# Entorno de Programación Ada

JOSE MANUEL DE LA RIVA GRANDAL, Tte. E.T.S. y
FRANCISCO JAVIER GOMEZ DE TERREROS, Cte. Médico del Ejército del Aire

## INTRODUCCION

Se entiende por Entorno de Programación al conjunto de herramientas diseñadas para facilitar el diseño, la construcción y la verificación de programas. Las herramientas que integran los entornos de programación tienen como objetivos ayudar a la producción industrial de programas, en el sentido de obtener productos eficientes, robustos y fiables, elaborados con el menor costo posible y en el tiempo más corto.

Desde un punto de vista estructural, las herramientas que componen un entorno de programación, se integran en un nivel superior al correspondiente a las herramientas clásicas (editores, depuradores, optimizadores, etc.).

En el caso más general el usuario del entorno de programación dispone de los siguientes tipos de herramientas:

- Herramientas de ayuda a la especificación del problema.
- Herramientas de ayuda a la verificación y control de la especificación.
- Herramientas de ayuda a la construcción del programa.
- Herramientas de ayuda a la verificación del programa.
- Herramientas de ayuda a la detección de errores de ejecución.
- Herramientas de ayuda al mantenimiento y evolución y mejora de programas.
- Herramientas de ayuda a la edición y documentación del programa.
- Herramientas de ayuda a la gestión y dirección de proyectos.

## ENTORNO Ada: APSE

El informe STONEMAN (febrero 80) definió los entornos y soportes del sistema anejo al lenguaje Ada. (fig. 1).

Nivel 0

Está constituido por el fisical de la máquina y por el sistema operativo.

Nivel 1

Está formado por el KAPSE (Kernel Programming Support Enviro) y sus funciones son tres:

a) Soporte para ejecución de programas en Ada.
b) Biblioteca de compilación separada.
c) Facilidades de depuración.

Nivel 2

Está constituido por el MAPSE (Minimal Programming Support Enviro) que es un conjunto mínimo de herramientas escritas en Ada y soportadas por el KAPSE que son necesarias y suficientes para el desarrollo y soporte de programas en Ada.

Dichas herramientas son:

- Editor
- Compilador de Ada
- Montador
- Depurador
- Base de Datos de utilidad
- Intérprete de comandos
- Biblioteca de utilidad

Los niveles 1 y 2 son internos al APSE (Ada Programming Support Enviro) el cual se ha creado para dar soporte a las aplicaciones particulares del usuario.

Los objetivos básicos del APSE son:

1) Un entorno para el soporte.
2) Entorno abierto, ampliable a las nuevas tecnologías.
3) Soporte de ejecución de programas Ada.
4) Control de configuraciones.
5) Soporte para proyectos de gran envergadura.
6) Transportabilidad de:
   - herramientas auxiliares,
   - objetos,
   - máquina de desarrollo,
   - programadores.

SINGULARIDADES Ada (figura 2)

**Paquete:**

Es la unidad básica del lenguaje Ada. Permite agrupar conjuntos de entidades, tipos, objetos y subprogramas. Su estructura está dividida en dos partes: (fig. 3).

- La especificación del paquete.
- El cuerpo del paquete.

La especificación agrupa todas las entidades que el paquete exporta y está desglosado en una parte visible, que es accesible al usuario y una parte privada, no accesible al usuario.

El cuerpo contiene los detalles de implementación, los cuales pueden estar ocultos para el usuario. Está formado por una parte declarativa y por una parte de inicialización.

La parte declarativa comprende la zona que agrupa los objetos propias del paquete y la zona que contiene la realización concreta de las unidades de programas.

La parte de inicialización es la que se ejecuta en el momento de la elaboración del paquete.

Es interesante hacer hincapié que mientras que todos los paquetes llevan la parte de especificación, no es obligatorio que tengan cuerpo y ambos permiten una compilación separada incluso una programación de analistas diferentes.

**Diseño modular:**

Está comprobado que las empresas gastan entre un 50%, y un 80% del presupuesto del proceso de datos en el mantenimiento y actualización del software. La reparación y mejora del software se suele realizar habitualmente, pero el mantenimiento preventivo suele hacerse esporádicamente.

Los módulos al igual que los bloques son grupos de instrucciones y/o datos que tienen unas características comunes y actúan como una unidad en el programa. El diseño modular permite utilizar la

técnica de Refinamiento Progresivo descomponiendo un problema complejo en un conjunto de problemas de menor grado de complejidad, a los cuales se les puede seguir aplicando la técnica de desglose hasta llegar a un subproblema de resolución obvia aplicando las primitivas del lenguaje.

El dividir un programa en módulos permite el que cada módulo pueda ser probado y depurado separadamente, y posteriormente agrupados en una estructura de programa único que inevitablemente será más fiable y de más fácil mantenimiento.

Mediante la metodología modular que aporta el lenguaje Ada, se facilita la modificación de programas, por extensión de la aplicación existente o variaciones en el Hardware, así como la captura y eliminación de errores.

**Lenguaje orientado al objeto:**

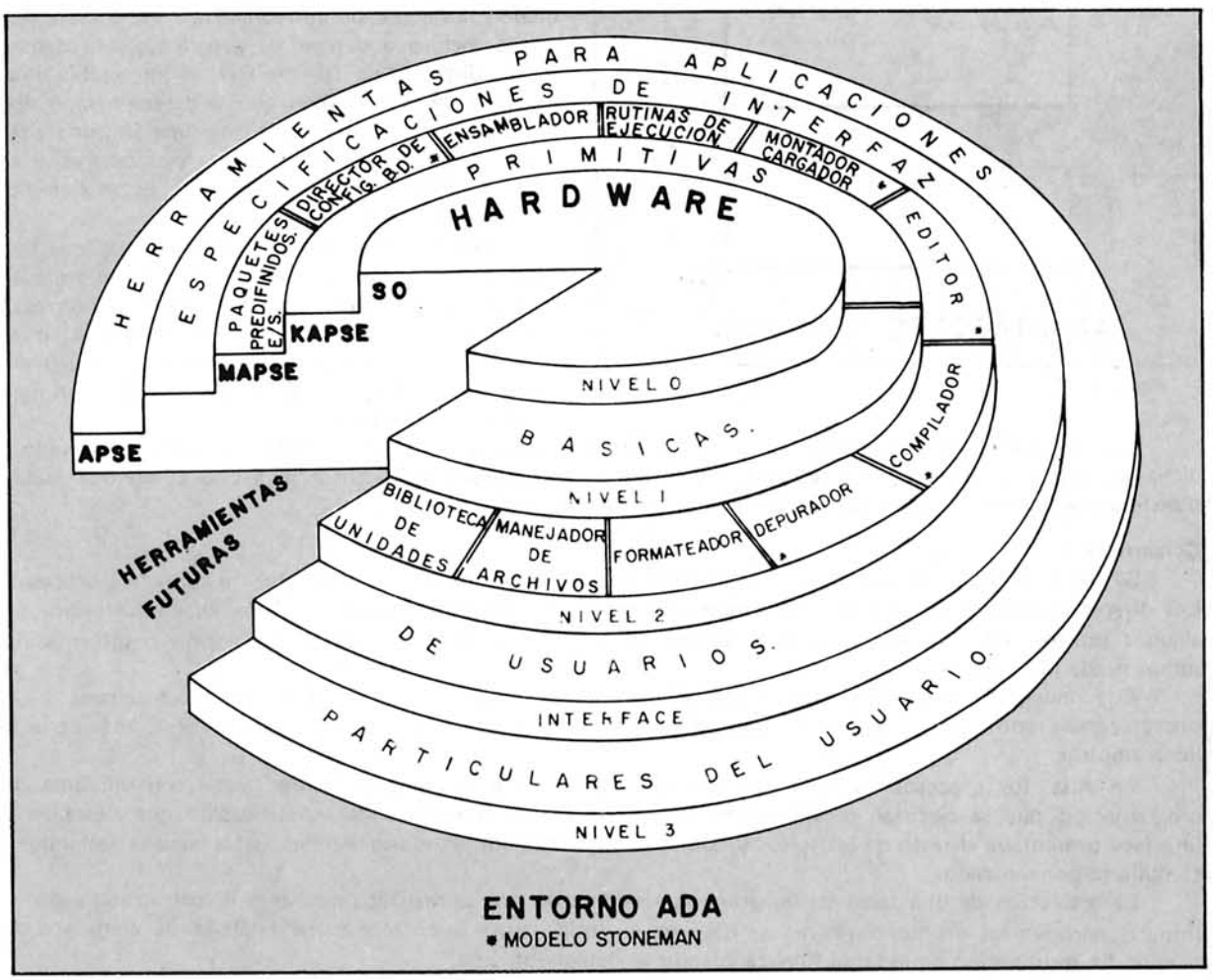
Ada es un lenguaje algorítmico y por tanto imperativo, como el Fortran, pero distinto de los lenguajes funcionales o descriptivos como el Lips o Prolog. Sin embargo, el Ada nos induce hacia los sistemas notacionales por lo que también podría decirse que es un lenguaje orientado hacia el objeto.

La introducción de la metodología orientada a las propiedades del objeto lógico nace en el Centro de Computación de Oslo con la creación del Simula-67, el cual permite a los usuarios crear sistemas orientados a objetos, pero utiliza el lenguaje Algol, prototipo de orientación a procedimientos y datos, para proporcionar números, valores lógicos, estructura de datos y estructuras de control.

**Tipos abstractos de datos:**

Es un concepto de importancia creciente en la metodología de programación. Se trata de pasar de la

▼ **Figura 1**



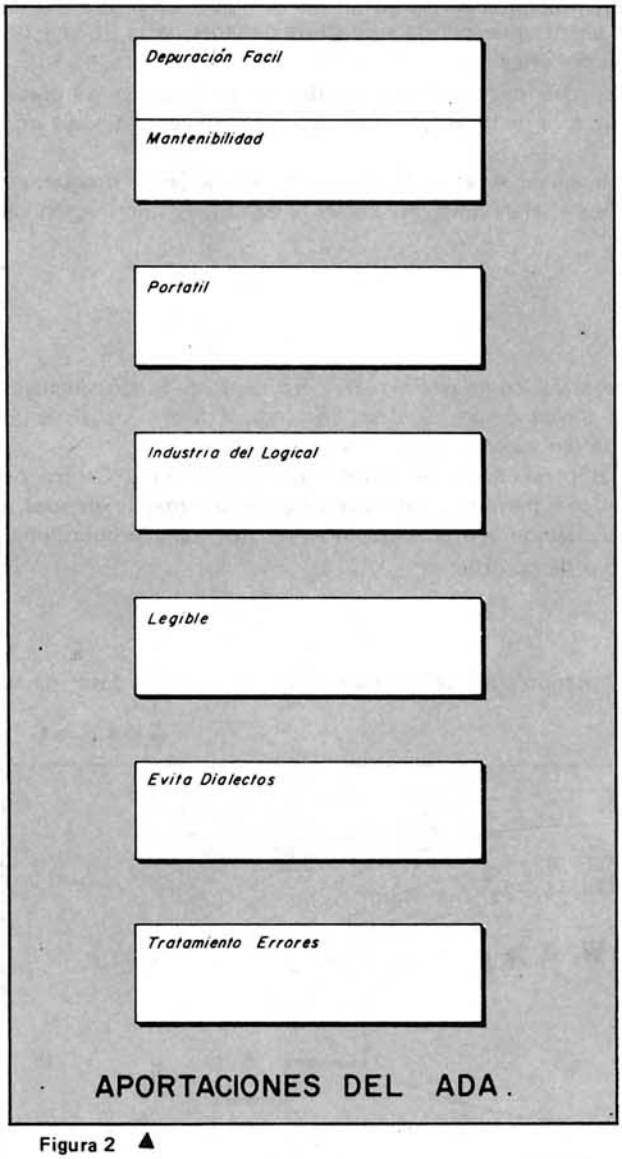



Figura 2 ▲

idea de objeto simple al objeto complejo o lógico definido por su identificador, un conjunto de atributos que lo caracterizan, junto con operaciones que lo manipulan y que esconden los detalles de esta estructura al resto del programa, es decir, los objetos lógicos consisten en una agrupación de información y una descripción de su manejo.

Los tipos abstractos de datos constituyen la mejor herramienta para el diseño, especificación, implementación y verificación de estructuras de datos y es un concepto potente para luchar contra los dialectos del lenguaje, pues hace a éste extensible o ampliable internamente sin necesidad de definir explícitamente nuevas estructuras a su sintaxis.

Ada está en dicha línea y permite librerías no sólo de subrutinas, sino también de objetos complejos o estructuras avanzadas como pilas, colas, etc.

**Excepciones:**

Las excepciones son el resultado normalmente de un conjunto de circunstancias desgraciadas que pueden o no representar un error. Esas situaciones se pueden dar en tiempo de ejecución impidiendo llevar a cabo correctamente una acción, llegando incluso a detener el programa, suceso que en los Sistemas de Tiempo Real es intolerable. Por lo tanto son condiciones que se desean excluir del procesamiento normal. Una interrupción por Hardware, especialmente las causadas por errores aritméticos, como la división por cero, es un ejemplo perfecto de excepción.

Ada ha intentado la solución al problema del manejo de excepciones, aportando para ello mecanismos que detectan y tratan la excepción (error), sin abortar el programa ejecutado. Para ello en el final de cada función o procedure e incluso en otros puntos del programa se puede definir un manejador de excepciones.

Cuando aparece una excepción en un punto del programa, se transfiere el control al ejecutor asociado a dicha excepción previamente especificado. Si no se ha especificado el ejecutor explícito el control puede transferirse a un ejecutor de excepciones definido en la realización.

**Concurrencia:**

Se define el proceso como la unidad de programa capaz de operar concurrentemente con otros procesos. Los diferentes procesos en que se estructura una determinada aplicación necesitan intercambiar información en algunos puntos determinados y también necesitan detenerse a la espera de alguna acción o información suministrada por otro proceso.

Ada incorpora mecanismos de concurrencia que permiten el intercambio de informaciones y la sincronización entre procesos que se desarrollan en paralelo, los cuales pueden comunicarse, retardarse o interrumpirse.

En Ada, los procesos concurrentes que se sincronizan se denominan Tareas. Estas son unidades de programación que se ejecutan concurrentemente y constan de una parte de especificación que describe la interface presentada al resto de las tareas y que es desconocida por el usuario. Ambas partes pueden declararse y compilarse por separado.

La activación de una tarea en las primeras versiones de Ada se realizaba mediante la instrucción especial Initiate, pero en las últimas versiones se hace de un modo automático al ejecutar el Begin de comienzo de bloque. La terminación de la tarea finaliza cuando se detecta un End.

Ada utiliza un mecanismo conocido como Rendezvous (RV) sugerido por Hoore, el cual trata a cada tarea como un proceso secuencial de comunicación. Las tareas se sincronizan en el tiempo y en el espacio cuando se comunican y sus características esenciales son:

- espera simétrica,
- una única transición en cada RV,
- exclusión mutua de la transición,
- intercambio de información en los dos sentidos.

**Compilación separada:**

En Ada una unidad de compilación puede ser un procedure, una función o un paquete y pueden ser o no declarados dentro de otra unidad de compilación. La compilación separada permite que varios programadores puedan desarrollar código independientemente, conociendo sólo los nombres y parámetros de los subprogramas de los otros desarrollos. Cualquier cambio efectuado más tarde en los subprogramas requerirá solamente que los segmentos dependientes sean recompilados y el programa total sea relinkado con el mínimo de depuración.

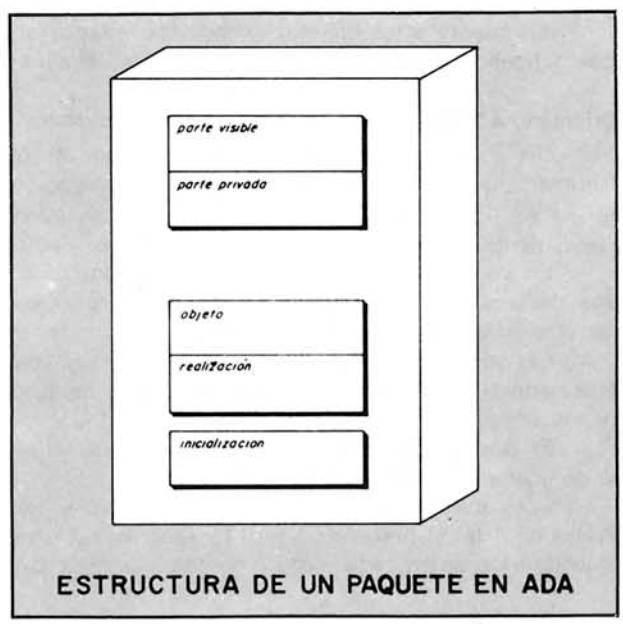
El inconveniente que incorpora Ada a diferencia de otros lenguajes como el Pascal o Modula-2 es que no permite pasar procedimientos ni funciones como parámetro.

**Código óptimo:**

El código generado por los compiladores suele ser muy deficiente, debido a que utiliza reglas muy generales para llevar a cabo la generación de código. Los compiladores actuales como el de Ada van dotados de una etapa de optimización de código, la cual actúa a dos niveles, local y global.

En la optimización local se examinan zonas o ranuras del programa objeto para detectar estructuras elementales que sean susceptibles de ser sustituidas por otras más eficientes utilizando para ello tablas de patrones o juegos de registros.

La optimización global trata el código generado en su conjunto, examinando macroestructuras tales como bucles, procedures, subexpresiones, etc., posteriormente elimina y depura todo aquello que sea redundante.




▲ Figura 3

**Portabilidad:**

El lenguaje Ada ha sido diseñado para que el compilador genere un código portátil, es decir, código fácilmente transferible a máquinas distintas y por tanto con diferente código máquina.

Se ha tratado de que el mínimo no portátil sea fácilmente identificable en un entorno de programación como paquetes que dependan del Hardware, los cuales serán necesarios reescribirlos ante cualquier variación de la máquina destino.

**Genéricos:**

Los lenguajes actuales no facilitan la posibilidad de que un único subprograma fuese utilizado por un conjunto de datos de Tipos diferentes, con lo cual problemas genéricamente idénticos tenían que ser implementados en subprogramas diferentes.

La genericidad en Ada permite la escritura de subprogramas independientes de la estructura de los datos tratados. Por lo tanto, este concepto potencia la noción de empaquetado, puesto que un mismo subprograma puede ser utilizado en datos de diferentes Tipos.

La diferencia entre un subprograma y una unidad genérica en Ada es que el primero implica una parametrización en el tiempo de ejecución, mientras que en los genéricos la parametrización se realiza en tiempo de compilación. En Ada pueden ser unidades genéricas los paquetes y subprogramas cuya estructura contiene una parte genérica en donde se incluyen las especificaciones de los parámetros formales, y de un cuerpo genérico.

**Especificación de representación:**

Ada es un lenguaje que puede ser utilizado para la realización de programas de Sistemas. Cuenta con unos recursos para controlar la especificación de la representación, acceder a niveles cercanos a máquina y realizar interfaces con otros lenguajes de alto nivel tales como el Pascal y el C.

El control de la representación de los datos puede realizarse de dos modos:

a) Definiendo las propiedades lógicas y formulando los algoritmos de estas propiedades.

b) Definiendo la representación física de los datos.

Ada puede controlar una especificación de representación por su longitud a través de los tipos enumerativos o de registros o por dirección.

**Estandarización:**

El principal objetivo del nuevo lenguaje es el de reducir el costo del Logical. En el caso del Ada se abrió en su día un período de consultas a todos los expertos de ingeniería de Software del mundo, recogiéndose sus sugerencias, de modo que el futuro lenguaje careciese de sombras sintáctico-semánticas susceptibles de generar en el futuro dialecto o subconjuntos del lenguaje.

La experiencia nos ha enseñado que no es posible caracterizar eficientemente un lenguaje, preveyendo de antemano su futura extensión, por lo tanto con el fin de salvar este problema se dotó a Ada de recursos suficientes y potentes para que el programador creara las estructuras de datos avanzadas necesarias para sus proyectos propios. De esta forma el lenguaje incorpora el concepto de extensión interna sin necesidad de salirse del conjunto de estamentos definidos.

**Paquetes predefinidos (E/S):**

El tratamiento de los ficheros en Ada es relativamente pobre, debido a que los diseñadores del lenguaje consideraron que la entrada y la salida no era un concepto implícito del lenguaje, sino de los usuarios.

Ada distingue dos tipos de entrada/salida: la binaria (Real, Write, etc.) y la de textos (Get, Put, etc.),

En cuanto a los ficheros existen los internos que los declara y manipula el programador y los externos que definen el soporte físico y se pueden asociar a un fichero interno en la apertura del fichero.

**Orientado a Sistemas Empotrados (Embeddod systems):**

Un Sistema Visible es aquel en el que el ordenador está en evidencia y puede ser identificado. Normalmente son de propósito general y la aplicación está determinada por un programa en Software o una rutina en Firmware. ( ). La función aplicación puede ser cambiada por el usuario o por el implementador rápidamente borrando funciones ya existentes o añadiendo nuevas funciones.

En contraposición en los Sistemas Empotrados el computador no es fácilmente identificable y sólo es una parte de un todo. Suele ser de propósito especial con el último programa en Firmware después una depuración del Software inicial y suelen tener limitaciones de peso, espacio, potencia y medio ambiente.

Las aplicaciones más usuales de los sistemas empotrados son las militares, tales como control de fuego, procesadores de radar, funciones especiales de navegación, procesamiento de señales, instrumentación, control de motores, etc., etc.

El DoD y USAF dictaron normas para que todo el Software futuro orientado a los Sistemas Empotrados se programe en Ada MIL-STD 1815.

Hoy el esfuerzo del DoD es más amplio y tiende a mejorar la tecnología del Software a través del Programa Ada, el programa STARTS (Software Technology for Adaptable Reliable Systems) y el Instituto de Ingeniería del Software fundado el pasado año en la Universidad Carnegie Mellon, Pittsburgh (USA).

## ESTADO ACTUAL DE LOS ENTORNOS DE PROGRAMACION Ada

Actualmente el DoD de los EE.UU. está financiando dos proyectos de construcción de entornos Ada, el ALS (Ada language Systems) construido por la compañía Softeh y el AIE (Ada Integrated Environment), desarrollado por Intermetrics. El primero está orientado hacia máquinas tipo VAX sobre VMS, y el segundo hacia máquinas I.B.M. Un proyecto de dimensiones más modestas pero también financiado por el DoD, es el ARCTURUS desarrollado en Irvine por la Universidad de California, para máquinas tipo VAX y sistema operativo UNIX. Como novedad incorpora un editor dirigido por la sistasis de Ada que maneja colores para diferenciar las distintas partes sintácticas del texto fuente, y ofrece la posibilidad de compilar o interpretar los programas Ada. ■

BIBLIOGRAFIA


1) Data General to offer Rolm Ada, Electronic News. 1983.
2) A new Ada compiler for Z-80 systems, Microsystems, 1982.
3) Concurrency in Ada and multicomputers computer language. Procedings of the Ada. International conference 1985. Cambridge University.