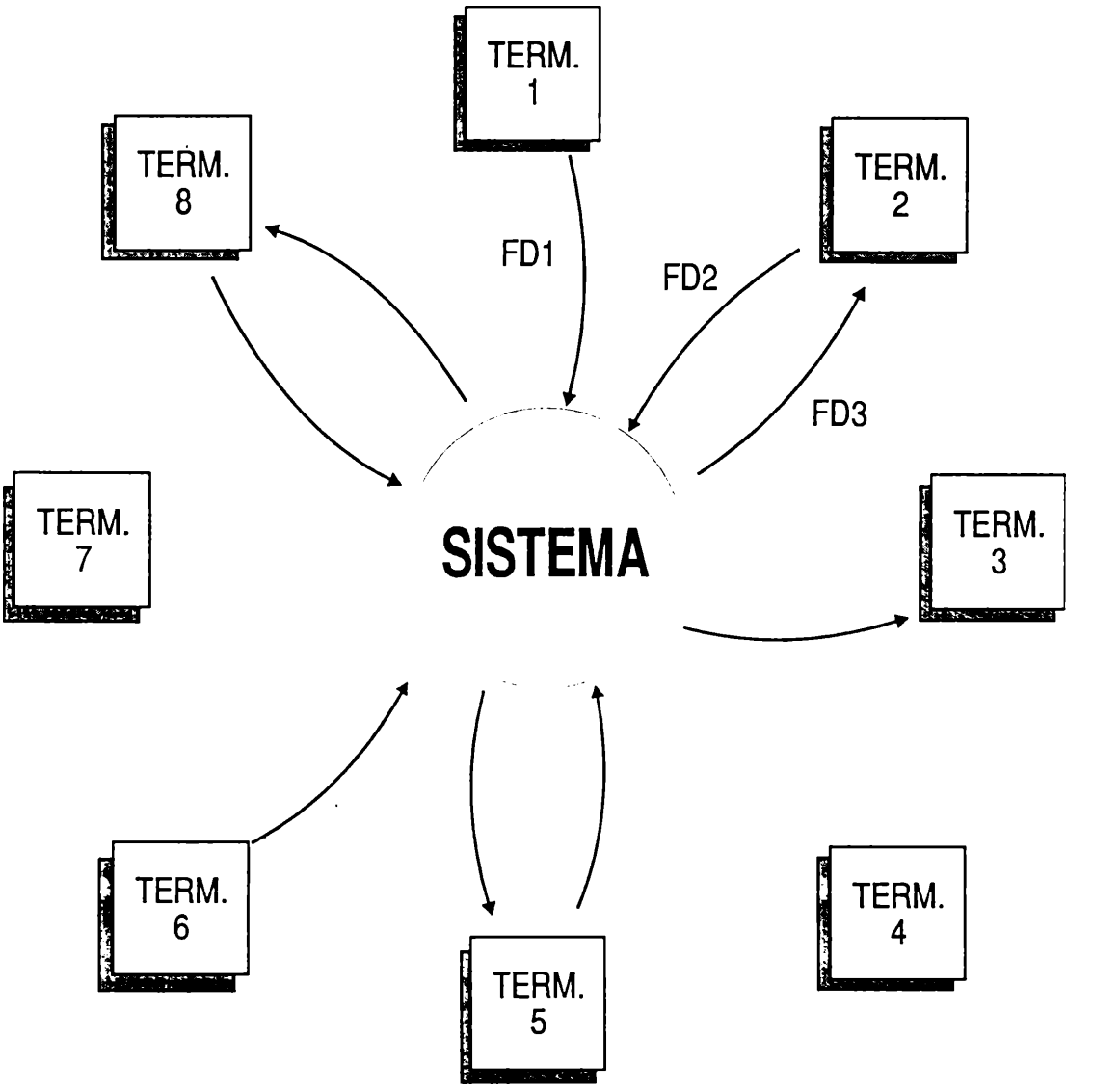


Figura 3.2 Un diagrama de contexto de datos

El proceso representa al sistema. Es mostrado como un círculo o burbuja. El nombre del proceso representa la función pura del sistema sin relacionarla con la eventual forma física del mismo, esto es, como se dijo anteriormente, ignorando los aspectos tecnológicos.

Los terminadores representan a las entidades externas con las cuales el sistema se debe comunicar. Son mostrados como rectángulos. No forman parte del sistema.

Los flujos de datos son los enlaces de comunicación entre el proceso y los terminadores. Son mostrados como arcos dirigidos. Estos arcos muestran el nombre de los datos y la dirección en la estos fluyen. Dichos datos aparecerán eventualmente en las terminales de entrada y salida del sistema implementado.

3.2.2 Diagramas de flujo de datos (DFDs).

Los DFDs son construidos, y están organizados, por niveles. Cada nivel de DFDs muestra más detalles de los requerimientos que el nivel precedente.

Un DFD (figura 3.3) contiene:

- procesos (círculos),
- flujos de datos (arcos dirigidos), y
- bancos de datos (líneas paralelas).

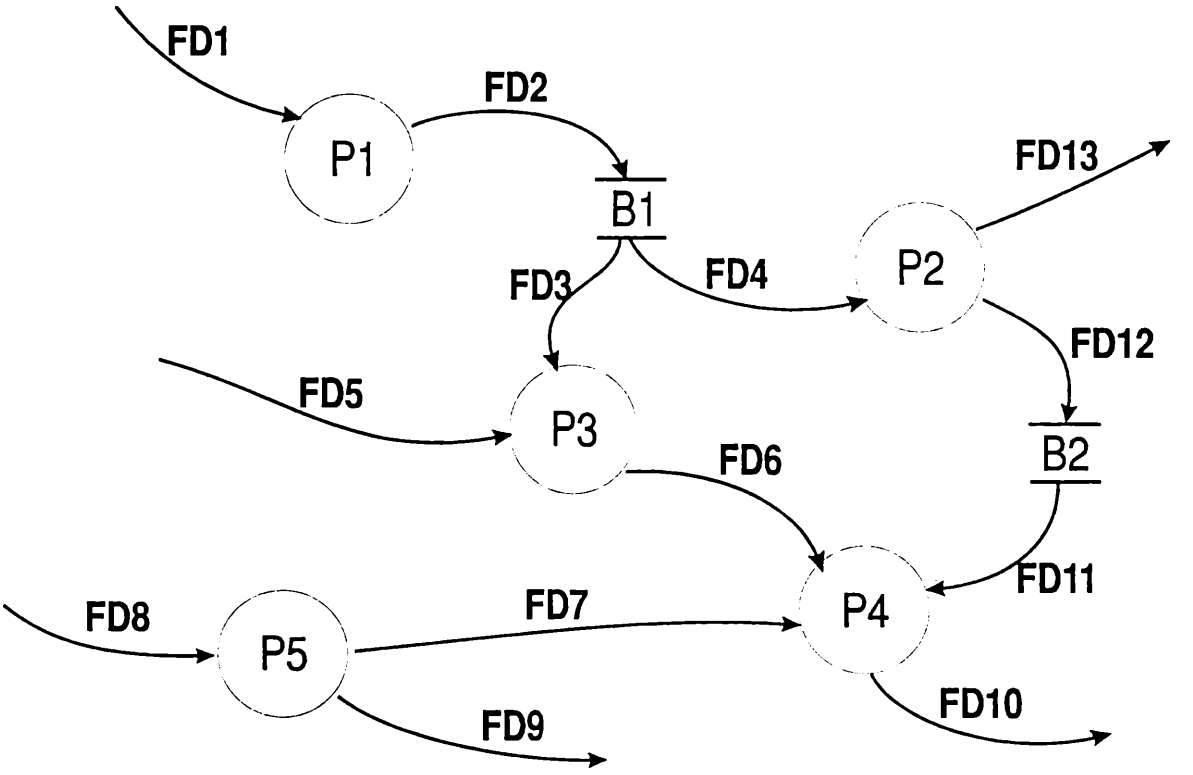
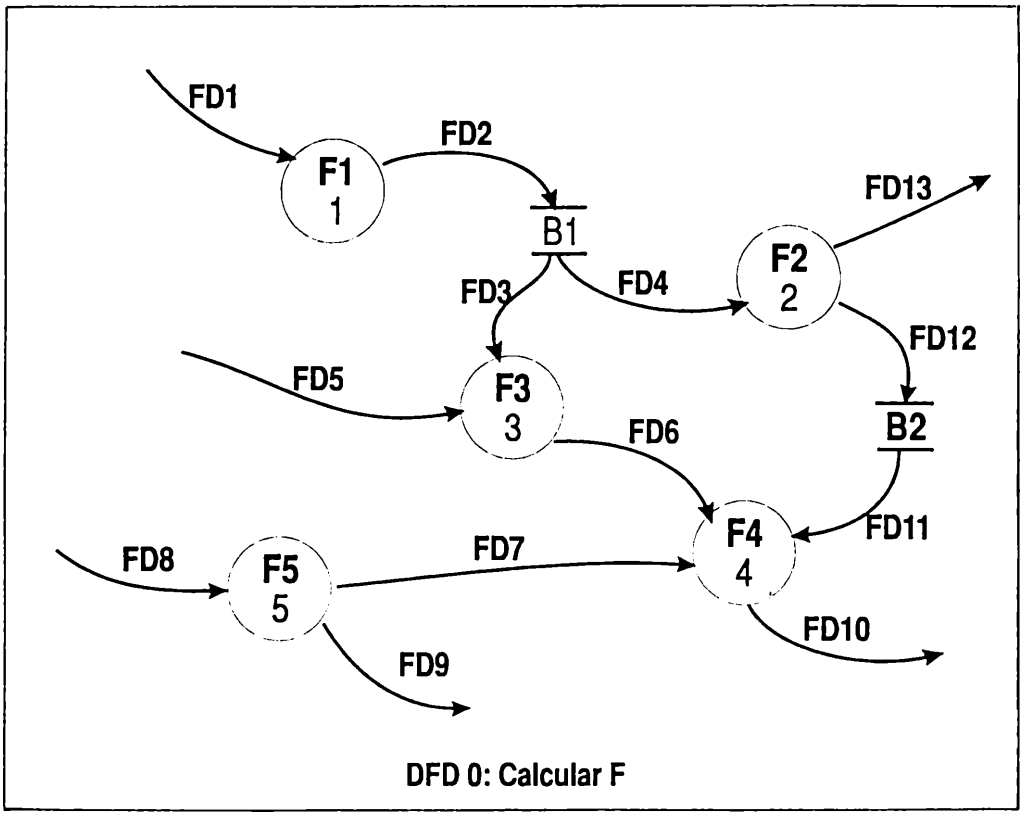


Figura 3.3 Undiagrama de flujo de datos.

Todo DFD es la descomposición de un proceso, llamado proceso padre, mostrado en un DFD del nivel precedente (figura 3.4). Un DFD hereda de su proceso padre el número y el nombre como título. El proceso es entonces descompuesto en subprocesos y sus flujos de entrada y salida en subflujos. Además se introducen nuevos flujos entre los subprocesos. Cada subproceso hereda del proceso padre el número al que se agrega un número propio. Estos subprocesos y subflujos sólo más detalles acerca del sistema.



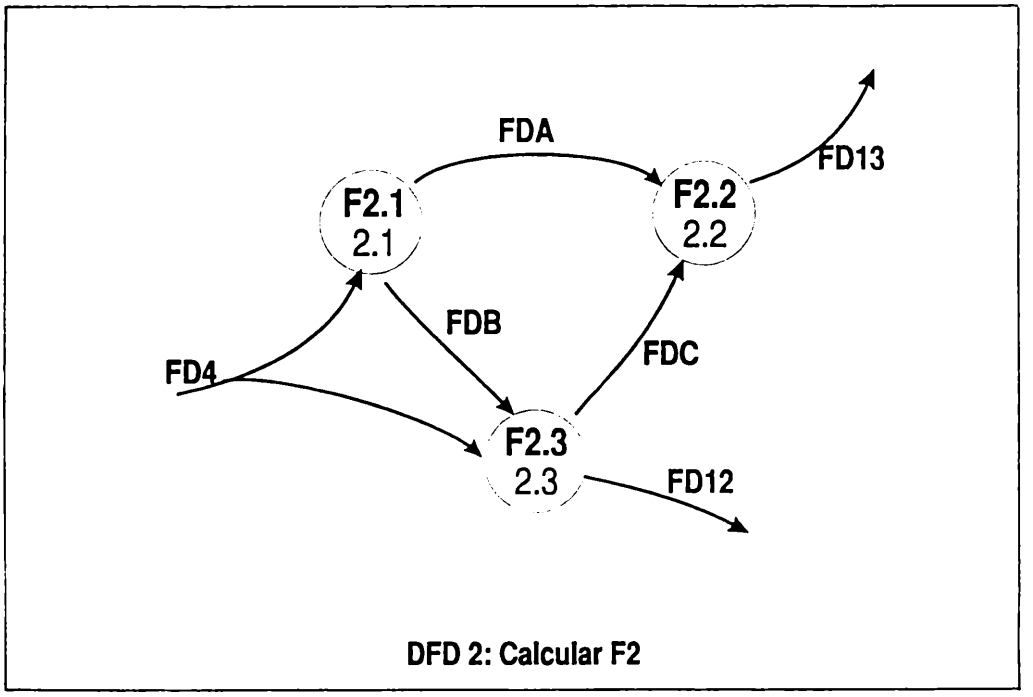


Figura 3.4 DFDs de dos niveles.

Procesos.

Un proceso representa una acción, produce información de salida a partir de información de entrada. En consecuencia, el nombre de un proceso debería comenzar con un verbo que describa la función del proceso. Por otro lado, los flujos y bancos de datos representan información, y sus nombres no deberían contener verbos. En relación con esto, se aconseja evitar el uso de abreviaturas, o al menos usar abreviaturas comprensibles y en forma congruente a lo largo de toda la especificación, esto es, que un nombre tenga a lo sumo una abreviatura.

Chequeo de consistencia padre-hijo:

Un DFD y su proceso padre representan la misma información en diferentes niveles de detalle. En consecuencia, sus flujos de entrada y salida deben ser idénticos. La verificación de tal propiedad es llamada "balancing". Un ambiente de especificación de sistemas de tiempo real debería realizar dicha verificación.

Flujos de datos.

Los flujos de datos representan flujos de elementos simples o flujos de grupos de elementos. Un flujo de grupo se descompone en flujos de subgrupos y/o en flujos de elementos simples. Estos subflujos se dirán hijos del flujo de grupo del cual son componentes.

Los flujos pueden ser descompuestos y combinados en un diagrama, y cuando pasan de un proceso a su diagrama hijo (y viceversa). Los flujos se agrupan para condensar la información y clarificar el diagrama.

Todos los flujos de datos son definidos en el diccionario.

Chequeo de consistencia de flujos de datos:

Todo flujo de datos debe ser definido en el diccionario, y las definiciones de grupos y subgrupos deben ser consistentes entre sí. La verificación de tales propiedades es llamada "cross-checking". Un ambiente de especificación de sistemas de tiempo real debería realizar dicha verificación.

Bancos de datos.

Los bancos de datos representan datos almacenados para ser usados (accedidos) posteriormente o más de una vez por un proceso.

Los nombres de los bancos de datos identifican a los datos que contienen. Los flujos de datos que entran a un banco de datos o salen de él deben ser nombrados sólo cuando llevan una parte de los datos del banco.

Los bancos de datos son definidos en el diccionario.

Principio de no redundancia:

Un banco de datos debe aparecer en sólo un DFD. En caso de que dicho diagrama sea complejo el banco puede aparecer en él más de una vez.

3.2.3 Especificación de proceso (EP).

La descomposición de un proceso continúa hasta que se alcanza un nivel en el cual el proceso, llamado proceso primitivo, puede ser descripto concisa y brevemente en una especificación de proceso.

Una EP (figura 3.5) es hija del proceso que describe. Comparte las mismas entradas y salidas, y hereda el número y el nombre.

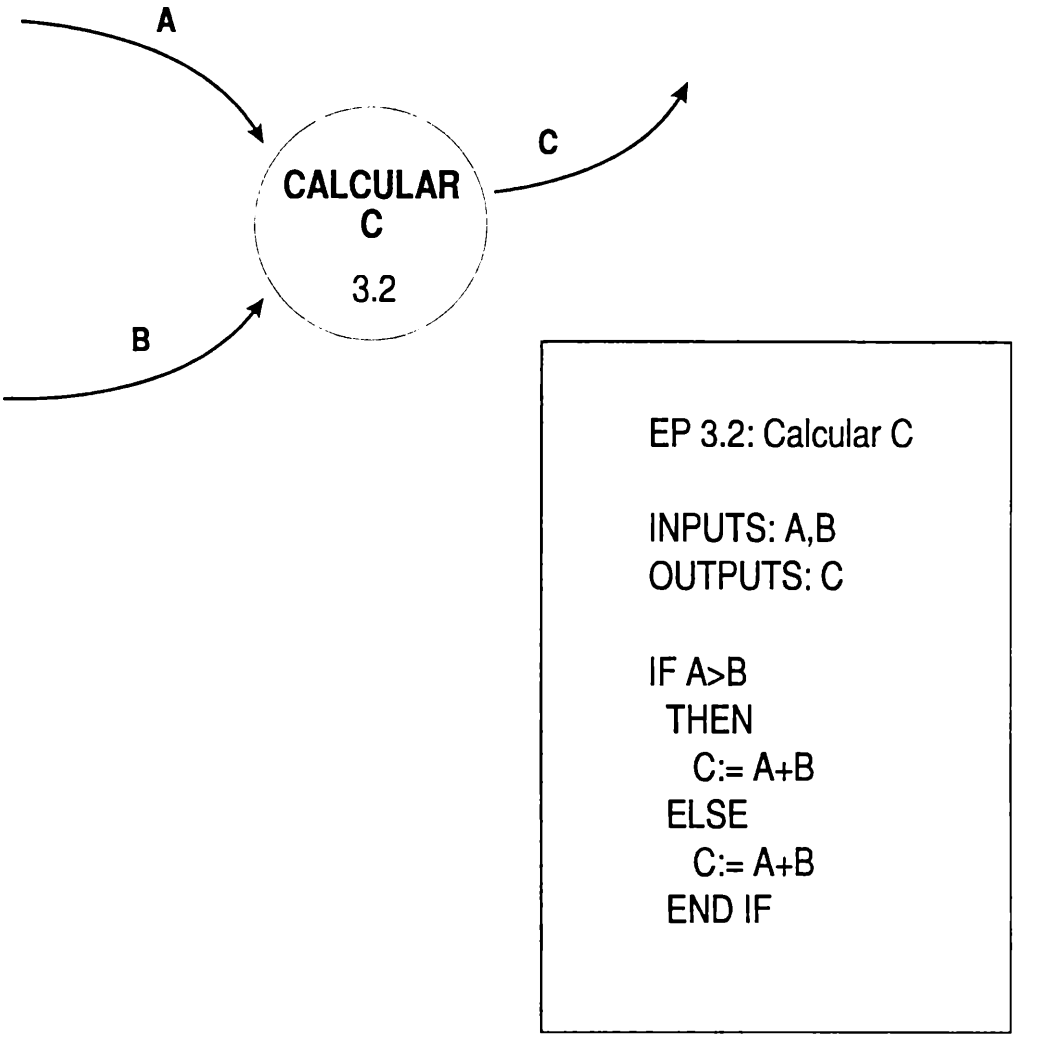


Figura 3.5 Una especificación de proceso.

El propósito de la EP es mostrar cómo sus salidas son generadas a partir de sus entradas. Para ello usa un lenguaje estructurado que incluye ciertos tipos de palabras:

- ❑ verbos infinitivos,
- ❑ flujos de datos (en letras mayúsculas),
- ❑ constantes físicas y matemáticas conocidas,

y estructuras gramaticales simples: los constructores para especificar concurrencia, secuencia, decisión y repetición de actividades (indentados para mostrar subordinación).

La simplicidad y brevedad asegura que las EPs no sean ambiguas. También se usan ecuaciones, tablas, diagramas y grafos. Comentarios informales son incluidos para facilitar la comprensión de las EPs.

3.2.4 Diagrama de contexto de control (DCC).

El DCC muestra la información de control que fluye entre sistema y su ambiente.

Es idéntico al DCD exceptuando que contiene flujos de control, mostrados como arcos dirigidos segmentados, en lugar de flujos de datos (figura 3.6). Incluye el proceso, representando al sistema, y los terminadores, representando a las entidades externas, que son mostrados en el DCD.

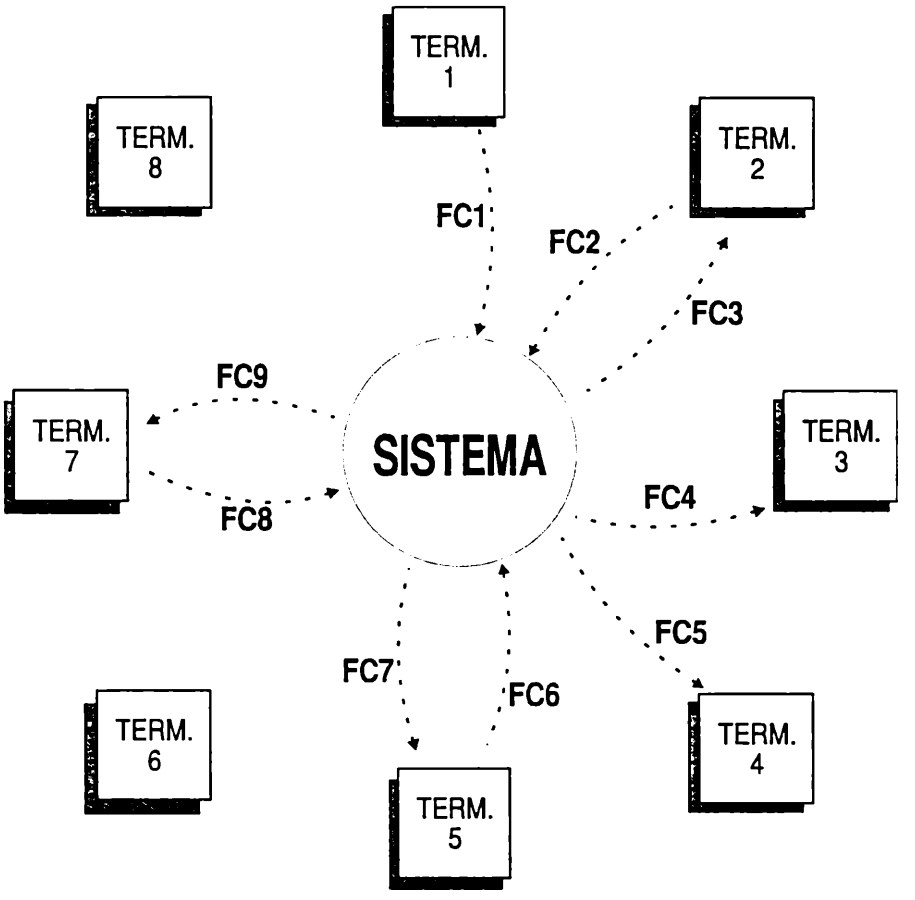


Figura 3.6 Undiagrama de contexto de control.

Ambos diagramas de contexto podrían ser superpuestos y dibujados como uno si no son demasiado complejos [SHU92], aunque resulta ventajoso mantener separadas las estructuras de datos y de control [HAT88].

3.2.5 Diagramas de flujo de control (DFCs).

Los DFCs muestran el flujo del control a través del sistema. Para un dado DFD se dibuja su correspondiente DFC si es necesario el control en esa parte del sistema. Ambos diagramas comparten número y nombre, e incluyen los mismos procesos. Difieren sólo en que el DFD muestra flujos y bancos de datos, mientras que el DFC muestra flujos y bancos de control (figura 3.7).

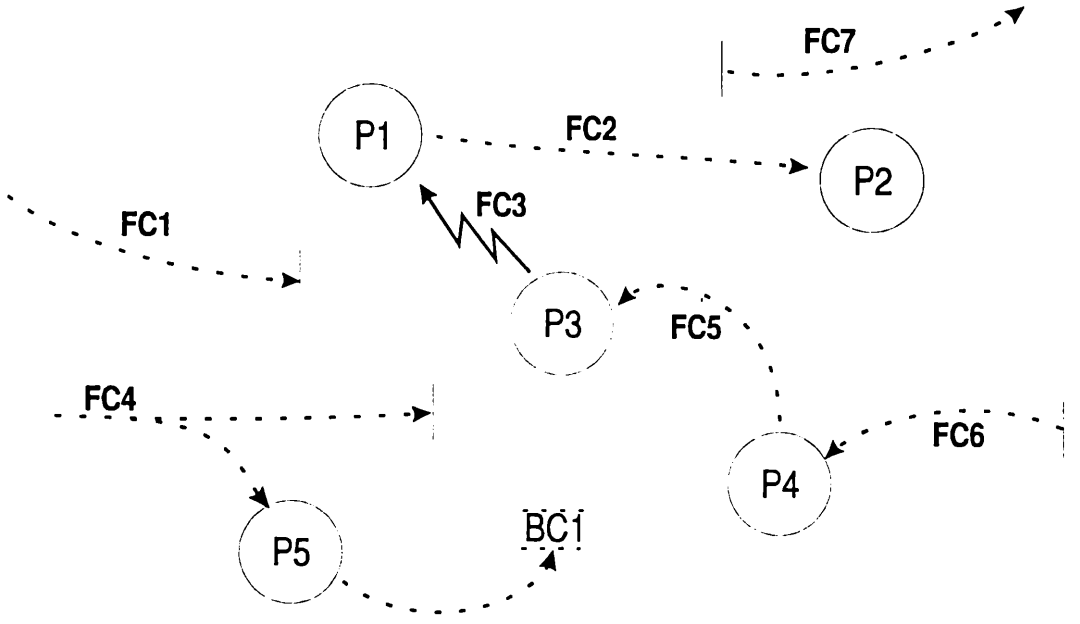


Figura 3.7 Un diagrama de flujo de control.

Un flujo de control que entra en un proceso no es procesado por este sino por una especificación de control (EC). Tampoco controla dicho proceso enteramente sino subprocesos del mismo.

Además, los DFCs incluyen barras de control (líneas verticales simples) que representan la interface entre un DFC y su EC. Varias barras de control pueden aparecer en un DFC pero todas están asociadas con la misma EC.

Los flujos de control que ingresan en un proceso terminan en un subproceso o en una barra de control del DFC hijo de dicho proceso. Aquellos que terminan en una barra de control ingresan en la EC asociada que es empleada para controlar (activar o desactivar) subprocesos.

Los flujos de control que egresan de un proceso provienen de un subproceso o de una barra de control del DFC hijo de dicho proceso, o bien, provienen de su EP si el proceso fuera primitivo. En este último caso, los flujos de control, generados en la EP a partir de testeos sobre datos, son llamados condiciones de datos [HAT88]. Aquellos que salen de una barra de control son generados en la EC asociada, y usados para controlar procesamiento fuera del alcance del diagrama.

Un proceso es activado o desactivado indirectamente por un flujo de control a través de una EC. Alternativamente, un proceso primitivo puede ser activado o desactivado directamente por otro proceso, incluido en el mismo diagrama, a través de un evento de proceso (representado por un rayo). El proceso que es activado por un flujo de control, vía la EC, permanece continuamente activo hasta que es explícitamente desactivado. El proceso primitivo que es activado por un evento de proceso continúa procesando dependiendo de la acción especificada en su EP [SHU92].

Además de procesos activados por ECs o por eventos, hay procesos que realizan su función toda vez que los datos de entrada necesarios están presentes. Se dice que son procesos activados por datos.

Flujos de control.

Los flujos de control y de datos presentan las siguientes diferencias:

- las señales de control son siempre señales discretas, las de datos son usualmente continuas;
- las señales discretas pueden tomar un valor de un conjunto finito de valores conocidos que no necesitan tener un orden relativo particular, las continuas pueden tomar un valor de un conjunto arbitrariamente grande de valores numéricos ordenados;
- las señales discretas representan usualmente un modo o condición; las continuas, una cantidad física continua [HAT88].

Las señales de control y de datos pueden formar parte de un mismo flujo de grupo, el cual será mostrado como flujo de datos hasta que sea descompuesto en subflujos de control y de datos [SHU92].

Bancos de control.

Los bancos de control almacenan flujos de control después que las fuentes de esos flujos han cesado de producirlos, y retienen dicho contenido hasta que es sobreescrito.

Sus nombres son descriptivos de la información que contienen. Los flujos que llevan toda la información del banco no tienen nombre. Los flujos que llevan subgrupos de la información almacenada tienen por nombre el nombre del subgrupo.

Los bancos pueden ser de lectura solamente, o de lectura y escritura.

Un banco de control sólo aparece en un DFC. En caso de que dicho diagrama sea complejo, el banco puede aparecer en él más de una vez. Los bancos de control son mostrados como líneas paralelas segmentadas.

Cada DFD y su DFC asociado pueden ser combinados en un sólo diagrama, en el caso de sistemas simples, o pueden ser dibujados por separado para distinguir los aspectos de datos de los de control, y limitar la cantidad de información incluida en un diagrama.

3.2.6 Especificación de control (EC).

Las ECs representan el comportamiento del sistema, en lo que respecta al procesamiento de control, por medio de máquinas de estados finitos. Las mismas serán descriptas más adelante.

El propósito de una EC es mostrar cómo sus salidas son generadas a partir de sus entradas. Las entradas son flujos de control que provienen del DFC asociado a través de una barra de control. Las salidas son flujos de control que vuelven al DFC asociado a través de una barra de control, o son activadores de procesos que activarán o desactivarán procesos del DFD/DFC asociado.

Una EC está presente cuando se requiere controlar (activar o desactivar) procesos del DFD/ DFC, o cuando se requiere convertir algunas señales de control en nuevas señales de control.

Chequeo de consistencia entre EC y diagramas de flujo:

Cada EC hereda el número y el nombre que comparten los diagramas de flujo (DFD y DFC) a los que está asociada. Los flujos de control que son entradas o salidas de la EC, van o vienen del DFC a través de sus barras de control. Los procesos controlados están en el DFD. Un ambiente de especificación de sistemas de tiempo real debería realizar la verificación de dichas propiedades.

Como hemos dicho, las ECs contienen máquinas de estados finitos. Las entradas y salidas de las mismas son las entradas y salidas de las ECs. Las máquinas de estados finitos pueden ser máquinas combinacionales o secuenciales.

Las máquinas combinacionales no contienen memoria. Sus salidas están determinadas únicamente por por sus entradas actuales. Son representadas por tablas de decisión.

Las máquinas secuenciales contienen memoria. Sus salidas están determinadas por sus entradas actuales y pasadas. Son representadas por diagramas (tablas o matrices) de transiciones de estados en cooperación con tablas de activación de procesos.

Tabla de decisión (TD).

Una TD (figura 3.8) es una tabla que muestra en cada una de sus filas una posible

combinación de valores de señales de entrada, con los correspondientes valores de señales de salida producidos. Todas las combinaciones de valores de señales de entrada deben estar presentes en la tabla. Dependiendo de las necesidades del sistema, pueden no ser usados todos los valores de señales de salida. Cuando varias combinaciones de entrada producen la misma salida, pueden ser condensadas en una única fila.

ENTRADAS		SALIDAS	
B	C	A	D
0	0	1	0
1	0	0	1
OTRA COMBINACION		1	1

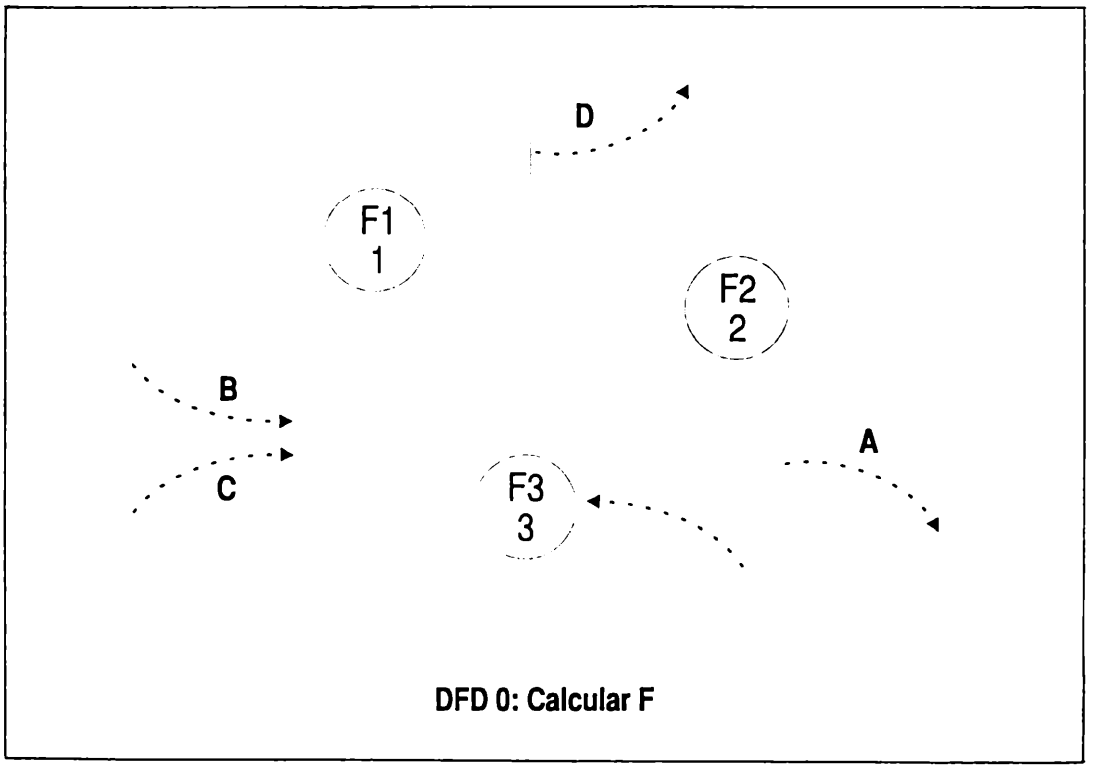


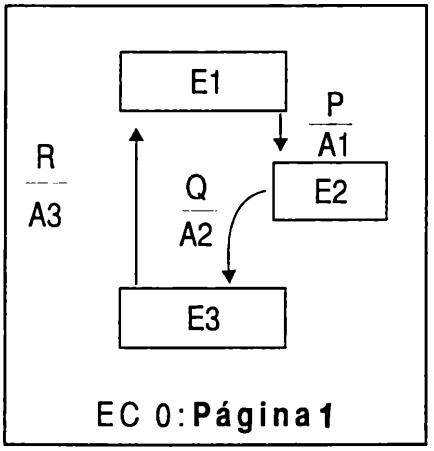
Figura 3.8 Una tabla de decisión.

Diagrama de transiciones de estados (DTE).

La memoria que contienen las máquinas secuenciales es representada en forma de estados. El estado es el modo o condición de la máquina. En un estado cualquiera, ciertas combinaciones de valores de señales de entrada (eventos) causarán que la máquina cambie su estado, o produzca salidas (acciones), o ambas cosas. Más adelante veremos cómo las acciones son convertidas en activadores de procesos y en flujos de control.

En un DTE (figura 3.9) los estados del sistema están representados por rectángulos. La transición de un estado a otro está representada por un arco dirigido que conecta los rectángulos de ambos estados. Los eventos son mostrados por sus nombres sobre los arcos de las transiciones que causan. Las acciones son mostradas por sus nombres junto a los eventos que las producen, separados de estos por una línea.

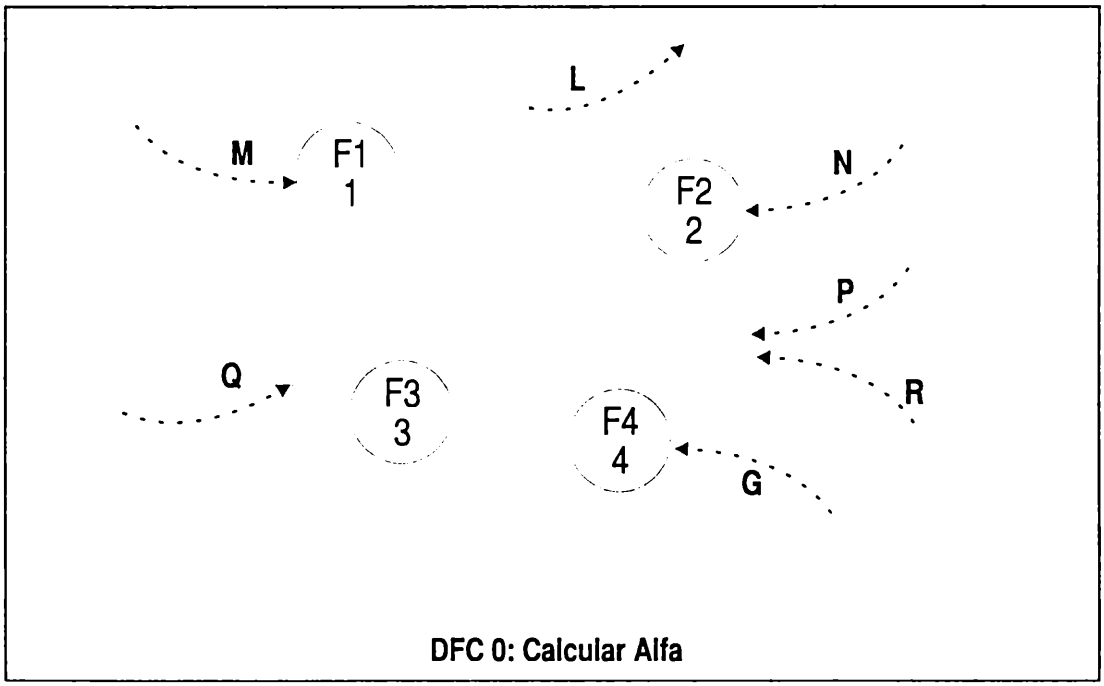
En un estado cualquiera, cuando ocurre un evento asociado a una transición desde ese estado, la máquina cambiará su estado y realizará simultáneamente la acción asociada, si la hubiere.



EC 0: **Página 1**

ACCIONES	PROCESOS				SALIDAS
	1	2	3	4	L
A1	1	0	0	1	SI
A2	1	0	1	0	_
A3	0	1	0	0	NO

EC 0: **Página 2**



DFC 0: Calcular Alfa

Figura 3.9 Una especificación de control compuesta: undiagrama de transiciones de estados (página 1) y una tabla de activación de procesos (página 2).

Tabla de transiciones de estados (TTE).

La TTE -al igual que la matriz de transiciones de estados (MTE)- representa, en un formato diferente, exactamente la misma información que el DTE. La TTE permite representar casos más complejos y grandes que el DTE, pero puede resultar más difícil comprender la operación de la máquina de estados a causa de su menor expresividad.

Una TTE (figura 3.10) consiste en cuatro columnas. La primera contiene una lista de los estados. La segunda muestra los eventos que causan transiciones desde cada uno de los estados. La tercera muestra la acción asociada con cada transición. Finalmente, la cuarta muestra el estado hacia el que se produce la transición.

ESTADO ACTUAL	EVENTO	ACCION	ESTADO SIGUIENTE
E1	P	A1	E2
E2	Q	A2	E3
E3	R	A3	E1

Figura 3.10 Una tabla de transiciones de estados.

Tabla de activación de procesos (TAP).

Una TAP (figura 3.9) es un tipo de tabla de decisión usada para controlar procesos de un DFD. Sus entradas son acciones asociadas a cambios de estados, y son listadas en la primera columna. Sus salidas son activadores de procesos y flujos de control de salida, y son listadas respectivamente en columnas encabezadas con nombres de procesos y de flujos de control.

Si al producirse un cambio de estado ocurre una acción, en la fila de la TAP asociada a esa acción se indicará:

- en su intersección con una columna encabezada con el nombre de un proceso, con un 1 o un 0, que dicho proceso es activado o desactivado, y

- en su intersección con una columna encabezada con el nombre de un flujo de control, con un valor de señal discreta, que dicho flujo sale de una barra de control con el valor dado.

Puesto que se supone que la acción asociada a una transición continúa en efecto hasta que ocurre la siguiente transición, los procesos activados por una acción particular permanecerán activados y las señales de control de salida serán producidas hasta la siguiente transición.

Como vemos DTE y TAP son usados en forma combinada constituyendo una EC compuesta.

Dado que una EC puede contener varias de las representaciones descriptas, podría llegar a extenderse a lo largo de varias páginas, siendo la única componente de la especificación de requerimientos con esta característica.

3.2.7 Especificación de tiempos de respuesta (ETR).

El analista necesita sólo especificar con exactitud y sin ambigüedad qué hace el sistema. Dicho sistema debe responder a ciertos estímulos externos dentro de ciertas restricciones de tiempo.

El analista estará interesado sólo en los tiempos de respuesta externos, ignorando los tiempos de respuesta internos del sistema que serán tratados por el diseñador. Los tiempos de respuesta externos se relacionan sólo con las señales incluidas en los diagramas de contexto.

La ETR consiste en la documentación del rango permitido del tiempo de respuesta (usualmente el máximo tiempo de respuesta) que debe verificarse desde que un evento ocurre en las terminales de entrada del sistema, hasta que el evento resultante ocurre en las terminales de salida del sistema. Los eventos de entrada ocurren en el ambiente exterior y los eventos de salida son acciones producidas por el sistema. Los eventos son expresados en términos de valores de señales en las terminales del sistema (figura 3.11). Las señales y sus valores permitidos están especificados en el diccionario.

ENTRADA	EVENTO	SALIDA	EVENTO	TIEMPO DE RESPUESTA
A	A = SI	D	D = NO	2 seg.
B	Se ingresa B	C	Se devuelve C	5 seg.

Figura 3.11 Una especificación de tiempos de respuesta.

3.2.8 Diccionario de requerimientos (DR).

El DR provee una definición exacta de todo flujo y banco de datos y de control. Los flujos o señales pueden ser primitivos o no primitivos. Un flujo primitivo es un elemento indivisible de información, que tiene sus propios valores y atributos. Un flujo no primitivo, o flujo de grupo, consiste en un grupo de flujos primitivos.

Atributos de flujos primitivos.

Una señal interna de datos necesita como atributos: nombre y unidades (figura 3.12). Una señal interna de control necesita como atributos: nombre, número de valores y nombre de cada valor. Una señal externa necesita, además de los antes mencionados, los atributos: tasa (que indica cada cuánto debe ser actualizada) y, en caso de ser señal externa continua (de datos), rango (que indica los límites dentro de los cuales la señal existe) y resolución (que indica el incremento de magnitud más pequeño que se requiere representar).

ACELERACION = * Aceleración del automóvil *
Unidades: Kilómetros por hora por segundo.

PEDIDO VEL = * Pedido de reporte de la velocidad promedio *
Valores: SI, NO

VELOCIDAD PROM = * Velocidad promedio *
Unidades: Kilómetros por hora.

Figura 3.12 Algunas definiciones de flijos primitivos.

Los flujos no primitivos representan grupos de señales. El diccionario especifica las componentes y estructura de cada uno de estos grupos usando símbolos especiales (figura 3.13).

SIMBOLO	SIGNIFICADO
=	compuesto por
+	junto con
{ }	iteraciones de
[|]	selecciona uno entre
* *	comentario

EJEMPLOS:
A = [B + C | D + E]
B = O {Q + R} 4
C = S + [N | M]

Figura 3.13 Los símbolos del diccionario de requerimientos y algunas definiciones de flujos no primitivos

3.2.9 Matriz de asignación de requerimientos (MAR).

Una MAR se proporciona para asegurar la completitud y la consistencia de la especificación en relación con los requerimientos del cliente.

Una MAR (figura 3.14) es una tabla que lista los requerimientos del cliente a lo largo de un eje, y las componentes de la especificación de requerimientos a lo largo del otro. Una "X" en una intersección indica que un dado requerimiento del cliente es satisfecho por (o ha sido asignado a) la componente listada.

	COMPONENTES DE LA ESPECIFICACION			
NECESIDADES DEL CLIENTE	P1	P1	P3	EC 0
1		X		
2	X			X
3			X	X
4				

Figura 3.14 Una matriz de asignació de requerimientos.

3.2.10 Diagrama evento-acción (DEA).

Para comprender mejor el sistema desde el punto de vista del cliente y facilitar la construcción del DFD inicial (DFD 0) se propone en [SHU92] el uso de: la lista de eventos, el diagrama evento-acción y la matriz acción/banco de datos.

El DEA, un seudo DFD, se construye en base a estímulos externos (eventos) y a las respuestas a esos estímulos (acciones). Las acciones tienen las mismas características que los procesos (de un DFD de nivel intermedio) y serán agrupadas (usando principios de composición) para construir los procesos del DFD 0.

La lista de eventos, el diagrama evento-acción y la matriz acción/banco de datos son parte de la documentación interna y no serán entregadas al cliente. Si se tiene idea de como debería ser el DFD 0, no es necesario usarlas.

3.2.11 Sumario de la especificación de requerimientos.

La especificación de requerimientos consiste en una estructura de proceso y una estructura de control altamente integradas.

En la estructura de proceso, el conjunto de los DFDs y EPs asociadas modeliza los requerimientos de procesamiento de datos del sistema, independientemente de la implementación del mismo.

En la estructura de control, el conjunto de los DFCs representa el flujo de señales de control hacia las ECs y desde ellas. Las ECs contienen máquinas de estados finitos (TDs, DTEs y TAPs) manejadas por los flujos de control de los DFCs. Las ECs activan y desactivan procesos de los DFDs. Por otro lado, los procesos primitivos generan, a través de testeos sobre señales de datos, señales de control que fluyen en los DFCs. Podemos ver, de este modo, como las ECs integran la estructura de control con la de proceso.

Durante la fase de análisis de requerimientos sólo son de interés los requerimientos de tiempos de respuesta a nivel de sistema, esto es, entre las entradas al sistema y las resultantes salidas. Los mismos serán especificados en la ETR.

Todos los flujos y bancos de los diagramas de flujo son especificados en el DR.

Todas las componentes guardan entre sí relaciones estrictas que deberían ser verificadas en un ambiente de especificación de sistemas de tiempo real.

3.3 Referencias bibliográficas.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.

Capítulo 4: Metodologías de especificación de requerimientos.

La tarea de construir una especificación de requerimientos, en líneas generales, es un proceso que comprende:

- ❑ recibir una especificación de las necesidades del cliente (que puede ser breve o extensa, vaga o específica, formal o informal, escrita u oral),
- ❑ interpretarla,
- ❑ testear su consistencia y completitud,
- ❑ contruir a partir de la misma la especificación de requerimientos,
- ❑ entregar la especificación de requerimientos al diseñador.

Los pasos particularmente involucrados en la construcción de la especificación de requerimientos varían de autor en autor. En general, estos pasos se llevan a cabo secuencial e iterativamente, aunque en la práctica se pueden hacer paralelamente.

Se presentarán a continuación dos de las metodologías usadas para construir una especificación de requerimientos. Se hará una descripción paso por paso de la metodología propuesta por Hatley y Pirbhai [HAT88], y simplemente se destacarán aquellos pasos de la metodología propuesta por Keller y Shumate [SHU92] que la diferencien de la primera.

Cada una de las metodologías estudiadas propone la construcción de una componente interna distinta, esto es, una componente de la especificación que no será entregada a los usuarios de la misma (cliente y diseñador), aunque si conservada para su posterior referencia. El objetivo de estas componentes internas es facilitar la comprensión del sistema desde el punto de vista del cliente para construir un DFD 0 razonablemente completo.

Un ambiente de especificación debería permitir construir especificaciones de requerimientos siguiendo distintas metodologías. El usuario del ambiente podría entonces elegir la metodología que resulte más conveniente, o que sea de su preferencia, para desarrollar su tarea.

4.1 Metodología de Hatley y Pirbhai.

Paso 1. Construir un diagrama de flujo de alto nivel.

- ❑ Organizar la especificación de las necesidades del cliente en grupos principales de funciones.
- ❑ Identificar las entidades externas (otros sistemas, operadores o paneles de control) con las cuales el sistema se debe comunicar.
- ❑ Identificar los grupos principales de información que deben fluir entre el sistema y las entidades externas.

- ❑ Contruir un diagrama de flujo de alto nivel representando los grupos principales de funciones como procesos, las entidades externas como terminadores, y los grupos principales de información como flujos de datos. Este diagrama, que llamaremos "DFD combinado" en razón de que combina en uno los diagramas de contexto (DCD) y nivel 1 (DFD 0), es una componente interna de la especificación de requerimientos (figura 4.1).

- ❑ Estudiar el diagrama resultante para determinar si:

  el alcance de la especificación es correcto, o quizás algún terminador debería ser proceso o algún proceso, terminador;

  los procesos guardan relación con las necesidades del cliente, o quizás la especificación de las mismas debería reparticionarse;

  los flujos son correctos, o quizás deberían modificarse o eliminarse, o inclusive agregarse nuevos flujos, para que todo proceso pueda producir las salidas correspondientes a partir de sus entradas.

- ❑ Hacer los cambios en el diagrama que sean apropiados según lo determinado en el punto previo.

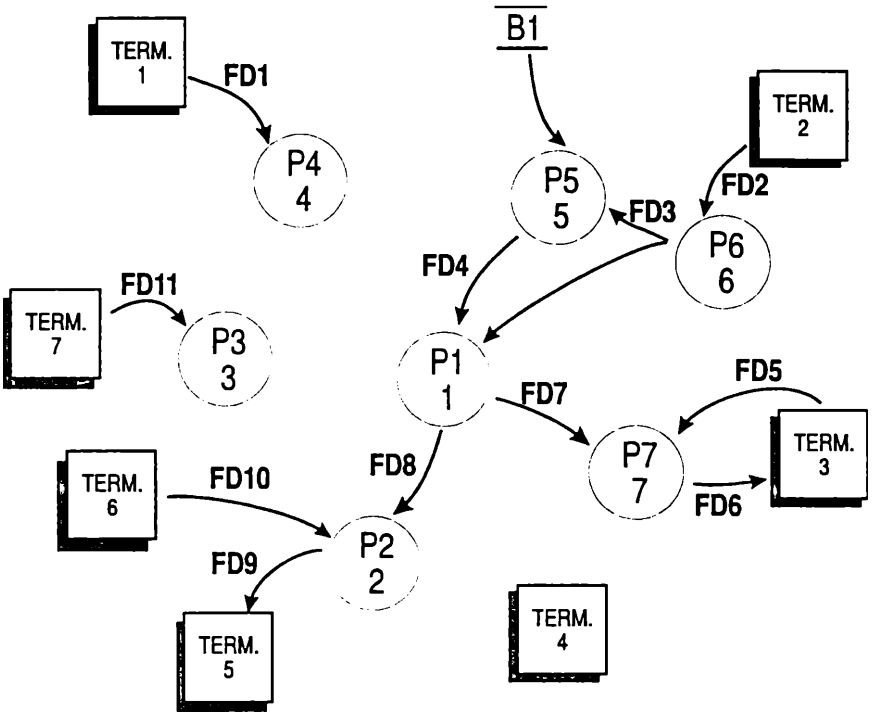


Figura 4.1 Un diagrama combinado de datos.

Paso 2. Construir el DCD.

Dibujar el DCD colapsando los procesos del DFD combinado en un único proceso que represente al sistema, y componiendo flujos, si fuera necesario.

Paso 3. Contruir el DFD 0.

Dibujar el DFD 0 eliminando los terminadores del DFD combinado.

Paso 4. Determinar la necesidad de control a alto nivel.

Examinar las principales necesidades del cliente, y decidir si hay algún modo o condición del sistema bajo el cual alguno de los procesos del DFD 0 requiera activación o desactivación. Si eso ocurre, identificar las señales de control que representan esos modos o condiciones, y construir el DFC y la EC asociados (DFC 0 y EC 0). Antes de construirlos se debería determinar si no se obtiene el mismo efecto cuando ciertos datos están presentes o ausentes en estos modos o condiciones. Se deberían agregar flujos de control sólo cuando sea absolutamente necesario.

Construir el DCC si hay flujos de control desde entidades externas o hacia ellas.

Paso 5. Construir el diccionario de requerimientos.

Cada vez que se dibuja un nuevo flujo de datos o control agregar su definición al diccionario de requerimientos.

Paso 6. Documentar los tiempos de respuesta.

Construir una ETR. Para ello se deberán determinar:

- las entradas al sistema, que son derivadas de los diagramas de contexto;
- el evento de entrada, la señal y el evento de salida asociados a cada una de las entradas;
- el tiempo de respuesta asociado a cada par de señales y eventos de entrada y salida, que es derivado de la especificación de las necesidades del cliente.

Paso 7. Documentar la asignación de requerimientos a los procesos de alto nivel.

Construir una MAR que describa la asignación de las necesidades del cliente a los procesos de alto nivel identificados en el DFD 0.

Paso 8. Construir el siguiente nivel de DFDs.

Tomar cada proceso, y descomponerlo en un diagrama hijo que represente más detalladamente lo que el proceso padre debe hacer. Paso 9. Determinar la nacesidad de control en este nivel.

Paso 9. Determinar la necesidad de control en este nivel.

De igual modo que en el paso 4, decidir si los procesos del nuevo diagrama requieren activación o desactivación. Si es necesario, construir DFCs y ECs.

Paso 10. Actualizar el diccionario de requerimientos.

Agregar cada nuevo flujo o banco, de datos o control, al diccionario de requerimientos.

Paso 11. Documentar la asignación de requerimientos a los procesos de este nivel.

Construir una MAR que describa la asignación de las necesidades del cliente a los procesos de este nivel. Debe incluir al menos aquellas necesidades previamente asignadas al proceso padre.

Paso 12. Continuar la descomposición de procesos.

Tomar cada proceso de este nivel, y descomponerlo en un diagrama hijo (reiterando los pasos 8, 9, 10 y 11), o bien, cuando su función pueda ser expresada claramente en unas pocas líneas de texto o ecuaciones, o con un diagrama simple, escribir su EP.

Cada vez que se han hecho 2 ó 3 niveles, revisarlos para resolver posibles errores, ambigüedades u omisiones, o al menos para anotarlos y resolverlos más adelante.

4.2 Metodología de Keller y Shumate.

Como hemos visto, Hatley y Pirbhai proponen inicialmente la elaboración de un diagrama de flujo de datos de alto nivel, el DFD combinado, que podríamos ubicar entre el DCD (nivel 0) y el DFD 0 (nivel 1), y que incluye procesos de alto nivel y terminadores.

Por su parte, Keller y Shumate proponen, después de la construcción del DCD (paso 1), la elaboración de un diagrama evento-acción o DEA (paso 2). Este diagrama, que podríamos ubicar entre el DFD 0 (nivel 1) y los DFDs hijos de los procesos del DFD 0 (nivel 2), incluye aproximadamente los procesos de nivel 2 (llamados acciones en el contexto del DEA).

Agupando convenientemente las acciones del DEA se identifican cada uno de los procesos del DFD 0 (paso 3).

Después se procede en forma análoga a la determinada por la metodología de Hatley y Pirbhai (paso 4 y subsiguientes).

4.3 Referencias bibliográficas.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.

Capítulo 5: Características de un ambiente de especificación de requerimientos de sistemas de tiempo real.

Un ambiente de especificación tradicional permite representar sólo los productos del análisis estructurado básico (DFDs y diccionario de datos). Estos productos junto con otros que describan el control (DFCs y ECs) y la performance del sistema (ETR) son los que un ambiente de especificación de requerimientos de sistemas de tiempo real debería permitir construir.

Un ambiente de especificación de requerimientos de sistemas de tiempo real debería acompañar, cuando no guiar, el análisis de requerimientos de tales sistemas, siguiendo una determinada metodología. El ambiente podría permitir al usuario seleccionar entre varias metodologías aquella que sea más adecuada para desarrollar su tarea. Las componentes de la especificación de requerimientos, internas y externas, así como el modo en el que estas sean construidas, dependerían entonces de la metodología empleada.

Se presentarán seguidamente los servicios que un ambiente de especificación de requerimientos de sistemas de tiempo real debería proveer. Dichos servicios se describen en términos de operaciones sobre la totalidad de la especificación y sobre cada una de las componentes individuales de la misma. En particular, a modo de ejemplo se describen las operaciones correspondientes al DCD.

5.1 Operaciones sobre la especificación.

Cualquiera que sea la metodología empleada, el ambiente debería permitir:

- ❑ crear una especificación nueva,
- ❑ modificar una especificación existente,
- ❑ testear una especificación, e
- ❑ imprimir una especificación.

El ambiente caracterizado utiliza como soporte físico una microcomputadora. Puede operar sobre una especificación que resida en memoria principal. Llamaremos "especificación de trabajo" a la especificación residente en memoria principal sobre la cual opera el ambiente.

5.1.1 Crear una especificación nueva.

Al crear una especificación de trabajo nueva, el ambiente crearía aquellas componentes de la misma que vayan a ser construidas inicialmente según la metodología empleada. Cada componente sería creada incluyendo toda la información que el ambiente determine automáticamente debería incluir cualquier componente de su tipo. Llamaremos a la misma "información necesaria" de una componente. Por ejemplo, la información necesaria de un DCD sería el proceso, representando al sistema, y un terminador.

Una sesión de trabajo con el ambiente podría iniciarse con una especificación de trabajo nueva, lista para ser completada.

Para conservar las modificaciones hechas sobre la especificación de trabajo, el ambiente debería permitir "salvar" la misma en memoria secundaria.

5.1.2 Modificar una especificación existente.

La especificación de trabajo debería ser actualizada cada vez que se requiera operar sobre una especificación distinta. El ambiente debería permitir "cargar" la especificación de trabajo con aquella especificación almacenada en memoria secundaria que requiera ser modificada.

La especificación de trabajo es modificada cuando alguna de sus componentes es creada, modificada o eliminada. A medida que una componente de la especificación de trabajo es modificada, el ambiente realiza:

- el testeo de la componente modificada de manera de asegurar su correctitud (llamaremos al mismo "testeo dinámico"), y
- la modificación de otras componentes de manera de asegurar su consistencia con la componente modificada.

Para conservar las modificaciones hechas sobre la especificación de trabajo, el usuario debería salvarla.

5.1.3 Testear una especificación.

Además del testeo dinámico realizado automáticamente durante la modificación de la especificación, el ambiente debería permitir realizar un testeo de todas las componentes de la especificación de trabajo de manera de asegurar que están completamente definidas. Llamaremos al mismo "testeo estático".

Durante el testeo estático, el ambiente debería permitir al usuario completar toda componente que así lo requiera. A medida que el usuario completa una componente, el ambiente debería completar automáticamente otras componentes de manera de asegurar la consistencia. Por ejemplo, al testear estáticamente la especificación de trabajo, el ambiente podría detectar que un flujo de datos del DCD no posee nombre, y seguidamente podría permitir al usuario completarlo. Cuando el usuario nombra dicho flujo, el ambiente debería incluirlo en el diccionario de requerimientos.

5.1.4 Imprimir una especificación.

El ambiente debería permitir imprimir todas las componentes de la especificación de trabajo.

Puesto que algunas componentes guardan una relación muy cercana (por ejemplo: el DFD 0, el DFC 0 y la EC 0; o cualquier otra terna formada por un DFD, un DFC y una EC asociados), el ambiente podría imprimirlas en grupo, y estos grupos en orden numérico, de manera de facilitar la lectura de la especificación.

5.2 Operaciones sobre las componentes.

El ambiente debería permitir operar sobre cualquiera de las componentes de la especificación de trabajo en forma individual.

Cuando el usuario esté operando sobre una componente, el ambiente debería permitirle pasar fácilmente a operar sobre otra. Esto resultaría muy conveniente, en particular, si esas componentes guardan entre sí una relación tan cercana que haga necesario iterar entre ellas con frecuencia.

En general, el ambiente debería permitir:

- crear,
- modificar,
- testear,
- imprimir, y
- eliminar una componente.

5.2.1 Crear una componente.

El usuario debería determinar la creación de algunas componentes de la especificación. Por ejemplo, una EP podría ser creada cuando el usuario lo considere apropiado.

Otras componentes se deberían crear automáticamente. Por ejemplo, el DCD podría ser creado por el ambiente cuando la especificación sea creada. En general, dependerá de la metodología empleada cuáles serán las componentes creadas automáticamente al crear una especificación nueva.

Cada componente podría ser creada incluyendo información necesaria, esto es, el ambiente determinaría automáticamente que información debe necesariamente incluir una componente nueva según su tipo, y de manera de asegurar su consistencia con las componentes previamente creadas.

5.2.2 Modificar una componente.

El ambiente debería permitir modificar cualquiera de las componentes de la especificación de trabajo. Esto es, el ambiente debería permitir ampliar o corregir una componente agregando, modificando o eliminando elementos de la misma.

A medida que una componente es modificada, el ambiente realiza:

- el testeo dinámico de la misma, para asegurar que sea correcta, y
- la modificación de otras componentes, para asegurar que sean consistentes con la componente modificada.

Para conservar las modificaciones hechas a una componente, el usuario debería salvar la especificación de trabajo.

5.2.3 Testear una componente.

El ambiente debería permitir el testeo estático de cualquiera de las componentes de la especificación de trabajo para asegurar que está completamente definida.

A medida que una componente es testeada, el ambiente debería:

- permitir al usuario completar la descripción de aquellos elementos de la componente testeada que así lo requieran, y
- completar automáticamente otras componentes, a medida que el usuario completa la componente testeada, de manera de asegurar la consistencia.

Para conservar la información agregada durante el testeo, el usuario debería salvar la especificación de trabajo.

5.2.4 Imprimir una componente.

El ambiente debería permitir la impresión de cualquiera de las componentes de la especificación de trabajo, ya sea interna o externa.

El ambiente también podría permitir la impresión parcial de ciertas componentes. Por ejemplo, el usuario podría desear imprimir del diccionario de requerimientos sólo las definiciones de flujos y bancos incluidos en el DFD 0, o en cualquier otro diagrama de flujo en particular.

5.2.5 Eliminar una componente.

En general, es el usuario quien debería determinar la eliminación de una componente de la especificación de trabajo. Por ejemplo, una EP podría ser eliminada cuando el usuario considere que el proceso padre fue prematuramente especificado y requiere, en cambio, ser descompuesto en un diagrama de flujo de datos hijo.

Cuando el usuario elimina una componente de la especificación de trabajo, el ambiente automáticamente debería eliminar aquellas componentes que sean dependientes, y modificar aquellas otras que, sin llegar a ser dependientes, estén relacionadas, de manera de mantener la consistencia entre las componentes. Por ejemplo, cuando el usuario elimina el DFD 0, el ambiente automáticamente debería eliminar los DFDs hijos y las EPs de los procesos incluidos en el DFD 0, y también el DFC 0. Junto con ellos debería eliminar sus componentes dependientes. Además, el ambiente automáticamente debería modificar el diccionario de requerimientos, eliminando las definiciones de los flujos y bancos de datos incluidos en el DFD 0.

De aquí en adelante, se describen algunas operaciones sobre el diagrama de contexto de datos, una de las primeras componentes de la especificación en ser desarrolladas.

5.3 Operaciones sobre el DCD.

5.3.1 Crear el DCD.

Como se dijo anteriormente, los pasos a seguir y el modo según el cual las componentes de una especificación son creadas y modificadas, dependen de la metodología empleada.

En la metodología de Hatley y Pirbhai [HAT88], el DCD se deriva del DFD combinado inicialmente construido. Si el ambiente emplea esta metodología, debería permitir al usuario crear un DCD que automáticamente incluiría como información necesaria: el proceso, los terminadores del DFD combinado y los flujos de datos del mismo diagrama que fluyen desde los terminadores o hacia ellos.

En la metodología de Keller y Shumate [SHU92], el DCD es inicialmente construido. Si el ambiente emplea esta metodología, cuando se crea una especificación nueva, debería automáticamente crear un DCD que incluiría como información necesaria: el proceso y un terminador.

5.3.2 Modificar el DCD.

El ambiente debería permitir ampliar o corregir el DCD agregando, modificando o eliminando elementos del mismo.

5.3.2.1 Agregar elementos.

Se debería permitir representar nuevos terminadores y flujos de datos entre el proceso y los terminadores, dibujando sus correspondientes rectángulos y arcos dirigidos.

Testeo dinámico.

No se debería permitir representar flujos entre terminadores, ni flujos con origen y destino simultáneamente en un mismo proceso o terminador. Tampoco se debería permitir superponer las representaciones de los elementos.

Consistencia.

Cuando sean agregados al DCD nuevos flujos de entrada al proceso (o de salida del mismo), el ambiente debería automáticamente representar en DFD 0, si éste existe, flujos que fluyan desde el entorno y con destino indefinido (o hacia el entorno y con origen indefinido).

Si el ambiente emplea la metodología de Hatley y Pirbhai, debería automáticamente representar en el DFD combinado los nuevos terminadores y flujos de datos del DCD, estos últimos en forma análoga a la enunciada en el párrafo anterior.

Si el DCC existe, el ambiente debería automáticamente representar en él los nuevos terminadores del DCD.

5.3.2.2 Modificar elementos.

Se debería permitir nombrar o renombrar el proceso, los terminadores y los flujos de datos del DCD.

Además, se debería permitir modificar la representación de los mismos. En el caso del proceso o de un terminador, se debería permitir mover su círculo o rectángulo; los flujos que lo tengan como origen o destino se deberían automáticamente mover con él. En el caso de un flujo, se debería permitir cambiar origen o destino de su arco dirigido.

Testeo dinámico.

Se debería impedir representar flujos entre terminadores, y flujos con origen y destino simultáneamente en un mismo proceso o terminador.

Consistencia.

Cuando el usuario nombra un flujo, el ambiente debería incluir en el diccionario una nueva entrada para dicho nombre. Cuando lo renombra, el ambiente debería actualizar las entradas del diccionario asociadas a dicho flujo y a sus flujos ancestros.

Si el DFD 0 existe, el ambiente debería automáticamente nombrar o renombrar los flujos de datos que posean nuevos nombres. De igual modo, debería actualizar el título del DFD 0 con el nuevo nombre del proceso.

Si el ambiente emplea la metodología de Hatley y Pirbhai, debería automáticamente nombrar o renombrar terminadores y flujos de datos del DFD combinado, o modificar sus representaciones, según corresponda.

Si el DCC existe, el ambiente debería automáticamente actualizar el nombre del proceso (y, junto con él, el título del DFC 0), o el nombre de aquel terminador que haya sido modificado.

5.3.2.3 Eliminar elementos.

Se debería permitir eliminar terminadores y flujos de datos, borrando sus correspondientes rectángulos y arcos dirigidos.

Cuando un terminador es eliminado, los flujos de datos que lo tengan como origen o destino serían eliminados.

Testeo dinámico.

Se debería impedir eliminar un terminador cuando sea el único terminador representado pues, al igual que el proceso, debería estar necesariamente presente en todo DCD.

Cuando el proceso del DCD es descompuesto en su DFD hijo, el DFD 0, sus flujos de entrada o salida, o subflujos de los mismos, pueden definirse como entradas o salidas de los subprocesos. Eliminar un flujo de un diagrama implicaría eliminarlo junto con sus subflujos, de éste y de todo diagrama dependiente. Si se elimina un flujo del DCD, el ambiente podría prevenir al usuario acerca de las consecuencias que ello traería, y solicitar confirmación para continuar.

Consistencia.

Como se dijo anteriormente, cuando un flujo es eliminado de un diagrama, él y sus subflujos deberían ser eliminados de todo diagrama dependiente. Si el DFD 0 existe, cuando se elimina un flujo del DCD, el ambiente debería automáticamente eliminarlo junto con sus subflujos del DFD 0 y de sus diagramas dependientes. También deberían ser eliminadas las entradas del diccionario y de las tablas que correspondan.

Si el DCC existe, cuando el usuario elimina un terminador del DCD, el ambiente debería automáticamente eliminarlo del DCC junto con los flujos de control que de él pudieran salir o entrar. Otro tanto debería ocurrir en el DFD combinado.

5.3.3 Testear el DCD.

El ambiente debería permitir testear estáticamente el DCD para que el usuario pueda:

- nombrar aquellos elementos (proceso, terminadores y flujos de datos) que no posean nombre, y
- definir el origen o el destino de aquellos flujos de datos que así lo requieran.

A medida que el usuario completa la componente testeada, el ambiente debería automáticamente completar las componentes relacionadas a ella, de manera de mantener la consistencia.

El prototipo del ambiente de especificación de requerimientos de sistemas de tiempo real, presentado en el capítulo que sigue, proveerá los servicios descriptos.

5.4 Referencias bibliográficas.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.

Capítulo 6: Prototipo de ambiente de especificación de requerimientos de sistemas de tiempo real.

6.1 Ambiente de desarrollo del prototipo: Smalltalk/V 286.

Consideremos la naturaleza jerárquica de los sistemas y de su proceso de desarrollo. Un sistema puede ser expresado como una jerarquía. Cualquier nivel de esta jerarquía representa al sistema con un grado de detalle particular. En un cierto nivel, un conjunto de requerimientos es generado (fase de análisis). Estos requerimientos son asignados a unidades físicas (fase de diseño). Los requerimientos asignados a cada unidad son definidos más detalladamente, y el proceso completo se repite en el siguiente nivel de la jerarquía. En el último nivel, tiene lugar la implementación. También el universo puede ser visto como una jerarquía, en este caso, una jerarquía de sistemas [HAT88].

Esta visión de los sistemas y el universo se corresponde con la jerarquía de clases de Smalltalk. Tal correspondencia nos llevó a considerar las ventajas de usar Smalltalk/V 286 para implementar el prototipo:

Este ambiente de desarrollo de programas proporciona herramientas de programación que permiten

- usar, reusar y modificar el código fuente Smalltalk que es parte del sistema básico; y
- detectar y corregir errores de programación.

Ello promueve el uso de prototipación para desarrollar rápida y eficientemente aplicaciones complejas, entre las que podría contarse nuestro ambiente de especificación de requerimientos de sistemas de tiempo real.

Además, como consecuencia de considerar que el usuario es la componente más importante en un sistema de cómputos (una de las ideas en base a las que se diseño Smalltalk), se facilita el desarrollo de interfaces amigables.

Requerimientos del prototipo.

Los requerimientos del prototipo son los propios del Smalltalk/V 286:

- Una IBM PC, PS/2 o compatible, con un procesador 80286 u 80386.
- Dos Megabytes de RAM.
- Un disco rígido y una disquetera.
- Un monitor monocromático o color.
- Una plaqueta controladora de gráficos.
- Un sistema operativo PC-DOS o MS-DOS (Versión 2.0 o posterior).
- Un "mouse".

El Smalltalk/V 286 soporta distintos adaptadores de video. Cuando se instala, el usuario debe ingresar el tipo de adaptador de video correspondiente a la configuración de hardware que posee. Esta es una propiedad que el prototipo hereda del Smalltalk/V 286.

ADVERTENCIA:

Si hay instalado un manejador de memoria extendida (por ejemplo, HIMEM.SYS) el Smalltalk/V 286 automáticamente considera que la memoria no está disponible y, en consecuencia, no puede correr. Se deberá eliminar del archivo de configuración CONFIG.SYS el comando correspondiente a dicho manejador (en nuestro ejemplo, DEVICE=C:\HIMEM.SYS) y reiniciar la operación del sistema ("reboot").

6.2 Descripción del prototipo.

6.2.1 Metodología de especificación.

El prototipo guía la especificación de requerimientos de un sistema de tiempo real siguiendo la metodología propuesta por Hatley y Pirbhai [HAT88]. Como vimos, dicha metodología comprende:

Paso 1. Construir un diagrama de flujo de alto nivel: el DFD combinado (componente interna de la especificación de requerimientos).

Paso 2. Construir el DCD.

Paso 3. Contruir el DFD 0.

Paso 4. Determinar la necesidad de control a alto nivel.

Paso 5. Construir el diccionario de requerimientos.

Paso 6. Documentar los tiempos de respuesta.

Paso 7. Documentar la asignación de requerimientos a los procesos de alto nivel.

Paso 8. Construir el siguiente nivel de DFDs.

Paso 9. Determinar la nacesidad de control en este nivel.

Paso 10. Actualizar el diccionario de requerimientos.

Paso 11. Documentar la asignación de requerimientos a los procesos de este nivel.

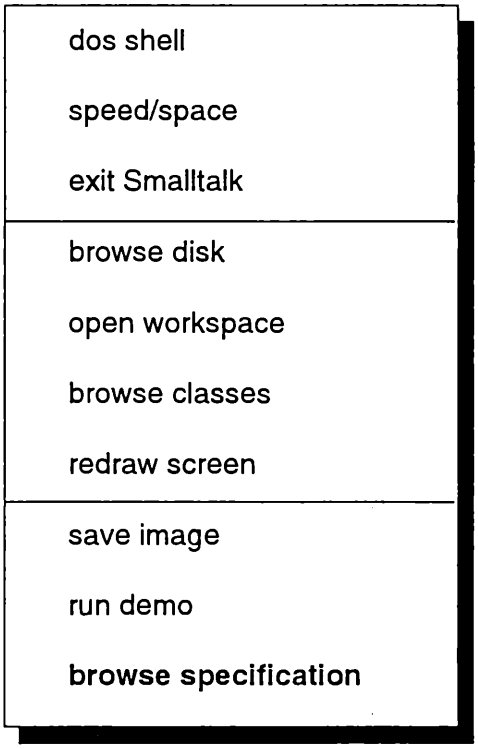
Paso 12. Continuar la descomposición de procesos.

La razón de esta elección reside en que la tarea de determinar el alcance del sistema se ve simplificada si inicialmente se combinan en un mismo diagrama terminadores y procesos principales. Esto facilita el decidir si un cierto terminador debería ser proceso, o viceversa.

La correcta identificación de las entidades externas y los principales flujos de información entre estas y el sistema, puede no ser una tarea de fácil resolución si no se tienen en cuenta los requerimientos funcionales principales. En consecuencia, iniciar la especificación construyendo el DCD, tal como proponen Keller y Shumate [SHU92], puede no ser apropiado.

6.2.2 Servicios que provee el prototipo.

El prototipo del ambiente de especificación de sistemas de tiempo real será llamado "browser" de especificaciones en el contexto del Smalltalk/V. Se abre seleccionando la opción "browse specification" del menú del sistema del Smalltalk/V, menú que es obtenido presionando el botón derecho del mouse sobre la zona gris de la pantalla.



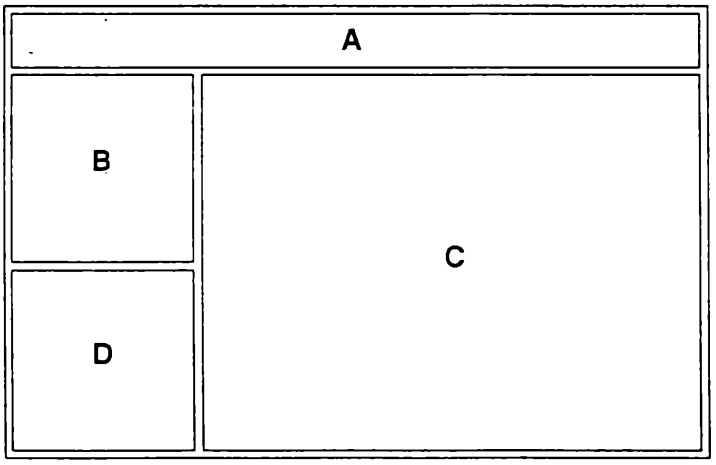
La ventana del browser comprende cuatro cuadros principales ("panes"):

A) Cuadro de operaciones.

B) Cuadro de la jerarquía.

C) Cuadro de edición.

D) Cuadro de panorama.



A) Cuadro de operaciones: ubicado en la parte superior de la ventana, es un cuadro con iconos que representan las operaciones que se pueden efectuar sobre una especificación. Cuando un icono es seleccionado, se realiza la correspondiente operación.

B) Cuadro de la jerarquía: ubicado en la parte central izquierda de la ventana, es un cuadro de lista que muestra las distintas componentes de la especificación sobre la que el browser está operando, ordenadas jerárquicamente. Cuando una componente es seleccionada, se permite operar sobre ella en forma individual.

C) Cuadro de edición: ubicado en la parte inferior derecha de la ventana, es un cuadro que permite editar (o testear) la componente de la especificación seleccionada en el cuadro de la jerarquía. Si la componente es textual o gráfica, comprende un único pane (de texto o gráfico); si es tabular, dos panes (uno de lista y otro de texto).

D) Cuadro de panorama: ubicado en la parte inferior izquierda de la ventana, es un cuadro que muestra una representación simplificada y a escala reducida de la componente editada (o testeada) en el cuadro de edición.

A) Cuadro de operaciones.

En este cuadro se muestran las operaciones asociadas a una especificación en su totalidad. Durante la edición de una especificación, se tiene el siguiente cuadro de operaciones:

Crear	Cargar	Renombrar	Salvar	Testear	Imprimir	Ayuda	Salir

La opción **Crear** permite crear una nueva especificación. Presenta una ventana de ingreso del nombre de la misma. Según la metodología elegida, la componente inicialmente construida en toda nueva especificación es: la especificación de las necesidades del cliente. Al crear una nueva especificación en memoria, el browser crea su correspondiente especificación de las necesidades del cliente. Una vez creada, la especificación del sistema puede ser editada: las componentes existentes pueden ser modificadas; y nuevas componentes, creadas.

Una especificación existente, almacenada en un archivo, debe cargarse en memoria para ser editada. La opción **Cargar** presenta una lista de las especificaciones existentes; la especificación seleccionada es cargada en memoria y puede ser editada.

La opción **Renombrar** permite modificar el nombre de la especificación que se está editando. Presenta una ventana de ingreso del nuevo nombre de la misma.

La opción **Salvar** permite almacenar en un archivo la especificación que se está editando, conservando las modificaciones hechas.

La opción **Testear** permite iniciar el testeo de la especificación. Durante el mismo, la

especificación puede seguir siendo editada. A las características propias de la edición, se agregan entonces las siguientes: se informa el estado de la especificación (completa o incompleta), y se identifican sus componentes incompletas para facilitar de este modo que su descripción sea completada. Durante el testeo de una especificación, se tiene el siguiente cuadro de operaciones:

Crear	Cargar	Renombrar	Salvar	Editar	Imprimir	Ayuda	Salir

La opción **Editar** permite continuar con la edición normal de la especificación, finalizando de este modo el testeo.

La opción **Imprimir** permite imprimir todas las componentes de la especificación que se está editando (o testeando). (Será implementada en una versión posterior del ambiente.)

La opción **Ayuda** permite obtener un mensaje de ayuda en función del contexto. (Será implementada en una versión posterior del ambiente.)

La opción **Salir** finaliza la sesión de trabajo con el browser y retorna al Smalltalk/V. Permite salvar la especificación si hubiera sido modificada desde la última operación de salvado.

El menú asociado al cuadro de operaciones, obtenido al presionar el botón derecho del mouse sobre dicho cuadro, a su vez presenta las opciones descriptas.

B) Cuadro de la jerarquía.

En este cuadro se muestran las componentes de la especificación sobre la que se está operando. Las mismas están ordenadas jerárquicamente. Para cualquier componente, la indentación indica su nivel en la jerarquía; y la componente precedente más próxima de nivel superior, su componente padre.

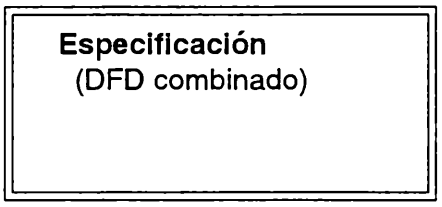
```
Especificación
  DCD
    DFD combinado
    (DFC combinado)
       DFD 0
         EP 1
         (DFD 2/EP 2)
  DR
  ETR
```

Una componente que aún no ha sido creada se muestra entre paréntesis en la jerarquía.

Cuando una componente de la jerarquía es seleccionada, su representación (gráfica, textual o tabular, según corresponda) —es creada incluyendo información necesaria, en el caso de una componente no creada— es mostrada en el cuadro de edición y puede ser editada.

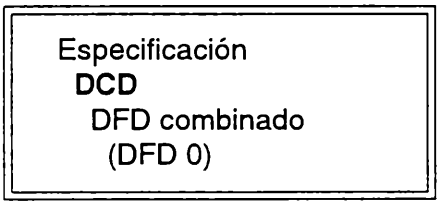
Durante el testeo de la especificación, una componente incompleta se muestra entre corchetes en la jerarquía (otro tanto ocurre durante el testeo individual de la componente que será descripto más adelante).

La metodología elegida determina qué componentes de la especificación pueden ser creadas y editadas siguiendo los pasos correspondientes. Así se tiene que inicialmente sólo puede ser editada la especificación de las necesidades del cliente (creada automáticamente al crear una nueva especificación), y sólo puede ser creado el DFD combinado (paso 1).



Después puede ser creado el DCD (paso 2);

más tarde, el DFD 0 (paso 3); y así siguiendo.



Por otro lado, las modificaciones que sufren las componentes existentes determinan que otras componentes sean creadas o puedan ser creadas. Por ejemplo, al representar el primer flujo se crea el diccionario de requerimientos, y al representar un proceso en el DFD 0 se puede crear su DFD hijo o su EP. Estas componentes, creadas o por crear, son mostradas automáticamente en la jerarquía. (En la presente versión del ambiente los procesos del DFD 0 serán especificados, pues sólo se representarán diagramas de nivel menor que dos.)

C) Cuadro de edición.

Este cuadro muestra la componente de la especificación seleccionada en el cuadro de la jerarquía. Permite ampliar o corregir dicha componente durante su edición (o testeo).

El número y el tipo de panes que componen el cuadro dependerán de la componente. Si la componente es gráfica (DFD combinado, DCD o DFD 0), comprende únicamente un pane gráfico; si es textual (especificación de las necesidades del cliente o especificaciones de proceso), un pane de texto. Finalmente, si la componente es tabular (DR o ETR), comprende dos panes: un pane de lista y un pane de texto.

Un menú asociado a la componente, incluye las operaciones que se pueden efectuar sobre la misma. A su vez, todo elemento (o fila) de una componente gráfica (o tabular) tendrá su correspondiente menú. Las opciones que cada menú presenta dependerán del modo de operación (edición o testeo) y del contexto.

C.1) Cuadro de edición de una componente gráfica.

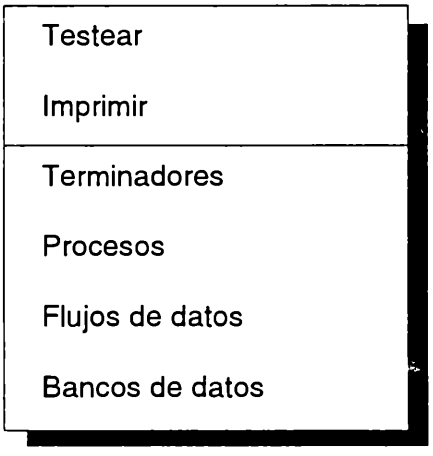
Las componentes gráficas que la presente versión del ambiente permite representar son, siguiendo el orden de su creación, el DFD combinado, el DCD y el DFD 0.

En ellos los terminadores son representados por rectángulos; los procesos, por círculos; los flujos de datos, por arcos dirigidos; y los bancos de datos, por dos líneas paralelas horizontales. (La representación de estos últimos será implementada en una versión posterior del ambiente.)

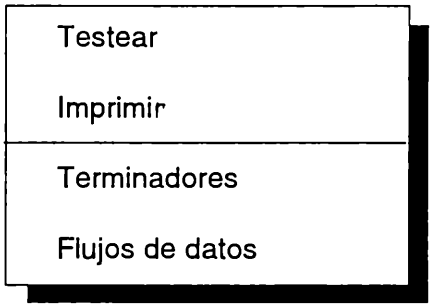
Edición de una componente gráfica.

Durante su edición, cada componente gráfica presentará su correspondiente menú, obtenido presionando el botón derecho del mouse sobre una zona del cuadro libre de elementos.

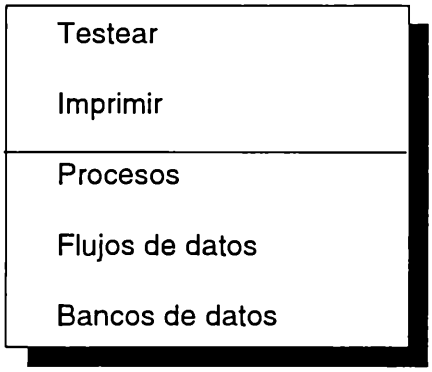
El menú del DFD combinado es:



El menú del DCD es:



El menú del DFD 0 es:



La opción **Testear** permite iniciar el testeo de la componente. Durante el mismo, la componente puede seguir siendo editada. A las características propias de la edición, se agregan entonces las siguientes: se informa el estado de la componente (completa o incompleta), y se identifican sus elementos incompletos para facilitar de este modo que su descripción sea completada.

La opción **Imprimir** permite imprimir la componente en forma individual.

Las restantes opciones permiten representar nuevos elementos en la componente. Toda vez que se selecciona una de estas opciones, nuevos terminadores, procesos o bancos, según corresponda, se representan de la siguiente manera:

1.- posicionar el cursor en un punto del cuadro de edición, y

2.- presionar el botón izquierdo del mouse.

La representación característica del nuevo elemento aparece; y en el centro del mismo, su nombre.

Para representar nuevos flujos:

1.- posicionar el cursor sobre el elemento origen del flujo y presionar el botón izquierdo del mouse,

2.- posicionar el cursor sobre el elemento destino del flujo y presionar el botón izquierdo del mouse.

Un arco dirigido aparece uniendo las representaciones de los elementos origen y destino; y en el centro del mismo, el nombre del flujo.

Con el propósito de identificar cada nuevo elemento, el browser le asigna un nombre. Un elemento no se considerará completamente descripto si el usuario no lo renombra.

Se rechaza representar nuevos elementos cuando se superponen con elementos existentes. (Será implementado en una versión posterior del ambiente.) Además, se rechaza representar nuevos flujos entre terminadores, entre bancos, o desde un elemento cualquiera hacia sí mismo.

En el DFD 0, para representar un nuevo flujo de salida del sistema:

1.- seleccionar el proceso origen del flujo, en la forma indicada anteriormente, y

2.- posicionar el cursor fuera del cuadro de edición y presionar el botón izquierdo del mouse.

En este único caso, se permite representar un nuevo flujo sin que sea necesario determinar su correspondiente elemento destino. Ello ocurre pues el elemento destino de un flujo de salida del sistema, algún terminador, está fuera de contexto en el DFD 0.

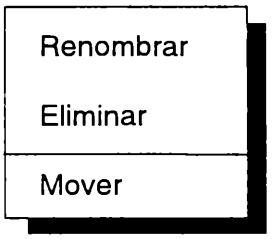
En el DCD, para representar un nuevo flujo de entrada al sistema basta seleccionar el terminador origen del flujo. Se determina automáticamente al sistema como destino del flujo, en razón de que no se pueden representar flujos entre terminadores.

Durante la edición de una componente gráfica, los elementos existentes pueden ser modificados. Las modificaciones que sufre un elemento en una componente, serán reflejadas en toda otra componente en la que también esté presente.

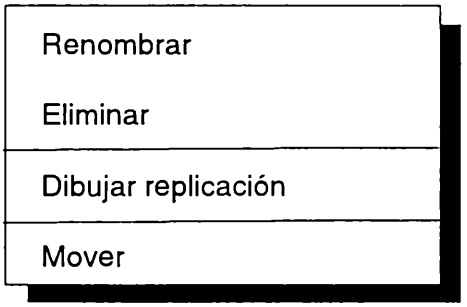
Si se mantiene presionado el botón izquierdo del mouse sobre un elemento, su representación se desplaza siguiendo los movimientos del mouse. Donde se libera el botón, el elemento es trazado nuevamente.

Si se presiona el botón derecho del mouse sobre un elemento, éste presenta su correspondiente menú.

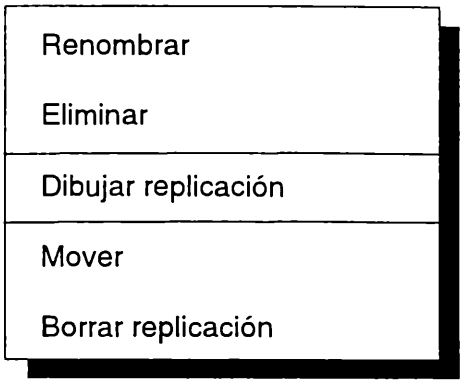
Un proceso presenta el siguiente menú:



Un terminador o un banco (según sea replicado -1-, o no lo sea -2-) presenta uno de los siguientes menúes:



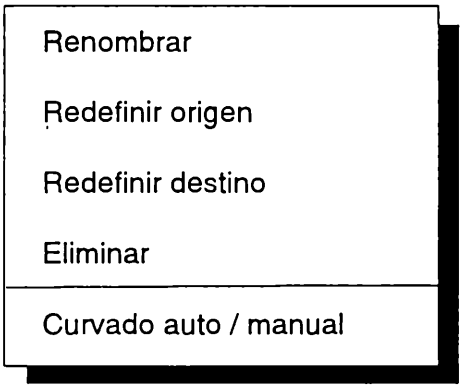
(1)



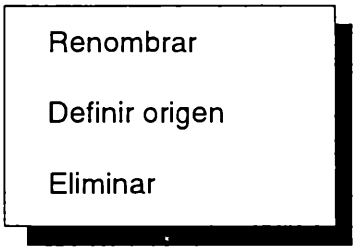
(2)

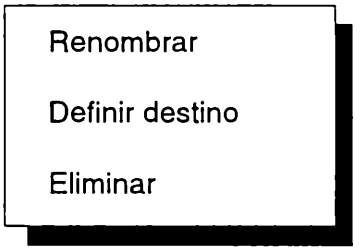
Un flujo (según tenga ambos extremos definidos -1-, o sólo uno de ellos -2-) presenta uno de los siguientes menúes:

(1)



(2)





La opción **Renombrar** permite modificar el nombre del elemento. Presenta una ventana de ingreso del nuevo nombre del mismo.

La opción **Eliminar** permite eliminar el elemento de la especificación. Cuando algún terminador, proceso o banco es eliminado, los flujos que lo tengan como origen o destino también serán eliminados. En función de su contexto, ciertos elementos no pueden ser eliminados. Por ejemplo, el sistema del DCD o el único terminador presente en el diagrama. En estos casos, la opción Eliminar se omite del correspondiente menú.

Un terminador o un banco pueden ser replicados, esto es, dos o más representaciones del elemento pueden ser dibujadas en una misma componente gráfica cuando sea necesario esclarecer el diagrama. (La representación de replicaciones será implementada en una versión posterior del ambiente. En la presente versión, los terminadores tienen el mismo menú que los procesos.)

La opción **Mover** permite desplazar la representación del elemento siguiendo los movimientos del mouse. Donde se presiona el botón derecho del mouse, el elemento es trazado nuevamente.

La opción **Redefinir origen (Redefinir destino)** permite seleccionar un nuevo elemento origen (destino) de un flujo.La opción **Definir origen (Definir destino)** permite seleccionar el elemento origen (destino) de un flujo. Si un flujo carece de uno de sus elementos extremos, su correspondiente menú presenta la opción que permita definirlo, y omite aquella que permita redefinir el extremo presente.

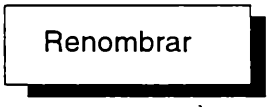
Un flujo es representado por un arco dirigido curvado. El punto del cuadro de edición hacia donde se curva un arco puede ser determinado automática o manualmente. La opción de curvado permite cambiar el modo según el cual se determina dicho punto. Todo nuevo flujo es curvado automáticamente; su correspondiente menú presenta la opción **Curvado auto/manual.** Si esta opción es seleccionada, el flujo puede ser curvado manualmente; entonces, su correspondiente menú presenta la opción **Curvado manual/auto.** (La representación de flujos curvados manualmente será implementada en una versión posterior del ambiente. En la presente versión, los flujos son curvados automáticamente; y la opción de curvado, omitida.)

Testeo de una componente gráfica.

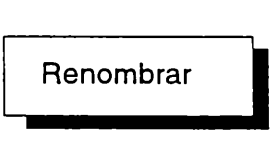
Durante su testeo individual, toda componente gráfica presentará la opción **Editar** en su correspondiente menú. Esta opción permite continuar con la edición normal de la componente, finalizando de este modo su testeo individual.

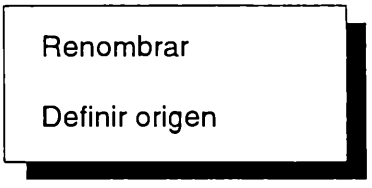
Los elementos incompletos de toda componente pueden ser fácilmente completados durante el testeo (individual o colectivo). Si se presiona el botón derecho del mouse sobre uno de tales elementos, este presenta un menú que incluye: la opción **Completar**, junto con aquellas opciones que permitan modificar los atributos del elemento que ya han sido definidos. La opción Completar presenta, a su vez, un menú que incluye sólo aquellas opciones que permitan completar los atributos del elemento que aún no han sido definidos.

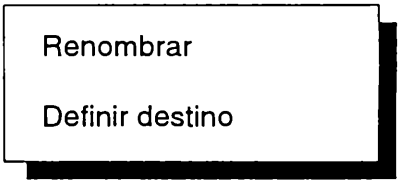
Al seleccionar la opción Completar, cualquier terminador, proceso o banco incompleto presenta el siguiente menú:



Un flujo incompleto presenta uno de los siguientes menúes:







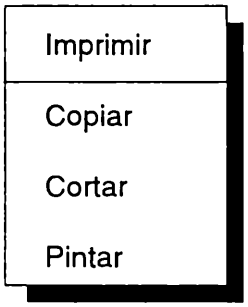
La opción **Renombrar** se incluye cuando el elemento conserva el nombre asignado por el browser.

C.2) Cuadro de edición de una componente textual.

Las componentes textuales que la presente versión del ambiente permite representar son: la especificación de las necesidades del cliente y las especificaciones de proceso.

Edición de una componente textual.

Durante su edición, toda componente textual presentará el siguiente menú:



La opción **Imprimir** imprime la componente en forma individual.

Se permitirá insertar, seleccionar, reemplazar y eliminar texto en la forma en que habitualmente se hace en Smalltalk/V [DIG88, p. 271-274]. Las opciones **Copiar**, **Cortar** y **Pintar** tienen el comportamiento de las funciones copy, cut y paste para la edición de texto en Smalltalk/V.

Testeo de una componente textual.

Las componentes textuales no son automáticamente testeables. Es el usuario quien debe determinar si están completas.

Durante el testeo de la especificación, una componente textual podrá ser editada normalmente.

C.3) Cuadro de edición de una componente tabular.

Las componentes tabulares que la presente versión del ambiente permite representar son: el diccionario de requerimientos (DR) y la especificación de tiempos de respuesta (ETR).

En el DR, cada fila define un flujo, un banco o un elemento auxiliar de información incluido en otras definiciones del diccionario con el objeto de aportar claridad. (En la presente versión del ambiente, sólo los flujos son definidos.) Un flujo primitivo está definido por los valores de sus atributos. Un flujo no primitivo está definido en función de otros flujos y/o elementos auxiliares de información, estructurados apropiadamente.

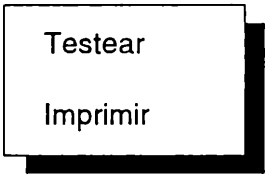
En la ETR, cada fila describe el tiempo de respuesta que media entre una entrada al sistema y su correspondiente salida; incluye flujos y eventos de entrada y salida, y el tiempo de respuesta asociado.

El cuadro de edición de una componente tabular comprende dos panes: un pane de lista y un pane de texto. El primero —ubicado en la parte izquierda del cuadro— lista las entradas que identifican a cada una de las filas de la tabla. El segundo —ubicado en la parte derecha del cuadro— muestra la tabla, y permite modificar la fila seleccionada, que es aquella que se identifica con la entrada seleccionada en el pane de lista.

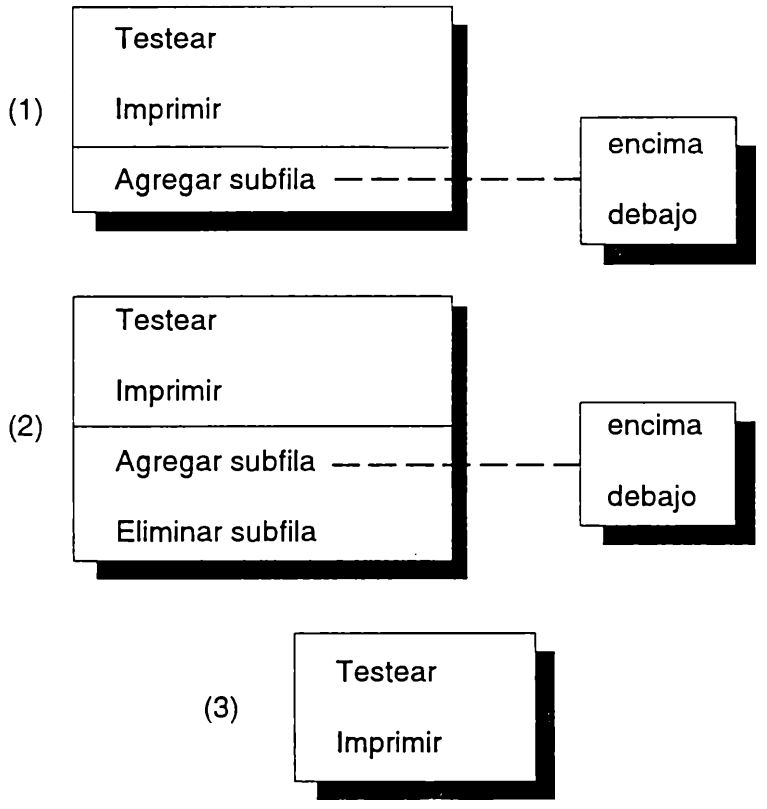
Edición de una componente tabular.

Durante su edición, cada componente tabular presentará su correspondiente menú, obtenido presionando el botón derecho del mouse sobre el pane de lista.

El DR presentará el siguiente menú:



La ETR (según esté seleccionada una fila -1- o una subfila -2-, o ninguna lo esté -3-) presentará uno de los siguientes menúes:



Las opciones **Testear** e **Imprimir** permiten testear e imprimir la tabla.

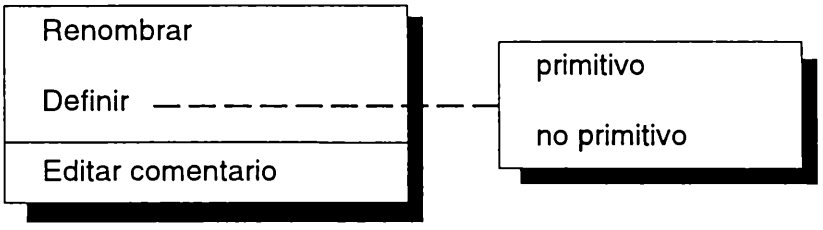
Un mismo flujo de entrada puede producir distintas salidas, cada una a su debido tiempo. La opción **Agregar subfila**, presente sólo cuando una fila de la ETR está seleccionada, permite agregar a la tabla una nueva fila, que estará asociada al mismo flujo de entrada al que la fila seleccionada está asociada. Presenta un menú para ubicar la nueva fila por **encima** o por **debajo** de la seleccionada. La nueva fila es entonces seleccionada automáticamente. Toda fila asociada

al mismo flujo de entrada al que alguna otra está asociada, se llamará subfila.

La opción **Eliminar subfila**, presente sólo cuando una subfila de la ETR está seleccionada, permite eliminar de la tabla la subfila seleccionada.

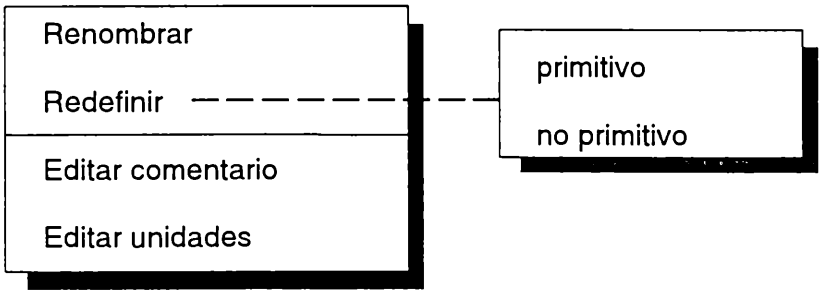
Durante la edición del DR, la fila seleccionada puede ser modificada. Si se presiona el botón derecho del mouse sobre el pane de texto, se desplegará el menú asociado a la fila seleccionada. Las características de un flujo determinan las opciones que presenta su menú.

Asociado a un flujo indefinido:

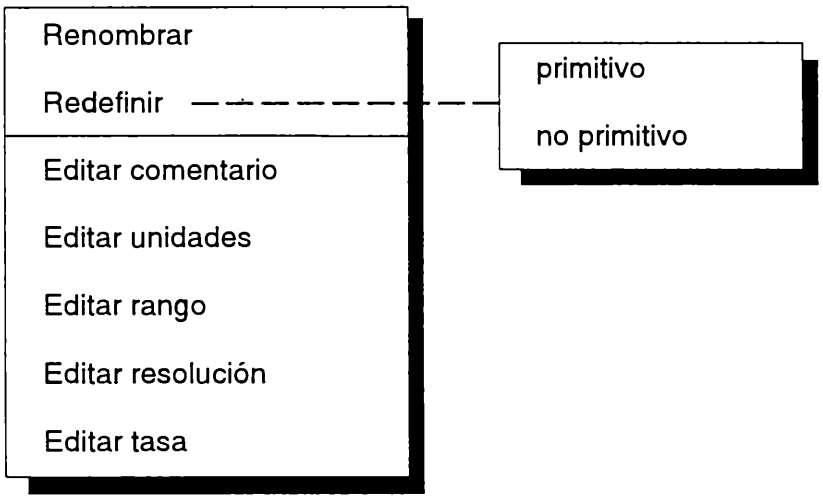


Asociado a un flujo primitivo de datos (según sea interno -1- o externo -2-):

(1)

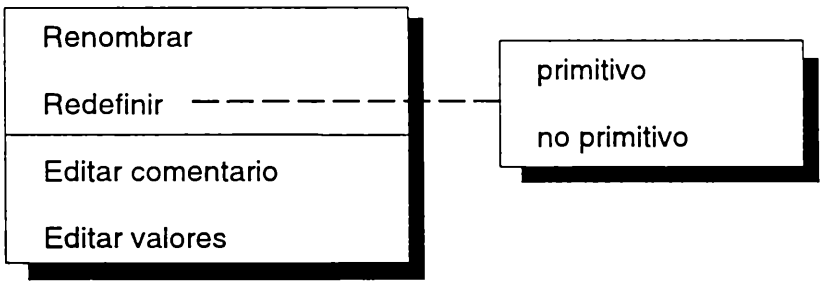


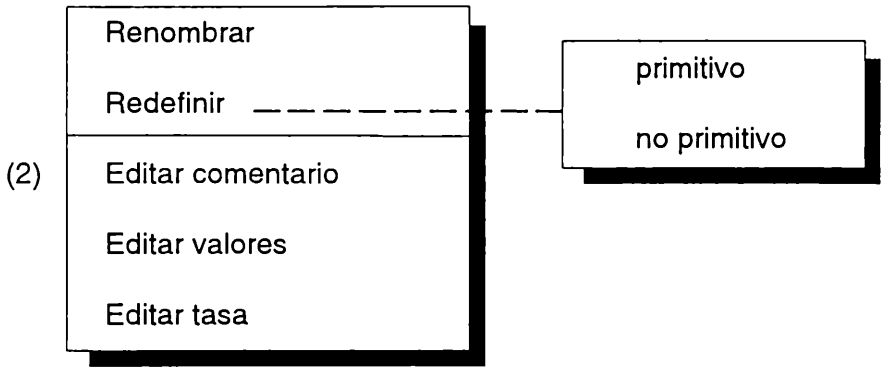
(2)



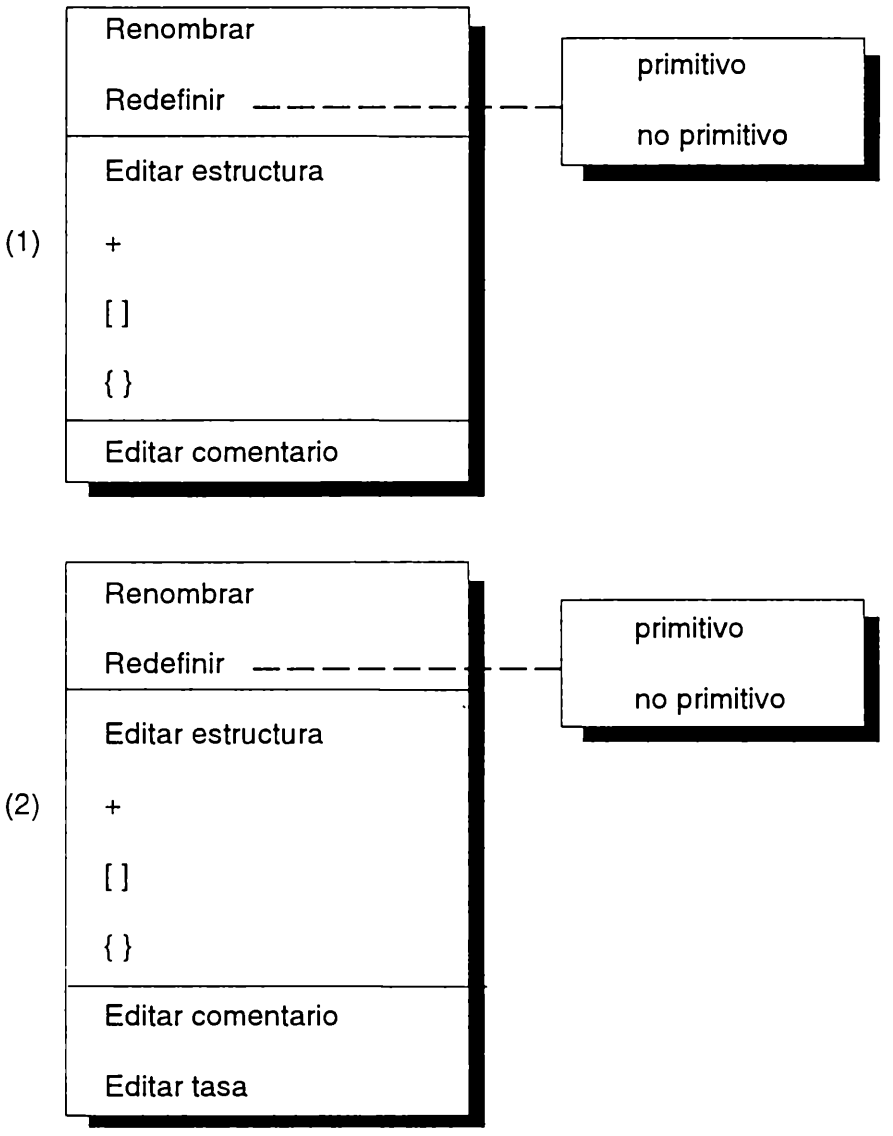
Asociado a un flujo primitivo de control (según sea interno -1- o externo -2-):

(1)





Asociado a un flujo no primitivo (según sea interno -1- o externo -2-):



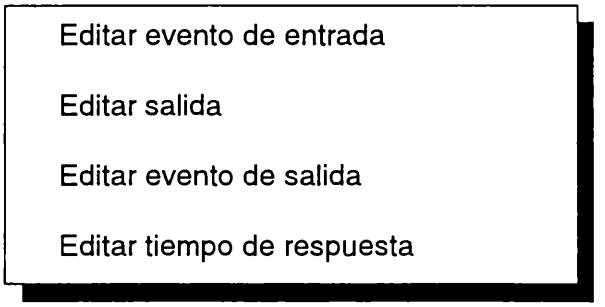
La opción **Renombrar** permite modificar el nombre del flujo. Presenta una ventana para ingresar el nuevo nombre del mismo.

Las opciones **Definir** y **Redefinir** presentan un menú para definir o redefinir el tipo del flujo: **primitivo** o **no primitivo.**

Las opciones **Editar...** presentan una ventana para ingresar el nuevo comentario o valor de atributo, según corresponda.

Las opciones +, [I] y { } permiten definir la estructura de un flujo no primitivo. Al seleccionar una de estas opciones, se obtiene una estructura modelo, donde las partes indefinidas de la estructura aparecen en mayúsculas y deben ser reemplazadas. La estructura definida es validada; filas asociadas a nuevos elementos de información incluidos en la estructura, son agregadas al diccionario. (La validación de la estructura de un flujo no primitivo será implementada en una versión posterior del ambiente.)

Durante la edición de la ETR, la fila (o subfila) seleccionada puede ser modificada. Si se presiona el botón derecho del mouse sobre el pane de texto, se desplegará el menú asociado a la fila (o subfila) seleccionada:



Cada una de las opciones presenta una ventana para ingresar un nuevo valor.

Testeo de una componente tabular.

Durante su testeo individual, toda componente tabular presentará la opción Editar en su correspondiente menú. El comportamiento de esta opción es análogo al de la correspondiente función sobre componentes gráficas.

Las filas incompletas de toda componente tabular pueden ser fácilmente completadas durante el testeo (individual o colectivo). Cuando una fila incompleta está seleccionada, si se presiona el botón derecho del mouse sobre el pane de texto, se desplegará un menú que incluirá: la opción Completar, junto con aquellas opciones que permitan modificar los atributos de la fila que ya han sido definidos. La opción Completar presentará, a su vez, un menú que incluirá sólo aquellas opciones que permitan completar los atributos de la fila que aún no han sido definidos.

D) Cuadro de panorama.

Este cuadro muestra la componente de la especificación seleccionada en el cuadro de la jerarquía, en forma simplificada y a escala reducida. Por ejemplo, sólo los terminadores, procesos y bancos de una componente gráfica son representados (por simples rectángulos, círculos y líneas paralelas horizontales); permitiendo de este modo visualizar su disposición colectiva en el diagrama. (Será implementado en una versión posterior del ambiente.)

6.3 Presente versión del ambiente.

Se han implementado las funciones del ambiente que permiten representar componentes de la especificación de requerimientos de todo tipo (gráfico, textual y tabular). Sin embargo, sólo algunas de las componentes pueden ser contruidas con la presente versión del ambiente.

De la estructura de proceso, se pueden representar componentes de nivel 0 (el DCD), nivel 1 (el DFD 0) y nivel 2 (las EPs de los procesos del DFD 0); además de una componente interna, característica de la metodología elegida, que combina los dos primeros niveles de componentes (el DFD combinado).

La representación de la estructura de control se omite a pesar de su relevancia. Su implementación no debería revestir mayores dificultades, pues la funcionalidad del ambiente para construir sus componentes gráficas (DFC combinado, DCC, DFCs y DTEs) y tabulares (TDs y TAPs), como se dijo anteriormente, ha sido desarrollada. En cambio, se hizo hincapié en el desarrollo de la ETR, considerada de interés primordial en la especificación de los requerimientos de un sistema de tiempo real.

Finalmente, se puede representar el diccionario de requerimientos, que cumple un rol vital en la especificación, pues contiene definiciones precisas de todos los flujos.

Sólo algunos de los servicios descriptos fueron implementados en la presente versión del ambiente. Se priorizaron las operaciones de edición y testeo (dinámico y estático, individual y colectivo) por sobre la impresión (que de todos modos se permite para cada componente en forma individual). Por último, la ayuda dependiente del contexto no se implementó. Se sugiere en razón de ello, cuando la ayuda sea necesaria, consultar el presente capítulo donde la funcionalidad del prototipo es descripta en su totalidad.

6.4 Referencias bibliográficas.

[DIG88] Digitalk Inc., "Smalltalk/V 286. Tutorial and Programming Handbook", Los Angeles, Palo Alto Research Center, 1988.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.

Capítulo 7: Etapas restantes.

La evolución hacia un hardware de mayor capacidad con menor costo, permite implementar sistemas de tiempo real de tamaño y complejidad crecientes. Se hacen necesarios métodos de especificación formal que definan los requerimientos funcionales y la estructura física de estos sistemas. Métodos que reflejen exactamente la naturaleza jerárquica de los sistemas.

Con este propósito, Hatley y Pirbhai [HAT88] introducen los "métodos estructurados", herramientas y técnicas de modelización usadas durante el proceso de especifición. El uso de estos métodos permite definir al sistema y a sus componentes de hardware y software, capturando:

1.- qué problema resolverá el sistema (los requerimientos del sistema),

2.- cómo será estructurado el sistema (la arquitectura del sistema), y

3.- cómo será implementada la arquitectura del sistema (los requerimientos y la arquitectura del hardware y del software).

Especificación de los requerimientos del sistema.

A lo largo de este informe, se presentaron las herramientas (diagramas de flujo, especificaciones y diccionario de requerimientos) y la metodología para especificar requerimientos, con el objeto de caracterizar un ambiente de especificación de requerimientos de sistemas de tiempo real.

Especificación de la arquitectura del sistema.

La especificación de la arquitectura del sistema (usando diagramas, especificaciones textuales y un diccionario) define: las entidades físicas que componen el sistema (módulos), el flujo de información entre ellas, y los canales a través de los cuales esta información fluye. Su propósito es asignar a los módulos: los requerimientos funcionales (definidos en la especificación de requerimientos del sistema) junto con requerimientos no funcionales asociados a las interfaces físicas involucradas.

Especificación de requerimientos y arquitectura de hardware y software.

Los requerimientos asignados a cada módulo, serán implementados en hardware o en software. Inicialmente se deberán particionar dichos requerimientos entre requerimientos de hardware y requerimientos de software. Los requerimientos de software deben ser evaluados para determinar además los requerimientos del hardware sobre el que correrá el software. Se desarrollan las arquitecturas del hardware y del software, asignando los requerimientos identificados a módulos físicos y lógicos respectivamente. Además, se asignan requerimientos adicionales asociados a las interfaces: en el caso del hardware, asociados a la interface de un módulo con otros módulos y con el ambiente del sistema; en el caso del software, asociados a su interface con el hardware.

Las especificaciones de las arquitecturas del hardware y del software permitirán construir el hardware y escribir el software respectivamente. Una vez implementado, cada módulo es

testeado e integrado en un subsistema junto con otros módulos. Cada subsistema es testeado e integrado en el sistema junto con otros subsistemas. Finalmente, el sistema es testeado y entregado al cliente.

Ciertos ambientes de especificación, desarrollados en este nivel, permiten: definir actividades de cálculo y flujos de información entre las mismas (requerimientos de software), definir procesadores y canales de comunicación entre los mismos (arquitectura de hardware), asignar actividades de cálculo a procesadores y flujos a canales (en forma manual o automática con la ayuda de heurísticas de optimización), y generar automáticamente un programa de aplicación

- para efectuar una simulación de la máquina destino (aquella cuyos procesadores y canales de comunicación fueron definidos) sobre una máquina de simulación (enfoque adoptado por el ambiente STER [CEN91]), o

- para ser ejecutado en tiempo real sobre la máquina destino (enfoque adoptado por el ambiente SYNDEX [LAV89]).

El desarrollo de un ambiente para especificar totalmente un sistema y sus componentes de hardware y de software, es en definitiva el objetivo hacia el que se orienta el proyecto. El presente trabajo pretende contribuir a tal fin como una primera etapa.

7.1 Referencias bibliográficas.

[CEN91] Centro Tecnológico para Informática, "STER. Manual de referência", Nova Friburgo, Laboratório de Tecnologia de Desenvolvimento de Software, 1991.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[LAV89] Lavarenne, Christophe y Sorel, Yves, "SYNDEX. Un environnement de programmation pour multi-processeur de traitement du signal. Manuel de l'utilisateur. Version v.0", Le Chesnay, Institut National de Recherche en Informatique et en Automatique, 1989.

Referencias bibliográficas.

[CEN91] Centro Tecnológico para Informática, "STER. Manual de referência", Nova Friburgo, Laboratório de Tecnologia de Desenvolvimento de Software, 1991.

[DEG91] De Giusti, Armando E., "Descripción y Validación de Hardware. Aplicaciones de Tiempo Real", Rio de Janeiro, V EBAI, 1991.

[DEM78] DeMarco, Tom, "Structured Analysis and System Specification", New York, Yourdon, 1978.

[DIG88] Digitalk Inc., "Smalltalk/V 286. Tutorial and Programming Handbook", Los Angeles, Palo Alto Research Center, 1988.

[HAT88] Hatley, Derek J. y Pirbhai, Imtiaz A., "Strategies for Real-Time System Specification", New York, Dorset House, 1988.

[LAV89] Lavarenne, Christophe y Sorel, Yves, "SYNDEX. Un environnement de programmation pour multi-processeur de traitement du signal. Manuel de l'utilisateur. Version v.0", Le Chesnay, Institut National de Recherche en Informatique et en Automatique, 1989.

[MAG86] Magalhães, Maurício Ferreira, "Software para tempo real", Campinas, UNICAMP, 1986.

[NIE90] Nielsen, Kjell W., "Ada in Distributed Real-Time Systems", New York, McGraw-Hill, 1990.

[PET84] Peterson, James Lyle y Silverschatz, Abraham, "Operating System Concepts", Massachusetts, Addison- Wesley, 1984.

[SHU92] Shumate, Kenneth C. y Keller, Marilyn M., "Software Specification and Design: A Disciplined Approach for Real-Time Systems", John Wiley, 1992.