UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

La Importancia de los Lenguajes de Cuarta Generación en el Desarrollo de Sistemas

T E S I S
Que para Obtener el Título de
M A T E M A T I C A
P R E S E N T A

ACT. ALEJANDRA MENDEZ MENDOZA

MEXICO, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

INTRODUCCION

En estos años de transición, los usuarios de los equipos de cómputo están exigiendo el desarrollo de herramientas de programación que les permita realizar su trabajo de manera rápida, fácil y eficiente. Es por esto que se están experimentando cambios radicales en los lenguajes de programación.

Se necesitan lenguajes que incrementen la productividad de aquellos que se dedican al desarrollo de sistemas de información así como facilidades para el usuario final.

También se habla de otros requerimientos que deben cumplir dichos lenguajes, a saber:

a) Reducción del ciclo de desarrollo de una aplicación.
b) Posibilidad de dar mantenimiento en base a cambios de políticas o requerimientos por parte de los usuarios, en una forma sencilla y eficaz.
c) Reducción de costos de mantenimiento.

Pués bien, ahora se cuenta con herramientas de programación que satisfacen todas estas necesidades.

Estas herramientas son a menudo conocidas como **LENGUAJES DE CUARTA GENERACION (4GL), LENGUAJES DE ALTA PRODUCTIVIDAD O GENERADORES DE APLICACIONES.**

Los LENGUAJES DE CUARTA GENERACION tiene principios básicos sobre los cuales actúan, estos son:

MINIMIZAR LA CARGA DE TRABAJO

CONOCIMIENTOS MINIMOS EN CUANTO HERRAMIENTAS AUXILIARES

MINIMIZAR LA COMPLEJIDAD DEL DESARROLLO

MINIMIZAR EL TIEMPO INVERTIDO EN EL DESARROLLO DE LA APLICACION

MINIMIZAR LA CANTIDAD DE ERRORES COMETIDAS POR EL PROGRAMADOR

MINIMIZAR LA LABOR DE MANTENIMIENTO

MAXIMIZAR LOS RESULTADOS

Como se puede apreciar los 4GL son una herramienta que facilita en gran escala el desarrollo de sistemas de información.

El objetivo de esta tesis es dar a conocer en una forma general las características básicas de los **LENGUAJES DE CUARTA GENERACION** para que de esta forma se pueda apreciar las ventajas que se pueden obtener con el uso de estos lenguajes.

C A P I T U L O I

Desarrollo histórico de los lenguajes de programación.

El presente capítulo tiene como objetivo dar una breve descripción del desarrollo histórico de los lenguajes de programación.
Es bien sabido que la tecnología ha ido cambiando radicalmente en todos y cada uno de los aspectos relacionados con el hombre.
Entre éstos se encuentra uno muy importante, ya que ha venido a revolucionar en gran escala la forma de realizar actividades en diferentes áreas, éste es la **computadora**.
El avance que ha tenido el desarrollo de las computadoras es realmente notable, pués en relativamente pocos años, ha sufrido cambios radicales, que van desde su tamaño y capacidad hasta la aceptación por parte de los usuarios, la cual se ha incrementado en magnitud. Esta aceptación depende en gran escala de los lenguajes que se utilizan para el manejo de las computadoras, pués entre más sencillos, amigables y eficientes sean, más personas aceptarán a la computadora como una herramienta para su trabajo.

1.1 La necesidad de tener lenguajes de programación

La computadora electrónica revolucionó increiblemente la velocidad del procesamiento de datos, pero dejó una tarea muy difícil al hombre: programar los problemas a resolver. Para ello hubo la necesidad de crear algún modo de comunicarse con la computadora, dando origen así a los llamados **"lenguajes de programación"**.

Así pués, cualquier persona que tenga como herramienta de trabajo a una computadora emplea alguna forma de lenguaje de computadora, y por lo tanto cualquier estudio de los lenguajes es

necesario para el entendimiento de un sistema de computación. Los lenguajes usados para programar son creaciones artificiales, construidas por el hombre y usadas por él mismo. Sin embargo, el diseñar un lenguaje no dá tantas libertades como pudiera pensarse, es decir no hay la posibilidad de crear un lenguaje de cualquier tipo, de ser esto posible, el lenguaje natural sería considerado por algunos como el ideal, aunque para otros resulta no ser apropiado para programar, esto porque la precisión necesaria para comunicarse con la computadora es mucho mayor que la utilizada en la comunicación diaria entre los hombres.

Hay una gran variedad de lenguajes de computadora, conocidos también como lenguajes de programación, ordenados en extensión desde un simple operador de comandos hasta un sistema completo para alguna aplicación en especial.

A pesar de esta diversidad, todos los lenguajes de programación comparten un conjunto de conceptos similares.

Cada lenguaje de programación tiene dos partes importantes. Primero, hay un conjunto de reglas las cuales nos permiten decidir si un texto está o no bien "formado", es decir si puede o no ser considerado un programa. La otra es un conjunto de reglas de interpretación, que relacionan la ejecución de un programa con el comportamiento de una máquina.

Estas dos partes conocidas como sintaxis y semántica, respectivamente, especifican completamente un lenguaje de programación.

Existen diversas clasificaciones para los lenguajes de programación de acuerdo a ciertas características, como pueden ser el tipo de usuario al que se destinan, el desarrollo cronológico de los mismos, etcétera.

Para este trabajo se clasifican en orden cronológico como sigue:

Lenguajes de Primera Generación
Lenguajes de Segunda Generación
Lenguajes de Tercera Generación
Lenguajes de Cuarta Generación

1.2 Lenguajes de Primera Generación

Los lenguajes de primera generación, fueron los primeros en ser usados para la realización de programas, y son conocidos también como

Lenguajes de Máquina

Lenguaje de máquina

Fué el primero en aparecer y por mucho tiempo el único lenguaje utilizado debido al tipo de aplicaciones en que se ocupaban las computadoras.

Consiste en instrucciones que son ejecutadas directamente por la computadora, las cuales son elementales, formadas por un código binario de 0's y 1's, el cual se conoce como código de máquina.

Limitaciones de los lenguajes de máquina

El hecho de que el lenguaje de máquina esté basado en un código binario, hace de un problema muy sencillo de programar, una tarea que requiere mucha atención para todo detalle, pues en el transcurso de la programación, con sólo una leve distracción, puede ser fácil cometer un error, tal como el de escribir un 0 en lugar de un 1, en algún lugar de la instrucción cambiando con esto la dirección del dato. Aún más, el programador debe saber los códigos binarios de todas las instrucciones, y la exacta localización en memoria de los operandos, esto era una enorme labor pero la mayoría de los programadores veían a la computadora solamente como un herramienta que les resolviera sus problemas de tal forma que les distrajera lo menos posible ya que las aplicaciones en general trataban con problemas que implicaban una gran cantidad de cálculos.

Sin embargo el uso de computadoras, requería que los programadores conocieran a todo detalle la estructura y el funcionamiento de la computadora, e incluso los llevaba a involucrarse más en la programación que en la solución del problema.

1.3 Lenguajes de Segunda Generación

Dos cursos de acción pudieron haberse tomado para saltar el abismo entre el lenguaje usado por la máquina y el usado por el hombre, con el fin de resolver algún problema.

El primero fue adaptar el problema a la computadora, el cual llevaba el riesgo, si el programador no era cuidadoso, de modificar el objetivo del problema al tratar de simplificar el proceso.

El otro era adaptar el lenguaje de comunicación de la máquina a las necesidades del problema.

El primero era una forma de decir "cómo" se iba a resolver un problema, y el segundo preguntaba "cuál" era el problema, dejando a la computadora que investigara después de detallar el problema, cuál era la solución.

Los lenguajes para la comunicación hombre-computadora, intermediarios entre el "cómo" y el "cúal" fueron desarrollados durante la Segunda Generación de lenguajes y están en uso hoy en dia.

Los lenguajes de Segunda Generación son conocidos también como:

Lenguajes Ensamblador

Las caracteristicas de un lenguaje ensamblador son:

a) Se pueden usar mnemónicos tales como ST para STore, A o icluso ADD para Add o bien, M o MLY para Multiply.

b) Las constantes pueden ser escritas en base 10 (i.e. 67), en hexadecimal (i.e. 5A4) o bien en binario (i.e. 01011010).

c) Un programador puede hacer uso de variables.

d) Permite al programador escribir macroinstrucciones. Una macroinstrucción es un subprograma que se inserta al momento de la compilación del programa, quedando incluido en el programa al momento de ser ejecutado. Esta caracteristica es una mejora a los primeros lenguajes ensambladores.

Ahora bien como este no es un lenguaje "entendido" por la máquina, fué necesario crear un traductor, al cual se le llamó también **ensamblador**.

Un **ensamblador** es un programa que traduce código escrito en lenguaje ensamblador a lenguaje de máquina.

Un ensamblador puede producir un programa objeto, indicando errores de escritura, si hay alguno en el programa fuente, haciendo una lista de estos, también puede traducir las constantes dadas en cualquier formato (hexadecimal,decimal, etcétera.) a lenguaje de máquina y además asigna localidades de almacenamiento para cada instrucción y se asegura de que estén disponibles.

Además si se realizan pequeñas modificaciones, no requieren

cambios extensos en el programa. Esto debido al hecho de que el ensamblador automáticamente reasigna todas las direcciones y referencias a instrucciones, constantes y datos, eliminándose así problemas potenciales en la modificación de un programa.

Limitaciones de los Lenguajes Ensambladores

El arribo de estos lenguajes, trajo como resultado varias inovaciones, la más importante de todas, la programación automática.

Pero los lenguajes ensambladores sólo fueron un paso en el camino hacia los lenguajes de alto nivel.

Aún requieren que el programador realice una secuencia de instrucciones muy cercanas a las relacionadas con el lenguaje de máquina, no siendo de todo fieles a su esencia. Por esta razón algunos autores no reconocen a los lenguajes de máquina y ensamblador como lenguajes de programación.

Debido a la existencia de los lenguajes de alto nivel, tanto el lenguaje de máquina como el ensamblador han perdido importancia para los programadores y son poco usados, excepto cuando es necesario hacer el mejor uso posible de la estructura de una computadora.

Cabe hacer la aclaración de que aunque el lenguaje de máquina ya no es usado por los programadores, sigue siendo una de las herramientas indispensables para el manejo de una computadora, pués como se ha dicho, el lenguaje de máquina es el único

lenguaje que ésta "entiende".

1.4 Lenguajes de Tercera Generación

Estos lenguajes son también conocidos como lenguajes de alto nivel.

Los lenguajes de alto nivel son hoy en dia la herramienta de trabajo más importante de la mayoria de los programadores, teniendo como consecuencia el no hacer uso de los lenguajes de 1a. y 2a. generación.

Una de las razones por lo que el uso de las computadoras está en auge, es que las herramientas de programación han cambiado radicalmente, en comparación a las existentes en los inicios de la computación.

Una forma de dar las caracteristicas de los lenguajes de tercera generación es mencionar las diferencias existentes entre los lenguajes de 1a. y 2a. generación contra los de alto nivel, estas son:

1. No requieren que el usuario tenga amplios conocimientos en lo que respecta a conceptos de la computadora tales como registros, representación interna de los datos, etcétera.
2. Permiten la posibilidad de transferir programas de una computadora a otra. Es decir hay cierto grado de independencia entre una computadora y otra.

3. Permite que los programas sean escritos de manera más concisa que en lo que se refiere a los lenguajes de máquina e incluso en los de ensamblador.

4. Hace uso de programas que traducen a su vez programas escritos en lenguaje de programación a lenguaje de máquina, estos programas son conocidos como compiladores, los cuales son traductores más complejos que los ensambladores.

5. Los programas pueden ser escritos en términos orientados al problema en cuestión. Es decir permite dar nombres y símbolos a los datos y también la inclusión de expresiones y operadores matemáticos.

El desarrollo de los lenguajes de tercera generación, en algún momento fué motivado por el afán de crear lenguajes que tuvieran las características linguisticas y estructurales de un lenguaje natural.

El desarrollo de este tipo de lenguajes ha traido consigo avances en las técnicas de programación, incluyendo la facilidad para construir y manipular estructuras.

En el capitulo correspondiente a Lenguajes de Tercera Generación, se mencionarán los más importantes además de su contribución al desarrollo de otros.

1.8 Lenguajes de Cuarta Generación

En estos tiempos de transición los programadores necesitan lenguajes que sean manejables, eficientes y de fácil uso, además que permitan un alto grado de productividad.

Es pues, que los lenguajes de cuarta generacio'n surgen como la herramienta que remedia estas situaciones de una manera radical.

Con estos lenguajes se han ido los tiempos en los que resultaba tedioso el escribir largos programas.

También ayudan a que el usuario esté atento en el problema a resolver y con más posibilidad de resumir la información.

Los profesionales de la computación han obtenido varios beneficios con el uso de los lenguajes de cuarta generación. entre ellos se pueden citar:

-Su productividad se ve incrementada en gran escala.

-El uso de prototipos permite que se aprecien los resultados antes de tener el sistema listo.

-Los menùs son una caracteristica natural de los lenguajes de programación de cuarta generación.

-Se tienen un conjunto de opciones y páginas de ayuda.

Estas y otras caracteristicas se detallarán más en el capitulo correspondiente a lenguajes de cuarta generación.

1.6 Otra clasificación de los lenguajes de programación.

Otra posible clasificación de los lenguajes de programación de alto nivel puede ser

Categoria Cientifica
Categoria Comercial

Los efectos de esta clasificación van más allá de la aplicación en si y son tomados como una influencia en el diseño de máquinas y sistemas de programación.

Es de pensarse entonces, que este sentir también ha llegado a los diseñadores de lenguajes de programación dando como origen, diseños que varian notoriamente en contenido y estructura.

Aunque algunos lenguajes contienen un conjunto de facilidades y se pueden considerar como de "propósito general", las diferencias entre las aplicaciones cientificas y comerciales son importantes pues han tenido una gran influencia en el diseño de lenguajes.

1.8.1 Categoria Cientifica

Los primeros usuarios cientificos se derivan de la Fisica . Todos los usuarios vieron en la computadora una herramienta que les permitia incrementar poderosamente la rapidez de sus cálculos, comparándola con la calculadora electrónica, la cual habia sido su herramienta más fuerte.

Su objetivo principal era incrementar el rango de fuerza en la solución algoritmica de problemas numéricos.

Lo que con más frecuencia interesa a los cientificos involucra la solución numérica a conjuntos de ecuaciones que representan un modelo a los problemas que ellos trabajan.

Los programas para usos cientificos son caracterizados comúnmente por una gran cantidad de cálculos y relativamente una escasa relación de datos de entrada y salida.

1.8.2 Categoria Comercial

Los usuarios comerciales entraron al mundo de las computadoras teniendo como experiencia a las máquinas registradoras, y estando acostumbrados a procesar datos de almacenamiento, adquisiciones, clientes, todo esto sobre tarjetas a base de métodos manuales.

Aunque suena simple, algunos movimientos podian traer consigo operaciones engorrosas, así es que estos usuarios, si bien no estaban interesados en la capacidad numérica de la computadora, si lo estaban en la capacidad de almacenamiento.

reordenamiento e impresión de grandes volúmenes de datos altamente estructurados, con lo que llegó así el conocido "procesamiento de datos".

Tenemos entonces, que uno de los objetivos principales de las aplicaciones comerciales es el almacenamiento y actualización de información usada día a día en una empresa de negocios.

CAPITULO II

LENGUAJES DE TERCERA GENERACION

En el capitulo anterior se habló de los lenguajes de programación de acuerdo a su clasificación por generación, dando caracteristicas generales de los mismos.
En este capitulo se pretende dar a conocer las caracteristicas de los lenguajes conocidos como de tercera generación o 3GL, los cuales han sido la herramienta utilizada por la mayoria de los programadores, pues son considerados como fáciles y versátiles en su uso.
Se puede decir que estos lenguajes han sido el puente que ha permitido la "comunicación hombre-máquina" de una forma "natural y fácil" para los programadores. De aqui que cuenten con la aceptación de muchos de los profesionales de la computación.

2.1 Caracteristicas generales de los lenguajes de TERCERA GENERACION

Los lenguajes de alto nivel difieren de los lenguajes de primera y segunda generación en que los primeros pretenden:

-Aproximarse al lenguaje usado diariamente por el hombre.

-Liberarse por completo de las restricciones de la máquina, de modo que un programa pueda ser utilizado en cualquier máquina que pueda tener el compilador adecuado.

Los lenguajes de alto nivel tienen principios básicos comunes pero también hay muchas pequeñas diferencias, las cuales se originan por un crecimiento con conceptos básicos mal definidos o bien por un diseño incorrecto.

Para 1950, los problemas originados por las diferencias citadas, se trataron de resolver, pensando en el diseño de los lenguajes como un procesador común de muchos lenguajes de programación. Ahora bien, se ha dicho que los lenguajes de alto nivel tienen más cosas en común que diferencias.

Los diferentes tipos de aplicaciones han dado como origen el diseño de diferentes tipos de lenguajes de programación con marcadas diferencias, entre los que se pueden citar ALGOL, COBOL y FORTRAN.

2.2 Clasificación de los lenguajes de tercera generación

Los lenguajes de alto nivel son clasificados de diferentes formas.
Algunos autores los clasifican, por ejemplo de la siguiente manera:

1)Los lenguajes primitivos de alto nivel
2)Los lenguajes abstractos de alto nivel

Los lenguajes primitivos de alto nivel corresponden a los primeros en haberse diseñado y tienen la característica de estar estrechamente relacionados con las características de la máquina en la que se habían desarrollado. El ejemplo más común de éstos es FORTRAN.

Los lenguajes abstractos de alto nivel comprenden lenguajes que se pensaron independientes de las caracteristicas de la computadora. Todos los lenguajes de este tipo fueron desarrollados más o menos de acuerdo a la arquitectura de la máquina de Von Neumann, de ahi que algunos autores hagan referencia a estos lenguajes como los "Lenguajes de Von Newmann". Caracteristicos de este grupo se encuentran ALGOL, COBOL y PL/I.

Otra posible clasificación se maneja de acuerdo al concepto de estructura, como :

1. Lenguajes estructurados entre los que se cuentan ALGOL y PL/I.
2. Lenguajes no estructurados como son FORTRAN y COBOL.

2.3 Principales caracteristicas de los 3GL.

Los lenguajes de tercera generación revolucionaron en gran medida las herramientas de programación que habia hasta antes de su nacimiento. Estos lenguajes aportaron numerosas herramientas que hicieron de la programación una tarea mucho más sencilla. Estas son las que caracterizan a los lenguajes de tercera generación.

Entre estas características se pueden mencionar:

1. Hacen uso de identificadores que van desde símbolos aritméticos hasta una gran diversidad de comandos, así como los nombres de las variables que se utilizan dentro de los programas.
2. Se cuenta con la posibilidad de definir constantes.
3. Dentro de un programa se trabajan datos. Estos a su vez, dependiendo del contexto del problema a resolver pueden ser constantes (de las que ya se mencionó, se pueden definir como tales) o bien variables.

   En el caso de estas últimas las puede haber de diferentes tipos como son enteras, reales, caracteres, cadenas, apuntadores, booleanas, etcétera. Pués bien, los lenguajes de tercera generación son muy versátiles en este sentido, ya que se pueden definir variables con todos y cada uno de estos tipos, aún más, se pueden definir subintervalos o conjuntos de estos tipos, dependiendo claro está del lenguajes que se esté trabajando.
4. Se cuenta con proposiciones condicionales. En ocasiones es necesario hacer una selección entre dos o más formas de acción.
5. Si hay necesidad de repetir un proceso un número finito de veces, se puede hacer con el uso de los ciclos, que son parte de los lenguajes de tercera generación.
6. Se tienen funciones propias del lenguaje, como por ejemplo, raíz cuadrada de un número, el cuadrado de un número, módulo, parte entera de un real, etcétera.

También se permite que el programador defina sus propias funciones.

7. Se trabajan tipos estructurados. Se han mencionado los tipos de variables existentes en los lenguajes de tercera generación. Otros tipos muy importantes por su uso son los arreglos y los registros.

   Un arreglo es una colección de variables con el mismo tipo.

   Un registro es un conjunto de variables de diferentes tipos.

8. Se trabajan operadores aritméticos, relacionales y lógicos.
9. Los datos se pueden escribir y/o leer en el formato deseado.
10. Manejo dinámico de memoria.

Estas son tan solo unas cuantas características de los lenguajes de tercera generación.

A continuación se enuncian algunos ejemplos de los 3GL.

2.4 FORTRAN

(FORmula TRANslator)

Fué el primer lenguaje de alto nivel. uno de sus propósitos era el establecer la posibilidad de codificar sin ningún problema y con precisión exacta problemas matemáticos.

Antes de usar FORTRAN se había logrado esto pero con ciertas restricciones tales como:

1) Las expresiones algebraicas se analizaban separadamente.

2) El programa sólo podía involucrar 26 variables (las correspondientes a las letras que forman el alfabeto).

En los años 50's, el costo de programar era al menos igual al costo del equipo, por lo que el desarrollo de hardware se vió retrasado. La única forma de solucionar este problema, era considerar la idea de la "programación automática", desarrollando lenguajes que fueran fáciles de usar y a su vez eficientes.

El compilador FORTRAN fué el primero en producir un programa en cinco veces menos del tiempo que llevaría hacerlo en lenguaje de máquina. Este compilador llevó más o menos 25 años hombre tenerlo listo para ser trabajado por los programadores.

La primera versión del compilador FORTRAN en Noviembre de 1954 fué casi tan eficiente como los ensambladores existentes en ese tiempo, demostrando así que las ideas en contra de los compiladores no tenían razón de ser.

Hubo una serie de defectos en esta primera versión entre los que se incluye la falta de un proceso automático que permitiera detectar errores de sintaxis.

Medidas para resolver este y otros problemas se dieron en la segunda versión dando origen a FORTRAN II en 1955.

Una tercera versión de FORTRAN (FORTRAN III) se dió a conocer en 1958. Este producto fué utilizado por varios clientes de IBM sin embargo nunca se llegó a comercializar.
FORTRAN IV apareció en 1962 y salvo algunos detalles es la versión de FORTRAN más usada hoy en dia.

FORTRAN de alguna manera contribuyó al desarrollo de lenguajes de alto nivel, ya que cualquier persona que intentara vender una computadora para fines científicos, debería ofrecer un lenguaje de alto nivel, al menos tan bueno como FORTRAN.

De lo anterior algunos adoptaron FORTRAN teniendo como consecuencia el hacer de FORTRAN un lenguaje independiente de la máquina, mostrando asi que el algoritmo puede ser expresado sin hacer referencia a las caracteristicas del equipo.

En resumen FORTRAN tiene como caracteristicas:

a) Manejo fácil y eficiente de cálculos numéricos.

b) Evaluación de fórmulas matemáticas.

c)Posibilidad de agregar bibliotecas que no estén disponibles en el lenguaje.

e)No tiene estructura de bloque.

f)Un programa consta de un programa principal y cualquier número subprogramas o procedimientos externos.

Algunos descendientes de FORTRAN son :

PAF (Programmateur Automatique de Formules,)

APT (Automatic Progammed Tools)

GPSS(General Purpose Simulation System)

Hay tres lenguajes que muestran el valor y las desventajas de la persistencia de los creadores de lenguajes de programación.

El primero de ellos es ALGOL, el cual no ha sido usado mucho en la práctica pero que sin embargo ha contribuido como medio para el desarrollo y estudio de otros lenguajes.

El segundo de estos lenguajes es COBOL, el cual se definió bajo la supervisión del Depto. de Defensa de los Estados Unidos y el cual era independiente de la máquina en la que se trabajara.

El tercero fué PL/I definido tomando la combinación de las ventajas de FORTRAN y COBOL, con la estructura de ALGOL.

Se mencionarán algunos aspectos sobre estos tres lenguajes y algunos otros.

2.5 ALGOL

(ALGOrithmic Language)

Un grupo formado en Europa se hizo a la tarea de definir un lenguaje de multi-propósito que fuera independiente de la máquina y en el que cualquier algoritmo pudiera ser fácilmente programado. Un reporte acerca de esto fué publicado en 1958 definiendo el IAL (Internacional Algebraic Language), el que después fué llamado ALGOL 58.

ALGOL fué concebido bajo la idea de que un lenguaje de programación no sólo deberia ser una forma de comunicar al programador con la máquina sino con toda la gente, siendo asi que se prestara un poco más de atención a los problemas de entrada y salida.

Como ya se mencionó, ALGOL 58 se diseñó para poder programar cualquier algoritmo pero no se consideró la flexibilidad necesaria para el procesamiento de datos en aplicaciones de negocios.

Para los años 60's se creó una nueva versión de ALGOL conocida como ALGOL 60.

La principal diferencia entre ALGOL y FORTRAN es estructural.

Todo programa de ALGOL está hecho en bloques mientras que FORTRAN no sigue ninguna estructura en especial.

Como ya se dijo, ALGOL no estaba adaptado al procesamiento de datos. Su descendiente, JOVIAL, tenia mucho más campo de aplicación en este sentido.

A finales de 1958 el Ing. Jules I. Schwartz, propuso un lenguaje derivado de IAL, este fué llamado JOVIAL (Jules's Own IAL).

JOVIAL fué el lenguaje más "universal" de sus tiempos, y en sus inicios tuvo mucho éxito. Pero después su uso se vió disminuido por la llegada de PL/I.

La mayor desventaja de ALGOL fué la carencia de elementos concernientes con entradas y salidas, pero esta desventaja fué relegada a un segundo término ya que como ALGOL fué usado para propósitos cientificos eran necesarios grandes cálculos dando así un segundo lugar a las facilidades de entrada y salida.

El éxito comercial limitado de ALGOL hizo que muchas computadoras adpotaran FORTRAN en lugar de ALGOL, sin embargo esto no fué un impedimento para que ALGOL fuera una gran influencia para el desarrollo de lenguajes y técnicas de programación. En particular la estructura de bloque fué ampliamente adoptada.

ALGOL 60 siguió su desarrollo en manos de los europeos dando como origen a ALGOL 68 (1968), y adaptando los lenguajes de programación a las necesidades docentes.

Este último punto (lenguajes para la enseñanza) dió como idea el desarrollo de PASCAL (otro descendiente de ALGOL 58).

WFL (Work Flow Language) es un descendiente más de ALGOL. Este lenguaje fué desarrollado por BURROUGHS, hoy UNISYS, y es muy utilizado hoy en dia para facilitar la operación de algunos procedimientos.

Las caracteristicas principales de ALGOL son:

a) Facilidad de manejo de cálculos matemáticos.

b) Fué creado como un lenguaje independiente de la computadora.

c) La "forma" de sintaxis utilizada en ALGOL fué tomada por otros lenguajes.

d) Es un lenguaje de formato libre con estructura de bloque.

e) La sintaxis de ALGOL fué definida a través de BNF.

ALGOL ha resultado ser un lenguaje muy útil para la enseñanza de los conceptos computacionales.
Como ya se dijo, ALGOL fué definido esencialmente para el área de métodos numéricos y para algunas áreas de procesos lógicos. En la actualidad ALGOL ha sido usado para dar solución a diferentes tipos de problemas. La principal ventaja de ALGOL parece ser su "universalidad" para la solución de estos problemas.

Para resumir, se darán algunos aspectos que son considerados como contribuciones técnicas de ALGOL hacia la tecnología:

1. Estructura de bloque.
2. Definición de un lenguaje formal.
3. Procesos recursivos.
4. Facilidad para la solución de procesos computacionales.
5. Requerimientos para el desarrollo de mejores herramientas técnicas.
6. Predecesor de una gran número de lenguajes de alto nivel.

Probablemente la mayor contribución de ALGOL no está relacionada directamente con el lenguaje como tal, sino con la definición formal de la sintaxis del lenguaje y la publicación de reporter al respecto. Esto ha contribuido ha que se hagan estudios de los métodos de definición de lenguajes y al desarrollo de compiladores orientados a la sintaxis.

2.6 COBOL
(COmmon Business Oriented Language)

COBOL fué creado por común acuerdo en cuanto a la necesidad de un lenguaje independiente de la máquina y para el procesamiento de datos en aplicaciones de negocios. Para realizar esto se nombró un comité (Comité de COBOL).

El comité de COBOL examinó tres lenguajes de negocios existentes en estos tiempos.

Estos fueron:

1. FLOWMATIC (Desarrollado por el grupo de Grace Hopper UNIVAC (Sperry 1958, hoy UNISYS)).
2. AIMACO (AIr MAterial COmmand) (Desarrollado por un grupo de la Fuerza Armada de los E.U.)
3. COMTRAN (COMmercial TRANslator) (Bajo la responsabilidad de Rov Golfinger. IBM)

El comité decidió que COBOL deberia de ser diseñado de manera muy similar a los tres lenguajes mencionados, de tal suerte que se tuviera un avance más rápido.

Las caracteristicas principales asi como el uso de palabras de un lenguaje natural fueron tomados de FLOWMATIC.

Entre las caracteristicas de COBOL se tienen:

a) Su creación fué generada por las necesidades de las aplicaciones de tipo administrativo.

b) Las bases de sus sintáxis se acercan mucho a lo que es el idioma inglés.

c) Fué creado por un grupo de personas usuarias de diferentes tipos de máquinas lo que le dió la caracteristica de tener la misma aplicación en diferentes equipos.

e)Cuenta con cuatro divisiones (Identification Division, Environment Division, Data Division, Procedure Division)

f)Está enfocado al manejo de datos.

g)Facilidad para describir y manipular archivos de datos.

2.7 PL/I

(Programming Language I)

PL/I fué sin duda el lenguaje "más universal". Su campo de aplicación es muy amplio, teniendo contemplado entre otros a FORTRAN, COBOL, JOVIAL y LISP. Sin embargo para tener dicha cobertura tuvo que permitir varias formas de expresiones teniendo como consecuencia un lenguaje "difícil" de manejar.

PL/I es considerado el lenguaje más usado después de FORTRAN y COBOL.

Se cita como característica principal de PL/I la de proporcionar facilidades para tratar aplicaciones de tipo científico y comercial entre otras, así como la característica de ser un lenguaje con estructura de bloque.

2.0 LISP

(LISt Processor)

Fué implementado por primera vez en 1958 en una IBM 704. LISP está basado en el concepto matemático de función.
La primera versión sólo se podia trabajar en esta computadora. Una segunda versión se realizó en una IBM 709. Hoy en dia LISP es utilizado independientemente de la computadora en la que se trabaje.

La esencia de LISP consiste en el manejo de ciertos problemas de inteligencia artificial.
LISP abarca desde la manipulación de expresiones algebráicas hasta juegos de ajedrez, así como demostraciones de ciertos teoremas.

Como se mencionó LISP es muy popular con los especialistas de inteligencia artificial, ya que es muy fácil de aprender y emplear, pues todas las instrucciones tienen la misma estructura y también se cuenta con la recursividad. Sin embargo se tiene una desventaja la cual es el uso de paréntesis (pues en ocasiones resulta dificil saber cual paréntesis izquierdo va con cual paréntesis derecho, debido a que las instrucciones van entre éstos), pero está desventaja se puede eliminar si se es ordenado al programar, es más, en la actualidad existen editores para LISP que resuelven este problema.

2.9 Lenguajes Orientados a Objetos

Existe otra clasificación de los lenguajes de alto nivel. Esta es conocida como lenguajes orientados a objetos.

La programación orientada a objetos es un método relativamente nuevo para el diseño e implementación de sistemas de información. Sus propósitos principales son mejorar la productividad del programador así como controlar el costo de mantenimiento.

Cuando se utiliza la programación orientada a objetos, la fase de diseño se ve ligada a la fase de implementación.

La programación orientada a objetos se centra alrededor de algunos conceptos importantes como : tipos de datos abstractos y clases, jerarquía de tipos y herencia.

La parte más importante de la programación orientada a objetos es la relación con datos abstractos. Un tipo de datos abstracto es un modelo que abarca un tipo y un conjunto de operaciones asociadas. Estas operaciones están definidas para el tipo y lo caracterizan.

En la mayoría de los lenguajes orientados a objetos, una clase describe el comportamiento del tipo abstracto definiendo la interfaz a todas las operaciones que pueden ser hechas para el tipo. La definición de la clase también especifica los detalles de implementación o la estructura de datos del tipo.

Hay varios lenguajes orientados a objetos, entre estos se pueden citar SIMULA, C++ y ADA.

2.9.1 C++

C++ fué desarrollado por Bjarne Stroustrup en 1980. El doctor Stroustrup desarrollo C++ para tener herramientas que consideraran la escritura de algunos procesos de simulación.

C++ está basado en C. La eficiencia y poder de las expresiones en C, también forman parte de C++.

Lo más importante tal vez es que C++ cuenta con medios para trabajar en la programación orientada a objetos a un nivel muy alto de abstracción.

C++ es un lenguaje que va más allá que ADA en su manejo para la programación orientada a objetos y es similar a MODULA-2 en su facilidad de uso y manejo de modularidad.

ESTOS LENGUAJES QUE SE HAN MENCIONADO, SON TAN SOLO UNOS POCOS DE LOS MUCHOS LENGUAJES DE PROGRAMACION EXISTENTES. Y LA MAYORIA DE ESTOS SIGUEN EN USO EN LA ACTUALIDAD.

C A P I T U L O III.

L E N G U A J E S D E C U A R T A G E N E R A C I O N.

Este capitulo tiene como objetivo mencionar algunas de las caracteristicas de los lenguajes de cuarta generación también conocidos como 4GL, así como algunos ejemplos de estos. Se prentende dar a conocer un serie de 4GL's con el propósito de hacer una comparación entre los 3GL's y los ya citados.

3.1 Objetivo de los lenguajes de cuarta generación.

En respuesta a la creciente demanda en la construcción y mantenimiento de sistemas de información, los especialistas en Informática han demostrado interés en desarrollar nuevas tecnologias y productos de software que permitan el incremento de la productividad en el desarrollo de sus tareas y faciliten la aplicación de los recursos hasta por personal no especializado, de tal forma que el usuario defina únicamente su requerimiento y el sistema lo interprete y lo resuelva.

El objetivo principal que persigue el uso de herramientas avanzadas, conocidas como **LENGUAJES DE CUARTA GENERACION** , es facilitar al máximo posible el desarrollo de aplicaciones en materia de sistemas de información.

3.2 Caracteristicas de los 4GL.

Las caracteristicas minimas que deben reunir los lenguajes de esta categoria, son:

- Lenguaje de manipulación accesible, con tendencias al natural.
- Empleo de pocas instrucciones para generar una aplicación.
- Facilidad para aprenderlo.
- Proporcionar resultados en menor tiempo, en comparación con el consumido por el empleo de los lenguajes tradicionales.

De acuerdo a lo dicho anteriormente, los expertos en la materia definen como lenguajes de **CUARTA GENERACION** a **"todo aquel producto que permita obtener resultados en una décima parte de tiempo o menos, del que necesitaria el software tradicional"**
Los lenguajes de cuarta generación ayudan a los usuarios finales y a los programadores para el conocimiento del proceso de su trabajo o sistema, y el entendimiento del mismo, lo cual resulta ser una herramienta de gran ayuda para el programador para ser más productivo y a su vez conocer los requerimientos del usuario.

Entre las caracteristicas principales de los lenguajes de cuarta generación se pueden mencionar las siguientes:

1. Permiten la creación rápida de prototipos.
2. El sistema desarrollado por estos es para usuarios finales, de aqui que el usuario deba intervenir desde el principio en el desarrollo del sistema que utilizará para tener un entrenamiento completo.
3. Permiten ayuda en linea, por lo que se obtiene una mayor aceptación por parte del usuario final del sistema desarrollado con una herramienta 4GL.
4. Facilidad de programación y mucho más productividad para los programadores profesionales.
5. Algunos lenguajes de cuarta generación son generadores de código.
6. Facilidad de comunicación hombre-máquina, es decir, deberá reunir aquellos elementos que permitan una comunicación fluida, tales como forma de introducir y presentar información, recuperación de errores.
7. Independencia física y lógica de los datos, lo que equivale a la manipulación de los datos sin tener preocupación alguna con respecto a sus caracteristicas físicas de formato y almacenamiento.
8. Estructura modular.
9. Están basados en estructuras que proporcionan facilidad de manejo de los datos.
10. Los desarrollos de sistemas con estos lenguajes están

enfocadas en mayor parte a aplicaciones comerciales, las cuales involucran un alto grado de manejo de base de datos. En lo sucesivo se hará referencia a estas aplicaciones como "negocios".

3.3 Prototipos.

Se ha hablado de los prototipos, por lo que cabe mencionar que el objetivo de estos es ayudar a establecer la estructura y funcionalidad del diseño de un sistema. Trabajar con prototipos permite al usuario examinar las implicaciones de sus requerimientos.

Las caracteristicas de un prototipo son las de ser creado en forma rápida y eficiente, generando una reflexión fiel de los requerimientos y una representación exitosa del sistema.

Los 4GL son considerados una herramienta para el análisis de estos requerimientos.

Es muy común que haya errores en la definición de los requerimientos y estos normalmente son detectados hasta que se está en la fase de prueba o bien en la operacional, y en esta etapa resulta muy costoso el corregirlos.

Si se dá mantenimiento a los prototipos, es mucho más sencillo entender los cambios hechos en las especificaciones permitiendo de esta forma tener actualizada la documentación "automáticamente".

Es así que las deficiencias en el proceso de diseño se pueden minimizar evitando el re-diseño y la re-programación, es decir, a

través de los prototipos se le puede presentar al usuario final un esquema de sus requerimientos tal como se tendrían una vez terminado el sistema, de tal suerte que las modificaciones que se realicen serán rápidas y fáciles.

Se dice que el común denominador de los beneficios que se pueden obtener de los prototipos es la "calidad del lenguaje", evaluando ésta a través de su formalidad, operabilidad, eficiencia y facilidad de mantenimiento.

3.4 Otra forma de ver a los 4GL.

Otra forma más de definir o explicar a los 4GL es mencionando las tareas en las que serán usados.
IBM explica estas tareas asignando porcentajes:

40% MANEJO DE PANTALLAS. Los lenguajes de cuarta generación tienen la ventaja de utilizar pantallas, las cuales son diseñadas con gran facilidad.

40% MANEJO DE BASE DE DATOS. Los 4GL se caracterizan por un manejo fácil y rápido de la base de datos.

20% PROCEDIMIENTO. Este porcentaje está dividido en :
15% dedicada a generación de código, en lo cual los lenguajes de cuarta generación son especialistas y,
5% algoritmos.

3.5 Influencia de los 4GL en la metodologia del diseño de sistemas de información.

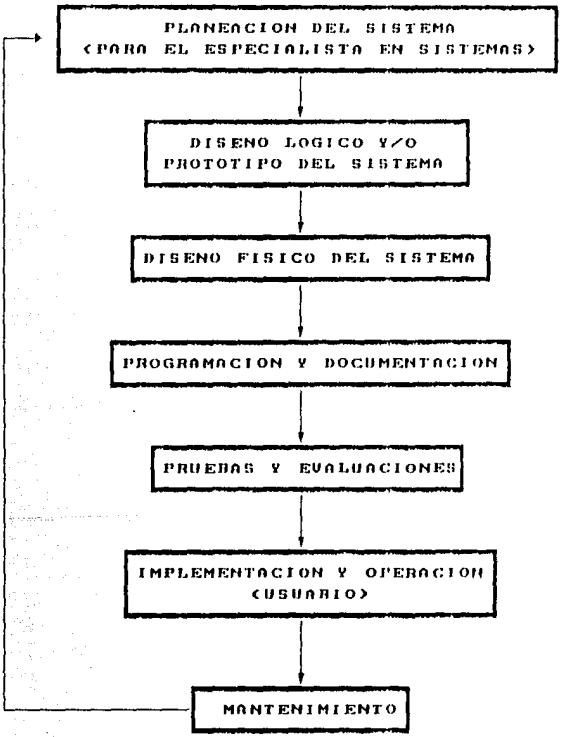
Los 4GL han tenido impacto en la modificación de la metodologia tradicional aplicada en la construcción de sistemas, en donde el técnico en Informática es el principal responsable de llevar a cabo las tareas y el usuario se encarga de especificar sus requerimientos y controlar que estos sean satisfechos.
Esto se puede ilustrar en el siguiente cuadro:

F A S E	RESPONSABLE TECNICO	USUARIO
Definición de requerimientos	X	X
Anàlisis del sistema	X	X
Diseño	X	
Programación	X	
Implantación	X	X
Producción	X	X
Mantenimiento	X	

Con la incorporación de los 4GL se tienen los siguientes cambios:

- Reducción de tiempo y esfuerzo de construcción y modificación de las aplicaciones.
- El usuario final asume un papel activo dentro del desarrollo, al grado que hasta él mismo es quien se encarga de crear y/o modificar sus aplicaciones.
- La metodología convencional para la elaboración de sistemas sufre una transformación radical, pués pasa de ser un conjunto de acciones lento y complicado a uno dinámico de "prueba y error", a través de la elaboración de modelos de sistemas que pueden ser modificados y vueltos a construir rápidamente como se muestra en el siguiente flujo.

lujo de desarrollo de un sistema de informacion con un 4GL



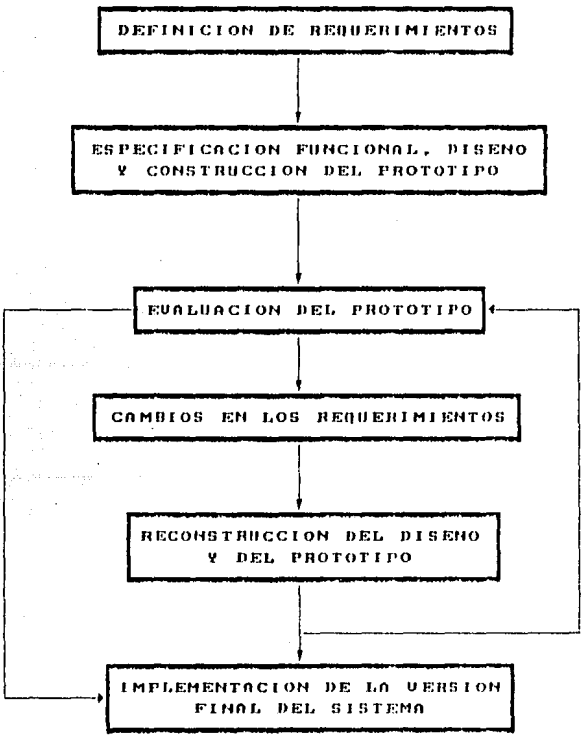
Tradicionalmente, un usuario final con un problema de negocio y un analista experto en sistemas de información, cada uno expresandose en su propio lenguaje, trabajan juntos para definir las especificaciones del programa. Luego, el analista traduce los requerimientos del usuario final a los programadores quienes generan el código del programa. Tipicamente varios programadores especializados utilizan diferentes herramientas de programación, escriben diferentes partes del programa, tales como la lógica del programa, las pantallas de entrada, los reportes, la base de datos y la comunicación de datos.

Consecuentemente cuando están terminadas, todas las partes no engranan; deben hacerse ajustes y secciones enteras de código deben reescribirse. El resultado, por supuesto, es el retraso del proyecto y un incremento de los costos.

Al continuar este proceso, la aplicación, repetidamente alterada y ajustada, se convierte en "frágil" y puede ser inestable y propensa a fallas. Una vez operativa y en linea, pueden requerirse ajustes por correcciones adicionales o para manejar condiciones cambiantes del negocio. Y el proceso tedioso, y costoso, de escribir y ajustar manualmente la aplicación, comienza de nuevo. El enfoque de los **lenguajes de cuarta generación** enfrenta la crisis de las aplicaciones por desarrollar, direccionando los requerimientos de negocio de los analistas y del usuario final de una manera similar. Ambos pueden disfrutar de un ambiente de dessarrollo integrado que sobrepase los intermediarios y responda al cambio rápidamente y en forma efectiva, desde el punto de vista de costos.

Esto se puede resumir en el siguiente cuadro.

CICLO DE DESARROLLO DE UN SISTEMA CON UNA HERRAMIENTA 4GL



3.6 Desventajas de los 4GL.

Debe mencionarse también que asi como los 4GL tienen muchas ventajas también tienen puntos en contra como pueden ser:

- Son exclusivos de la marca de computadora.
- Resulta muy poco compatibles con sistemas convencionales, ya que no apoyan en base de datos y/o archivos no estándar.
- Se requiere de un cierto grado de conocimiento en el área de sistemas.

Estas son algunas de las caracteristicas de los 4GL.
A continuación se dan algunos ejemplos de estos y sus caracteristicas particulares.

3.7 INFORMIX

INFORMIX es uno de los productos UNISYS más poderosos dentro de la herramientas disponibles en UNIX.

INFORMIX está basado en estándares tales como el Lenguaje de Consulta Estructurado mejor conocido como SQL (Strucured Query Language) y C-ISAM el método de acceso estándar para los sistemas operativos UNIX.

Entre las ventajas que puede ofrecer INFORMIX se encuentran la de hacer respaldos en línea, auto recuperaciones las cuales permiten manejar la información en un ambiente de demanda con mayor capacidad de almacenamiento y tiempo de respuesta.

INFORMIX-SQL, diseñado para toda la gente (programadores y no programadores), facilita las herramientas necesarias para la creación de aplicaciones con base de datos, como pueden ser: la facilidad de crear menús, consultas a través de SQL, un generador de reportes y un conjunto de herramientas para el mantenimiento y administración de base de datos.

Además INFORMIX-SQL ayuda a diseñar pantallas con los mismos formatos que se tienen en los reportes impresos.

Se puede dar mantenimiento a las estructuras de datos, pantallas, menús y reportes fácilmente a través de INFORMIX.

INFORMIX provee tres niveles de seguridad para el acceso de las bases de datos. Uno de estos es el nivel de seguridad que permite sólo aquellos usuarios con permiso, accesar una base de datos. Así también se tiene la seguridad de no permitir que la base de datos sea actualizada por más de un usuario al mismo tiempo.

INFORMIX ejecuta todas las funciones de un 4GL, como son la construcción de menús, pantallas, reportes. Se puede fácilmente mover información de una pantalla a un reporte.

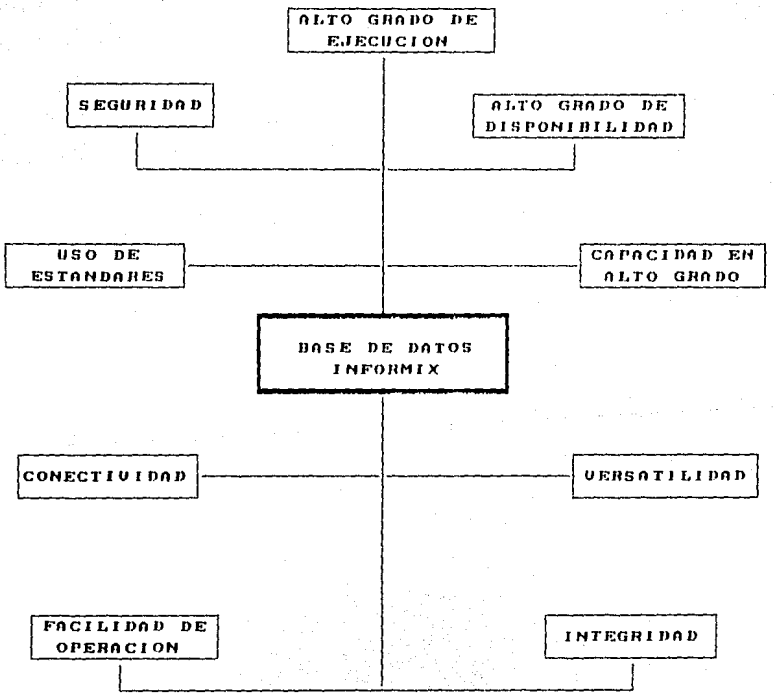
INFORMIX está diseñado con herramientas de SQL de ahí que se reduzca en gran escala el código necesario para el acceso y manipulación de las bases de datos. Con tan solo unos cuantos comandos, INFORMIX puede crear y alterar bases de datos, tablas, archivos indexados.

INFORMIX incrementa la productividad de la programación, teniendo una reducción de esfuerzos.

Usando comandos como SELECT, DELETE, INSERT y UPDATE, se pueden codificar subrutinas de mantenimiento a una base de datos en una fracción del tiempo requerido con los lenguajes convencionales de programación. Funciones que antes requerían algunas horas, ahora se pueden tener en tan solo unos cuantos minutos, claro está haciendo uso de INFORMIX.

Las características principales del manejador de base de datos INFORMIX se puede resumir en el siguiente cuadro.

CARACTERISTICAS SOBRESALIENTES DE UNA BASE DE DATOS INFORMIX



3.0 ORACLE

ORACLE permite a los usuarios tener acceso a una base de datos dentro de una red de computadoras de la misma forma como si estuviera en una sola computadora.

ORACLE distribuye información a quien la necesita sin importar la localización de la computadora dentro de la red.

Adicionalmente las caracteristicas de ORACLE le permiten hacer conecciones entre PC's, estaciones de trabajo, minicomputadoras y mainframes no importando los sistemas operativos que trabajen.

ORACLE tiene la posibililidad de integrarse con otros lenguajes de cuarta generación como MAPPER, LINC y ALLY.

ORACLE cuenta también con un generador de reportes el cual le permite crear todo tipo de textos para consulta. Se puede aprovechar la existencia de funciones matemáticas asi como la capacidad de formateo de datos.

Para aplicaciones de alto nivel se cuenta con interfases a COBOL, FORTRAN y C.

ORACLE cuenta con la facilidad de hacer respaldos o bien recuperación de bases de datos en linea mientras otras aplicaciones continúan ejecutándose.

Se cuenta con formatos que manejan las altas, bajas y consultas de la base de datos sin tener la necesidad de escribir una sola linea de código.

Los datos son introducidos a través de pantallas, teniendo

validaciones para estos, es decir, se revisa que la información dada corresponda a alguno de los formatos definidos.

Los menús del sistema se hacen a través de la introducción de información a la base de datos acerca de las opciones deseadas.

3.9. A L L Y

ALLY tiene su origen en 1986.

ALLY permite accesos simultáneos a diferentes bases de datos, permitiendo obtener información de cualquier parte de éstas.

Se pueden diseñar aplicaciones usando una base de datos y después si es necesario, hacer cambios sin ningún problema o dificultad.

Cuando se adoptan las herramientas que brinda ALLY no existe el riesgo de perder las inversiones hechas en lenguajes de tercera generación, por el contrario, se puede lograr integrar el ambiente de desarrollo existente con ALLY, teniendo así una disminución de costo y esfuerzo.

ALLY es un sistema de desarrollo UNISYS que no está ligado a una linea fija de hardware, sistema operativo o estructura de base de datos.

La facilidad de migración de aplicaciones o base de datos de un sistema operativo a otro es otra de las facilidades que proporciona ALLY.

Aún más, ALLY puede trabajar con programas escritos en lenguajes de tercera generación como por ejemplo COBOL o C, o con aplicaciones hechas en MAPPER o LINC II.

En el desarrollo tradicional la mayoria del tiempo se emplea en la codificación de las aplicaciones, mientras que con ALLY se describe únicamente la aplicación con el uso de menús y dando los datos requeridos. No hay necesidad de codificar ni compilar.
Con estos diálogos, el ciclo de desarrollo se reduce a sólo tres etapas, diseño, descripción y pruebas.
Para la creación de una aplicación se seleccionan una serie de bloques como pueden ser menús, formas, listas, gráficas, estructuras de consulta, reportes, o bien una serie de programas. Una vez hecho esto lo único que hace falta es conectar estos bloques en la forma deseada, lo cual permite establecer el flujo de la aplicación.
Esta forma de realizar el prototipo, permite a los usuarios finales hacer todo tipo de sugerencias o bien modificaciones a sus requerimientos durante el desarrollo y no cuando está terminado.
ALLY también cuenta con una serie de herramientas muy útiles para:

- La creación de pantallas.
- La migración de programas y/o bases de datos entre sistemas.
- El mantenimiento y elaboración de archivos.
- La distribución de aplicaciones que no estén sujetas a modificaciones (ambiente RUN-TIME).
- Dar ayuda a diferente nivel.
- Consulatar información para manejo de menús.

ALLY reconoce bases de datos generadas por :

ORACLE

INFORMIX

C

dBASE III

Algo más acerca de ALLY es su capacidad para accesar bases completamente diferentes dentro de la misma aplicación, por ejemplo, se puede combinar información de ventas proveniente de una base de datos ORACLE con información de inventario proveniente de una aplicación INFORMIX.

Si se desea cambiar la filosofía de una base de datos, las aplicaciones ALLY se pueden conectar de una base de datos a otra sin tener que alterar ninguna lógica en los programas.

ALLY ha sido diseñado para ser independiente de las bases de datos que trabaje.

ALLY es un sistema completo de desarrollo con ambiente independiente con lo que se refiere a hardware, sistemas operativos y bases de datos.

ALLY es fácil de trabajar a través de una serie de menús, además se cuenta con la ventaja de generar prototipos en una forma sencilla y rápida.

ALLY tiene la caracteristica de crear gráficas y reportes para aplicaciones. Un reporte en ALLY es una opción que traduce datos previamente seleccionados en una gráfica. Esta gráfica puede ser un diagrama de barras por ejemplo.

Como ya se ha mencionado, el ciclo tradicional de desarrollo tiene varias etapas: diseño, codificación, compilación, pruebas e instalación. El sistema ALLY reduce este ciclo a tres fases: diseño, descripción del prototipo e implementación. EL uso de ALLY elimina la pérdida de tiempo que se consume al codificar, compilar y ligar todo el código. Con ALLY la aplicación es diseñada, codificada y probada.

ALLY también cuenta con las herramientas necesarias para la migración de programas y bases de datos entre sistemas, así como la combinación de programas, obtención de listados, mantenimiento del archivo de mensajes.

Cuando un sistema ALLY ejecuta alguna aplicación, utiliza el flujo programado como un camino y extrae las estructuras de datos conforme las va necesitando. Esta estructura de datos permite que la ejecución de aplicaciones ALLY se realice tan rápido como cualquier aplicación codificada con un lenguaje de tercera generación.

3.10 M A P P E R

MAPPER es un sistema de software de UNISYS. Las siglas MAPPER corresponden al inglés "**MAintaining, Preparing and Processing Executive Reports (Mantenimiento, Preparación y Proceso de Informes Ejecutivos)**". Su capacidad de proceso ha ido creciendo constantemente a lo largo de los años y en la actualidad es capaz de realizar muchas de las funciones que se encuentran normalmente en aplicaciones comerciales de proceso de datos. No obstante, en la mayoría de los centros donde se ha instalado, las aplicaciones son creadas por los usuarios finales.

MAPPER es un sistema en línea. Sus usuarios crean archivos, informes y procedimientos desde terminales de pantalla. También pueden obtenerse informes impresos. En un sistema a gran escala, MAPPER puede ser utilizado por cientos de usuarios al mismo tiempo y transmitir informes de una terminal a otra.

Sus funciones básicas son fáciles de utilizar y son las que se usan la mayor parte del tiempo. No obstante tiene muchas funciones y variaciones de funciones que llevan cierto tiempo en aprender.

Los cálculos y la lógica pueden expresarse en instrucciones sencillas tipo FORTRAN. Esto permite a MAPPER realizar un

Los cálculos y la lógica pueden expresarse en instrucciones sencillas tipo FORTRAN. Esto permite a MAPPER realizar un procesamiento complejo si así se requiere.

El usuario final crea, procesa y actualiza reportes usando las funciones manuales de MAPPER. También diseña las formas que necesita para sus reportes.

La base de datos de MAPPER puede compararse con un cuarto de archivo de un sistema convencional de archivado. Un cuarto de archivo contiene archiveros para guardar la información. Cada archivero tiene varios cajones. Cada cajón tiene varios folders. Estos folder consisten en información similar agrupada.

En lugar del archivero, la base de datos MAPPER agrupa la información en MODOS. Cada modo contiene hasta 8 tipos. Cada tipo contiene uno o más reportes. Los modos son similares a los archiveros con su candado. La seguridad de MAPPER es la llave que prevee que usuarios no autorizados vean el contenido de los modos. Una vez que el archivero se ha abierto, no hay forma de restringir el acceso a la información dentro del archivero. Una vez que se accesa un modo los usuarios pueden ver todos los reportes en un modo.

La base de datos estÁndar de MAPPER contiene muchos modos, ubicados por la generación inicial del sistema operativo. Estos modos se contabilizan desde el modo 0 hasta el modo 420 (pudiendo ser mayor a 420). Los modos se asignan como 2 números, uno par y otro impar, por ejemplo 116/117. Esta numeración dual de un modo, por lo general se le denomina "par de modos".

Los datos a través de un modo impar no se pueden cambiar, sólo pueden desplegarse o imprimirse. En cambio los reportes accesados a través de un modo par se pueden desplegar y actualizar. El tener dos formas de acceso a los datos proporciona seguridad contra el cambio accidental de la información. La entrada de un usuario se puede restringir al uso de sólo el modo impar, de tal forma que no sea posible modificar la información.

Un modo contiene 8 tipos o cajones de información. Los tipos se identifican por las letras comprendidas entre la B a la I.

Los reportes tipo A pueden ser utilizados para introducir información en cualquier formato. Los datos no necesitan estar organizados en columnas. La única limitante es que no puede exceder de 80 caracteres. Los reportes de tipo A son accesibles desde cualquier modo. Esto significa que todos los usuarios de todos los departamentos que usan MAPPER tienen la facilidad de desplegar y actualizar información en los reportes de tipo A.

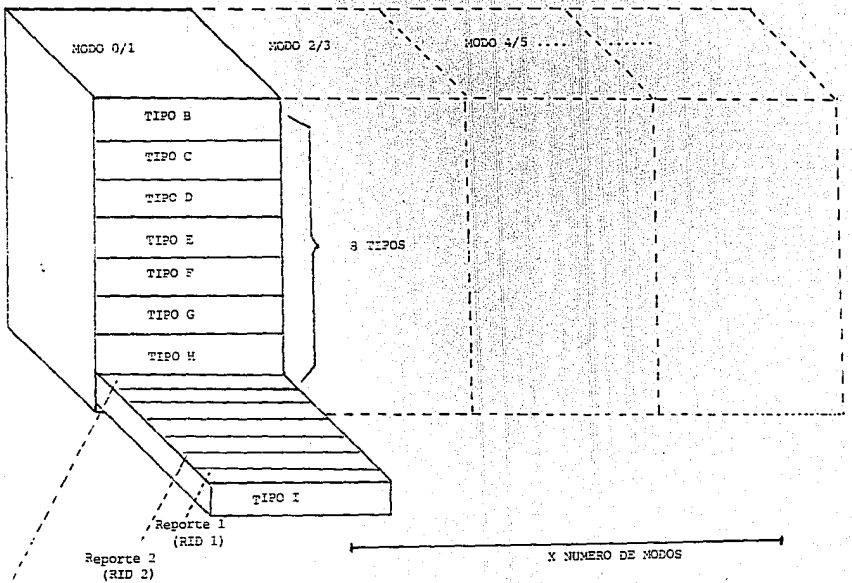
Estos tipos contienen reportes que consisten de datos organizados en columnas. Un reporte es un conjunto de datos dentro de cada tipo. Este consiste de un número de información que se organiza de acuerdo a la forma del reporte diseñado por el usuario para este tipo de reportes.

Los reportes se identifican por un número de reporte y una letra que indica el tipo (de la A a la I). A esto se le llama RID/tipo. ejemplo de estos serían 3C, 5C, 73F.

Este identificador se llama número de Report IDentification.

Los términos reporte y RID se usan indistintamente.

La siguiente figura muestra como están organizados los reportes, tipos y modos en una base de datos.


MODO 0/1
MODO 2/3
MODO 4/5 .....
TIPO B
TIPO C
TIPO D
TIPO E
TIPO F
TIPO G
TIPO H
8 TIPOS
TIPO I
Reporte 1
(RID 1)
Reporte 2
(RID 2)
Reporte n
(RID n)
X NUMERO DE MODOS

En MAPPER es necesario tener un coordinador el cual tiene como funciones el vigilar las actividades del sistema MAPPER.
Además es el responsable del sistema MAPPER en lo que respecta a su diseño, implementación, uso, mantenimiento y seguridad.
Así también es el que tiene la jerarquía necesaria para asignar privilegios y restricciones.
Los datos incluidos en los archivotes son creados y compartidos por muchos usuarios. Es necesario que el coordinador se encargue de controlar esto. Debe saber qué tipos de datos hay en cada archivo y hacer que sean tan homogéneos como sea posible.
Controla también los procedimientos de seguridad relacionados con el uso de los datos.
El trabajo del coordinador es aproximadamente equivalente al del administrador de una base de datos, pero muchos más sencillo. El administrador de una base de datos es un profesional calificado que se ocupa de estructuras de datos complicadas y del rendimiento de la computadora. El coordinador de MAPPER puede llegar a ser un usuario final, ya que cuenta con herramientas que se le proporcionan con el fin de asegurar que el sistema se use en forma racional e impedir que se pierda en el control del mismo.

El usuario final tiene como facilidades la creación, manipulación y eliminación de reportes, todo esto a través del uso de funciones manuales.

Además tiene la posibilidad de diseñar sus propios reportes.
Así mismo puede trabajar con el coordinador y el diseñador de aplicaciones para planear nuevos formatos de reporte.
MAPPER se utiliza generalmente como sistema interactivo, pero también puede conectarse con otros sistemas. Puede recuperar un archivo para procesarlo en MAPPER o pasar datos de MAPPER a otros sistemas.
Para lograr una mayor eficiencia o versatilidad, MAPPER se puede conectar a las ejecuciones en batch que se llevan a cabo en un lenguaje convencional. Así se podrá accesar una base de datos.
Los usuarios pueden transmitir datos desde su terminal a las terminales de otros usuarios. Pueden transmitir una pantalla de datos, o bien un informe completo. A menudo se transmiten mensajes que se crean en la terminal en lugar de datos almacenados en los archivos.

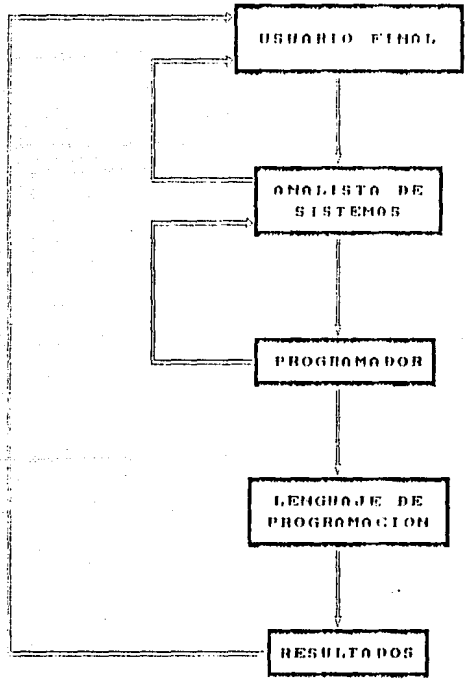
El sistema **MAPPER** está dirigido hacia el usuario final, aún cuando puede generar y correr aplicaciones sofisticadas para los profesionales de sistemas de información. El fuerte del sistema MAPPER está en generar aplicaciones para el usuario final, manipulando y organizando datos operacionales para apoyar la toma de decisiones. Y a través de su lenguaje de consulta SQL el sistema MAPPER puede fácilmente comunicarse con bases de datos IBM, DEC, UNISYS y otras.

Con su orientación familiar a "gabinetes de archivos electrónicos" y sus comandos de lenguaje ordinario, el sistema MAPPER ha aumentado el poder del usuario final para construir sus propias aplicaciones. Con un mínimo de entrenamiento, el usuario final desarrolla pantallas y reportes en la forma en que mejor se ajuste a sus necesidades. El sistema MAPPER automáticamente genera las bases de datos, lógica de programa, comunicación de datos y documentación.

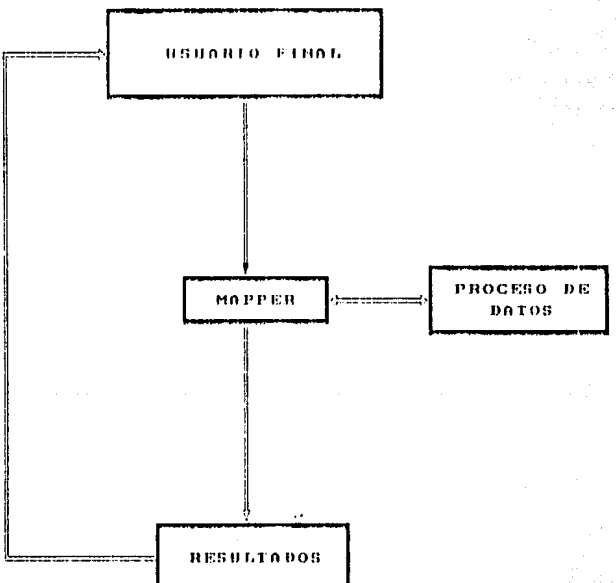
MAPPER es un lenguaje orientado a terminal que provee un ambiente muy poderoso para el uso de la computadora sin necesidad de programarla. También es un lenguaje interactivo de alta productividad para el desarrollo de aplicaciones mayores y en tiempo real. MAPPER ha ganado adeptos debido que sus comandos y conceptos del lenguaje son comprensibles y tan simples que tanto los usuarios finales como los profesionales de proceso de datos pueden diseñar aplicaciones mayores y en tiempo real en su computadora.

El ciclo de desarrollo en MAPPER es en gran escala diferente al ciclo tradicional. Esto se puede apreciar en los siguientes cuadros.

DESARROLLO TRADICIONAL DE UN SISTEMA.



DESARROLLO DE UN SISTEMA MAPPER.



MAPPER es el único producto de software que ofrece en un solo sistema lo siguiente:

- Cómputo sin programas usando una gran selección de funciones macro genéricas. Esto puede ser ejecutado interactivamente ya sea de manera individual o secuencial para convertir los datos en información de control. Consta de un juego completo de funciones de cálculo, actualización y selección así como procesamiento de palabra, correo electrónico y poderosas formas de graficación.

- Es un lenguaje interactivo sencillo de comprender, con controles de diccionario y directorio para desarrollo de aplicaciones mayores en tiempo real.

- Capacidades para base de datos en gran escala que están diseñadas exclusivamente para soportar eficientemente el proceso y la actualización de altos volúmenes de información en tiempo real. Interfases para acceso en ambos sentidos con otros sistemas y base de datos.

- Juego de funciones completas para seguridad, coordinación del servicio y monitoreo que proporciona un control total.
- Sistema de facturación para usuarios y contabilidad de sus recursos internos.
- Herramientas de usuarios final para análisis, diseño, documentación y desarrollo de aplicaciones con la ayuda de la computadora.

- Capacidad de recuperción de archivos en una forma total.
- Permite el uso de redes de comunicación dentro del sistema y con otros sistemas.
- Permite arquitecturas múltiples; desde una micro hasta macrocomputadoras.

MAPPER se distingue de otros lenguales de cuarta generación en lo siguiente:

- ORIENTACION A USUARIO.

  MAPPER es el único en su habilidad para soportar el diseño de aplicaciones mayores en tiempo real, tanto por usuarios finales como por los profesionales de programación.

- FUNCIONALIDAD.

  MAPPER tiene más de 100 funciones.

- ALTO GRADO DE PRODUCCION.

  Existen numerosos sistemas desarrollados en MAPPER de alta producción con un gran número de terminales en línea con un promedio de respuesta de menos de un segundo.

- CONTROL Y REDES.

  Seguridad, coordinación y monitoreo de servicios además de comunicaciones dentro del mismo sistema y con sistemas externos.

- ARQUITECTURAS MULTIPLES.

El sistema MAPPER esta disponible en computadoras personales, minis, macros así como en sistema de gran escala con multiprocesadores.

El sistema MAPPER es el único producto de cuarta generación con las siguientes capacidades:

- El usuario final define y desarrolla sus reportes.
- No son necesarios conocimientos de programación.
- Sistema multiusuarios.
- Respuesta inmediata.
- Integridad de los datos.
- Impide el uso de los datos por usuarios no autorizados.

Es importante hacer notar que los lenguajes de cuarta generación han contribuido notablemente para que los recursos humanos sean apovechados racionalmente. El siguiente cuadro muestra una comparación entre los rescursos en cuanto inversión de MAPPER y una herramienta de tercera generación como lo es COBOL.

COMPARACION DE LA INVERSION EN EL CICLO DE DESARROLLO ENTRE MAPPER Y COBOL

DESARROLLO Y MANTENIMIENTO	UNIDADES MONETARIAS	
	COBOL CLASICO	MAPPER
DISENO	10	20
ANALISIS DEL SISTEMA	10	
CODIFICACION	10	
PRUEBAS	20	
MANTENIMIENTO	50	5
	100	25

En resumen:

MAPPER es un sistema para procesamiento de reportes de propósito general en tiempo real. Permite al usuario final capturar, almacenar, recuperar, actualizar o imprimir reportes con una respuesta inmediata y sin requerir asistencia del programador.

3.11 L I N C II

LINC II toma su nombre de Logic and Information Network Compiler (Compilador II de Lógica de Información de Redes de Comunicación).

Para el desarrollo de esta sección se hará referencia a LINC II como LINC.

El sistema **LINC** es usado para el desarrollo de sistemas de información, utilizando la terminología cotidiana de negocios, con un proceso "automatizado" que genera código de programas compilables en COBOL, pantallas de entrada de datos, reportes, bases de datos y comunicación de datos.

Con el sistema LINC el analista puede desarrollar un prototipo del nuevo sistema e inmediatamente efectuar cambios en tiempo real, de manera que el usuario final pueda ver el sistema en acción y no solamente las especificaciones escritas.

Una vez que el modelo está completo, todos los elementos del sistema, documentación y código de programación son generados y probados automáticamente.

El sistema LINC comprime en forma dramática el ciclo de desarrollo, automatizando lo que varios programadores con varias herramientas diferentes hacen con tecnologías de lenguajes de tercera generación.

Para mantenimiento, el analista junto con el usuario final ajustan el modelo de negocios original, luego, al igual que antes, el sistema LINC genera automáticamente el nuevo código, base de datos, reportes y código de programas según las nuevas especificaciones.

LINC refleja un nuevo ambiente de desarrollo integrado de alto nivel, que genera rápida y eficientemente sistemas de información y aplicaciones para el usuario final capaces de responder a las condiciones cambiantes de los negocios. Como resultado, los usuarios de LINC pueden reducir en gran medida los ciclos de desarrollo y mantenimiento de aplicaciones, además de proveer acceso fácil al usuario final.

El ciclo de desarrollo de un sistema de información consiste en :

- Planeación
- Análisis y Especificaciones
- Diseño
- Codificación
- Pruebas
- Implementación

LINC reduce estas funciones a dos actividades: desarrollo y generación del sistema.

Para esto sólo se utiliza un ambiente de desarrollo, un conjunto de habilidades y un equipo de desarrollo, esto para completar la primera fase, la cual integra análisis y especificaciones,

prototipos y diseños. De aquí LINC continúa con sus tareas de generación, codificación automática e integración. Todo lo que se tiene que hacer es ver cuáles son las necesidades del usuario.

La clave para que LINC reduzca el ciclo de desarrollo de tal forma está orientado a su técnica de diseño.

El ciclo de desarrollo de LINC está basado en un proceso llamado prototipo evolutivo. El usuario final trabaja con un Analista LINC y el desarrollo requiere un esfuerzo conjunto durante todas las etapas del proceso. El prototipo se desarrolla continuamente hasta llegar a ser completamente funcional. El equipo de desarrollo explora diversas alternativas y selecciona la más adecuada.

Una vez definida la operación específica, LINC automatiza rápida y eficazmente el desarrollo del sistema. Acepta e incorpora especificaciones del usuario final y genera los programas requeridos para establecer, mantener y reportar la información correspondiente.

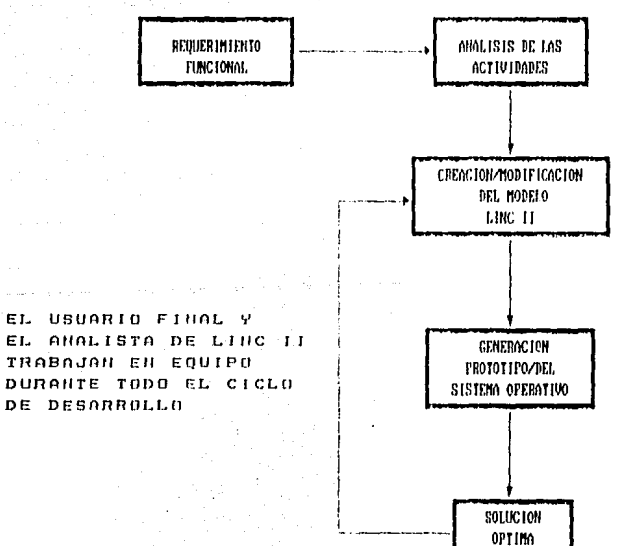
Cuando es necesario cambiar el sistema, el actual se convierte en un nuevo prototipo. Este proceso evolutivo continúa hasta obtener una solución satisfactoria. De esta forma se cuenta con un nuevo plano en una fracción de tiempo requerido para el mantenimiento tradicional del sistema.

Al reducirse el tiempo y el esfuerzo también disminuyen los costos, ya no es necesario, según estadísticas, destinar un 80% del presupuesto de proceso de datos al mantenimiento del software, ahora menos del 20% es suficiente.

GRAFICA PROTOTIPO EVOLUTIVO CICLO PARA EL DESARROLLO DE LINC II

P R O T O T I P O E V O L U T I V O

CICLO PARA EL DESARROLLO DE LINC II



LINC puede ayudar en los siguientes aspectos:

- Crear un sistema de información que maneje las necesidades de una compañía.
- Responde rápidamente a los cambios de ambiente en un negocio.
- Reduce el riesgo de tener sistemas obsoletos.
- Virtualmente elimina el mantenimiento de software, dejando el 80% del presupuesto de software para nuevos desarrollos.

De acuerdo a un estudio realizado en la Universidad de Auckland, Nueva Zelanda, se tienen los siguientes resultados:

- LINC reduce el tiempo de desarrollo en una razón de 20 comparado con los sistemas convencionales.
- Los sistemas LINC requieren un equipo de desarrollo mucho menor.
- En promedio, 14 lineas en código LINC equivalen a 114 de COBOL.
- Las características de LINC contribuyen a las mejoras de la productividad en donde:
  - El usuario tiene mayor oportunidad de estar envuelto en el desarrollo del sistema.
  - El programador solo hace uso de un solo lenguaje integrado en lugar de un conjunto de lenguajes y herramientas.

LINC crea un sistema de información único y totalmente integrado, al mismo tiempo permite a cada uno de los departamentos de la compañía definir y evaluar sus propias necesidades de información. Y al usar una sola base de datos, LINC preserva la integridad del diseño y de los datos. La información se captura y se almacena una sola vez, y después es compartida por quienes necesitan tener acceso a ella.

LINC está diseñado para gente que trabaja realizando sistemas de información, y lo hace de la siguiente manera:

- Las personas usan terminales para enviar y recibir mensajes desde LINC.
- El propósito de esta comunicación es que las personas definan sus requerimientos para la realización de su sistema.
- LINC genera sistemas para el almacenamiento y extracción de información, sobre las bases en que fué definido el sistema generado.
- Las definiciones hechas a LINC se almacenan en la base de datos LINC. Un conjunto de definiciones acerca de una organización constituye una definición para LINC.
- LINC puede almacenar diferentes especificaciones en su base de datos.
- LINC puede establecer comunicación entre diferentes personas sobre una misma especificación LINC, y a su vez lo puede hacer con grupos de personas sobre grupos de especificaciones.

Una vez que el programador y el usuario están de acuerdo con el concepto general del sistema, se define un conjunto de bloques u objetos que simulan el sistema.

El ambiente LINC utiliza términos de negocios y trabaja con estructuras orientadas a negocios.
Este define los bloques como **COMPONENTES, EVENTOS Y PROFILES.**

COMPONENTES. Describen la información estática acerca del negocio, tales como clientes, productos o códigos de venta.

EVENTOS. Son las actividades que se realizan dia a dia y transacciones de negocio tales como compras, ventas u ordenes.

PROFILES. Describen las perspectivas o vistas de los componentes o eventos.

LINC también ayuda a ver cómo se relacionan los bloque entre si.

El proceso para crear un sistema de información LINC en base a las especificaciones hechas a LINC es llamdado **GENERACION**.

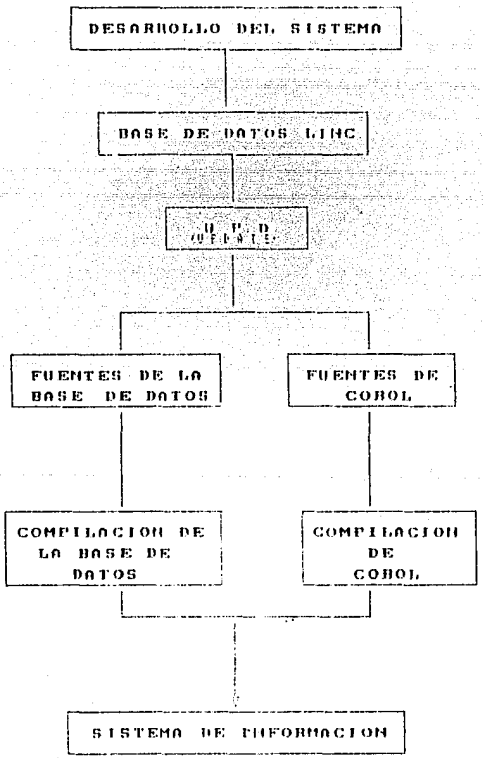
Cuando LINC genera el sistema, éste crea código fuente en COBOL y DASDL (Data And STructure Definition Language, lenguaje que es usado por el manejador de la base de datos para describir las caracteristicas fisicas y lógicas de la base de datos y el criterio ha ser usado para asegurar la integridad y seguridad de los datos contenidos en esta) y maneja la compilicación para obtener:

Todos los programas para el sistema de información deseado
La base de datos y el sistema de manejo de base de datos

Adicionalmente LINC puede ser utilizado para especificar, generar y compilar los programas necesarios para correr la red de terminales necesarias para el sistema.
Cuando se ha terminado de dar las especificaciones del sistema a crear o cuando se tiene una parte de estas ya concluida solo hace falta generar el sistema para hacer las pruebas necesarias.

El flujo de desarrollo de un sistema LINC y la generación del mismo se reflejan en los siguientes cuadros.

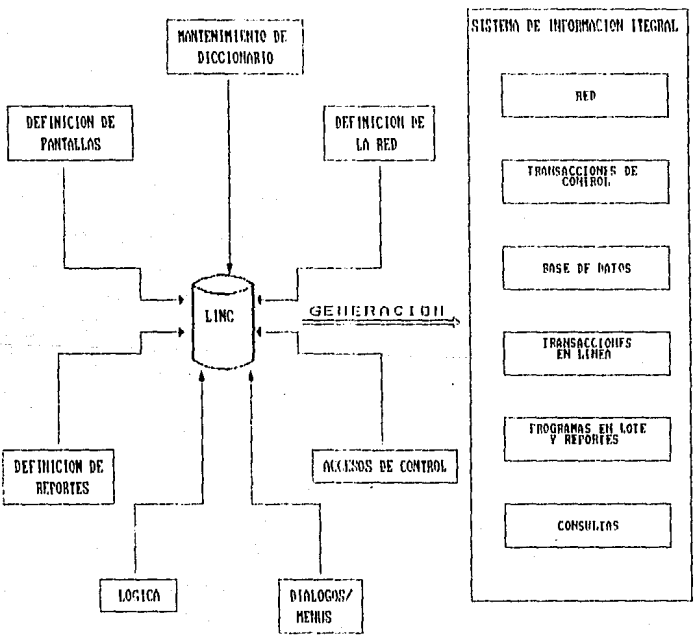
GENERACION DE UN SISTEMA LINC



ELEMENTOS INVOLUCRADOS EN EL DESARROLLO DE UN SISTEMA LINC

DESARROLLO | PRODUCCION

IMPLEMENTACION Y MANTENIMIENTO



A continuación se describe la estructura de LINC.

Se tiene programas llamados UPDATE los cuales tienen todas las operaciones que puede realizar el sistema de información generado por LINC. La base de datos en en donde se tienen almacenados los registros capturados.

La codificación de funciones pueden estar en un UPDATE o bien se pueden dividir las funciones en varios subsistemas.

Los subsistemas permiten ejecutar UPDATES con estructuras diferentes (COMPONENTES o EVENTOS). Se pueden ejecutar varias copias de cada uno de estos UPDATES (esta estructura se puede apreciar en la siguiente gráfica).

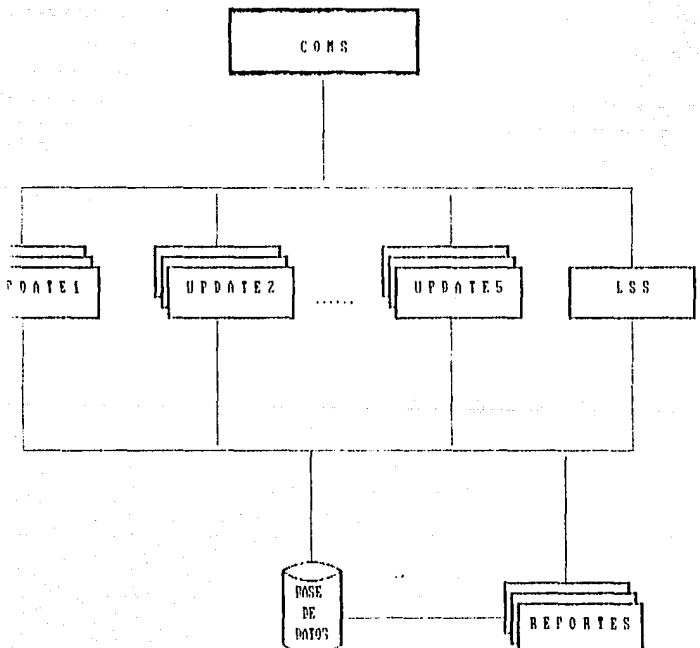
El sistema desarrollado en LINC es controlado en su totalidad por un supervisor llamado LSS (Linc System Supervisor).

La comunicación entre todos los UPDATES es manejada a través de COMS, nombre que se le asigna al MCS (Message Control System). Por ejemplo, un sistema bancario puede necesitar atender a 50000 transacciones de cajero y 50 de préstamo durante las horas pico del dia.

Puede resultar deseable ejecutar 10 unidades de funciones de cajero y sólo una de cualquiera otra de las funciones, entre las que se encuentra préstamos.

Si se ponen las funciones de cajero dentro de un subsistema llamado TELLER y el resto en el subsistema por omisión que lleva por nombre PRIMARY, entonces se pueden ejecutar 10 copias del subsistema TELLER y sólo una del subsistema PRIMARY. Cada uno de estos subsistemas están en diferentes UPDATES y cada una de estas copias es un UPDATE diferente que se ejecuta en la computadora.

ESTRUCTURA DE UN SISTEMA LINC



Como se mencionó, el subsistema por omisión es el PRIMARY. Si las funciones no se divide en subsistemas entonces todos los UPDATES son del subsistema PRIMARY.

Los sistemas LINC pueden tener hasta 5 subsistemas, y cada uno de ellos tiene un nombre y un número asociado para su identificación.
Siempre que se transmiten datos desde el sistema LINC los mensajes enviados a COMS contienen código de transacciones, los cuales corresponden a los nombres de los ispecs (COMPONENTES o EVENTOS).
COMS utiliza estos códigos para determinar cual UPDATE debe procesar la información.
Cualquier mensaje recibido por COMS que no contenga un código válido es enviado al Sistema Supervisor de LINC. Un ejemplo de mensaje de este tipo sería la ejecución de un reporte.

Los reportes tienen entre sus objetivos:

-Permitir la facilidad de tener información impresa en la presentación deseada por el usuario.

-Consolidar los datos, por ejemplo, para el borrado de registros es conveniente saber cuáles y cuántos fueron los borrados, o bien saber cuál información es o no relevante.

Los reportes se ejecutan en línea o bien en modo de lote, accesando la misma base de datos.

Un reporte puede ser iniciado desde la terminal en que se está trabajando con el sistema en línea. La salida del reporte puede ser dirigida a esta terminal, como una salida video (VD); o dirigida a una impresora cercana a la terminal (TP) o a una impresora de impresión rápida (LP).

OPCIONES DE SALIDA DE LOS REPORTES

La base de datos es un concepto fundamental de la filosofía de LINC y una parte intrinseca de un sistema de información LINC. Cabe mencionar la función de una base de datos en un sistema LINC:

-Es el medio que sirve de almacenamiento a todos los datos que se capturan en el sistema.

-Los datos contenidos en la base de datos, son las fuentes para la generación de la información que se espera obtener del sistema.

-Una herramienta muy fuerte de su manejador de bases es la de reorganizarla siempre que hay cambios en la estructura de ésta.

-La base de datos existente en un sistema es conservada cuando hay cambios, reintegrándose a la nueva base de datos.

-Otra contribución de la base de datos, es la de recuperarse automáticamente.

-LINC implementa su base de datos usando DMS (Data Managment System el cual es un manejador de base de datos que permite establecer la descripción de la base de datos). DMS tiene facilidades de recuperación, las cuales son utilizadas por LINC.

Lo que hace a LINC diferente de otros lenguajes se puede resumir de la siguiente manera:

' Se puede almacenar cualquier número de especificaciones de los sistemas a crear.

- Más de una persona puede trabajar a la vez con las especificaciones de un sistema, pero sólo una persona puede trabajar con un ISPEC.
- Se pueden hacer copias de las especificaciones dentro de las mismas (por ejemplo, si hay código de programación que sea aplicable a diferentes programas se puede hacer la copia con sólo teclear algunas instrucciones).
- Se cuenta la posibilidad de definir un diccionario de datos, el cual representa una gran ayuda, pues si se tiene que una variable es manejada en diferentes programas, no hay necesidad de definirla en cada uno de éstos, basta hacerlo en el diccionario una vez.

  Además si hay necesidad de modificar los atributos de estas variables, sólo se hace en el diccionario, de lo contrario, si no se define en el diccionario, se tendría que modificar en todos y cada uno de los programas.
- LINC revisa la información nueva que se especifica contra la ya existente y responde inmediatamente cuando detecta algún conflicto.
- LINC permite en gran medida revisar la sintaxis antes de compilar. Cuando se está programando, por cada pantalla de código realiza una revisión de sintaxis.

  Se tiene la opción de pedir que se verifique la sintaxis de todo el programa.

Estas son algunas de las caracteristicas que hacen que LINC esté dentro de los lenguajes de cuarta generación con mayor aceptación.

Esto se puede corroborar en el siguiente cuadro:

COMPARACION DE ALGUNOS LENGUAJES DE CUARTA GENERACION

	LINC	ORACLE	SAPIENS	MANTIS	NATURAL	NOMAD	IDEL	PROKIT
ANALISIS Y DISEÑO	MUY BUENO	BUENO	REGULAR	MUY BUENO	REGULAR	NO APLICA	NO APLICA	BUENO
MANEJO DE BASE DE DATOS	BUENO	MUY BUENO	MUY BUENO	BUENO	MUY BUENO	BUENO	REGULAR	REGULAR
TRANSACCIONES COMPLEJAS	EXCELENTE	EXCELENTE	BUENO	REGULAR	MUY BUENO	BUENO	BUENO	BUENO
CONSULTAS	BUENO	EXCELENTE	EXCELENTE	MUY BUENO	EXCELENTE	MUY BUENO	EXCELENTE	NO APLICA
REPORTES COMPLEJOS	EXCELENTE	MUY BUENO	REGULAR	MUY BUENO	BUENO	MUY BUENO	REGULAR	BUENO
INTERFASES	EXCELENTE	MUY BUENO	MUY BUENO	MUY BUENO	MALO	EXCELENTE	BUENO	REGULAR
TRANSACCIONES CON ARCHIVOS	MUY BUENO	MUY BUENO	EXCELENTE	BUENO	MUY BUENO	BUENO	BUENO	BUENO

ESTE CUADRO MUESTRA LOS RESULTADOS OBTENIDOS EN LA COMPARACION DE VARIOS LENGUAJES DE CUARTA GENERACION, EL CUAL FUE REALIZADO EN INGLATERRA EL 21 DE MAYO DE 1991.

Se ha hablado de la generación de un sistema LINC.
Pudiera pensarse que ésta es una desventaja ya que para cualquier cambio sería necesario "lanzar una generación", por muy pequeño que este fuera.

Esto no es así ya que LINC cuenta con una herramienta de gran ayuda, **"LITE"** (LINC Interpretive Test Environment)
LITE permite a los encargados del desarrollo hacer pruebas sobre un componente nuevo o bien con cambios a los mismos así como a los reportes sin necesidad de tenerlos generados.
LITE tiene las siguientes características:

- Es una extensión de un sistema LINC ya generado.
- Este parece ser otro UPDATE o bien otro reporte.
- No significa cambios al sistema en producción.
- No necesita ninguna herramienta ni hardware especial.
- No es necesario tener cambios en el ambiente de desarrollo.
- Utiliza la terminología que se trabaja en el desarrollo.

LITE ofrece las siguientes facilidades:

- Hacer pruebas sobre componentes nuevos o bien a los que se les ha dado mantenimiento sin necesidad de realizar ninguna compilación. Esto redunda en el gran beneficio de no tener que realizar un gran número de generaciones.
- Identifica, verifica y corrige los errores de codificación antes de que se lance la generación del sistema. De aquí que el número de generaciones también se vea reducido, pues se ganrantiza que al lanzar la generación esta tendrá éxito en lo que a código se refiere.

- Revisa la eficiencia y productividad de los ISPECS y reportes, esto lo logra haciendo un seguimiento del código y ayudándose con la información estadística que se produce.
- Diagnostica problemas relacionados con datos en el sistema de producción, sin necesidad de modificar o generar dichos sistemas.
- Incrementa la productividad de los programadores. Esto se logra debido a que una vez que se han hecho las modificaciones necesarias, no deben esperar a que la generación se haya lanzado.
- Se logra verificar si lo que piden los usuarios es lo correcto, pues antes de tener LITE se hacían los cambios requeridos, y una vez hechos resultaban no ser óptimos.

En resumen, LINC es la cuarta vesión de un generador de aplicaciones que puede ayudar a los analista a desarrollar programas que satisfagan los requisitos del sistema a un costo menor y hasta diez veces más rápido. Cualquier cambio posterior a los programas se podrá hacer rápida y fácilmente a un costo mínimo.

C A P I T U L O IV.

Elementos que intervienen en el desarrollo de un sistema LINC.

El objetivo de este capitulo es dar a conocer en una forma general los pasos a seguir en el diseño de un sistema hecho en LINC II. Con esto se pretende mostrar las ventajas que presenta LINC II como una herramienta de cuarta generación.

4.1 Fases de la Metodologia LINC para el desarrollo de sistemas de información.

La Metodologia LINC para el desarrollo de sistemas de información es una metodologia de diseño para un ambiente de generación de sistemas de información avanzados, enfocados esencialmente a apliaciones comerciales, en donde se hace uso de alguna base de datos y la cual puede hacer accesada en cualquier momento para la obtención de reportes. El Método LINC para desrrollo de sistemas se basa en un diseño orientado a objetos denominado Modelo de Objetos LINC. Este modelo permite a los diseñadores de sistemas concentrarse en "objetos" (actividades de negocios) completos en lugar de tener que segmentar el análisis de las actividades de un negocio en tres componentes individuales: datos, procesos (o funciones) y flujo de control.

La metodología LINC para el desarrollo de sistemas segmenta el ciclo de vida de desarrollo de sistemas en cinco fases:

1. INVESTIGACION DEL SISTEMA.
2. DEFINICION DEL SISTEMA.
3. DESARROLLO DEL SISTEMA.
4. IMPLANTACION DEL SISTEMA.
5. REVISION DEL SISTEMA.

Cada una de estas fases estará dividida en:
Descripción General: Se dan las caracteristicas generales de la fase.
Objetivos: Los objetivos a cumplirse dentro de esta fase.
Productos: Los elementos que se van a obtener en esta fase.
Tareas a realizar: Actividades que se realizan en esta fase.

Acontinuación se describen cada una de estas fases.

4.2 FASE 1: INVESTIGACION DEL SISTEMA

4.2.1 Descripción general

La Fase 1 se concentra en las actividades que ocurren alrededor de un segmento de negocios específico, con el objeto de entender la forma en que se lleva a cabo el negocio actualmente. Se investigan tanto las actividades manuales como las automatizadas.

Se asume que antes de iniciarse esta fase se ha desarrollado un

plan de información para la empresa, el cual provee una definición del alcance y los objetivos del segmento de negocios que está siendo investigado. El plan debe ser revisado y consultado durante la Fase 1.

Los resultados de la investigación son usados para identificar el alcance y los objetivos del nuevo sistema de información automatizado así como los recursos del proyecto que se requerirán para diseñar y desarrollar el sistema.

4.2.2 Objetivos

Se tienen los siguientes objetivos dentro de esta Fase 1.

- Definir las actividades que se llevan a cabo dentro y alrededor del segmento de negocios.
- Identificar los límites y objetivos del sistema de información.
- Identificar al personal y los recursos requeridos para el diseño y desarrollo del sistema automatizado.

4.2.3 Productos

- Descripción general del sistema de información.
- Plan de proyecto.

4.2.4 Tareas a realizar en esta fase

1 Identificar las actividades registradas dentro y alrededor del segmento de negocios.

2 Establecer los objetivos y limite de la implantación del sistema de información.

3 Desarrollar estimadores preliminares del tamaño del sistema.

4 Identificar el personal requerido.

5 Identificar otros recursos requeridos.

6 Preparar la descripción y el plan del proyecto del sistema de información.

4.3 FASE 2: DEFINICION DEL SISTEMA

4.3.1 Descripción general.

La Fase 2 define el sistema automatizado cuyo alcance y objetivos fueron determinados en la Fase 1. Comienza con la creación de un modelo conceptual orientado a objetos del nuevo sistema basado en las actividades identificadas en la Fase 1, pasando de ahí a definir los atributos técnicos del sistema tales como seguridad, respaldo y recuperación, equipo y software ambiental. Se diseñan los mecanismos para realizar la conversión de los sistemas actuales que pudieran existir al nuevo sistema, así como en las interfases con sistemas externos, en caso de existir éstas últimas. Además se evalúa el impacto que el nuevo sistema tendrá sobre el usuario.

4.3.2 Objetivos

- Definir el sistema automatizado.
- Diseñar el proceso de conversión.
- Diseñar las interfases.
- Determinar el impacto sobre el ambiente del usuario.

4.3.3 Productos

- Propuestas del sistema.
- Prototipos.
- Plan de desarrollo.

4.3.4 Tareas a realizar en esta fase.

1 Recolectar documentación.
2 Crear modelo conceptual del sistema.
3 Determinar los atributos técnicos para el sistema.
4 Establecer estándares de programación.
5 Definir en LINC el sistema automatizado.
6 Desarrollar prototipos.
7 Diseñar el proceso de conversión de sistemas previos en caso de existir.
8 Determinar el impacto sobre el ambiente del usuario.
9 Planificar para la Fase de Desarrollo del Sistema.
10 Conjuntar la propuesta del sistema y el plan para la fase de desarrollo.
11 Obtener la cooperación del usuario.

4.4 FASE 3: DESARROLLO DEL SISTEMA

4.4.1 Descripción general.

Esta fase se concentra en el desarrollo del sistema de información, una continuación del proceso iniciado en la Fase 2. Las principales tareas que se llevan a cabo en esta fase son las siguientes:

- desarrollo y prueba del sistema de información
- desarrollo y prueba del software de conversión
- preparación de toda la documentación
- capacitación

Esta es la fase en la que se desarrolla y se prueba el sistema final en preparación para su instalación como software en producción.

4.4.2 Objetivos

- Desarrollo del sistema completo.
- Preparación de la documentación.
- Finalización de las pruebas.
- Finalización de las capacidades.

4.4.3 Productos

- Sistema LINC, listo para su instalación.
- Software de conversión.
- Documentación.
- Resultados de las pruebas.

- Reporte de aceptación (opcional)
- Plan de implantación.

4.4.4 Tareas a realizar en esta fase.

- Desarrollar ispecs y reportes en LINC (en el anexo se muestran algunas pantallas de LINC para ejemplificar esto y algunos otros aspectos, siendo estos unos cuantos de los muchos existentes dentro de LINC)
- Desarrollar software de conversión para sistemas previos.
- Preparar documentación para el usuario.
- Realizar la prueba de aceptación.
- Realizar la capacitación de los usuarios.
- Planear la fase de implementación del sistema.

4.5 FASE 4: IMPLANTACION DEL SISTEMA

4.5.1 Descripción general

El objetivo de la fase de implantación es instalar el sistema probado y aceptado por el usuario en su ambiente de trabajo. Esto implica la migración del sistema en transición al sistema en producción y puede involucrar la instalación de equipo y software y la conversión de datos.

4.5.2 Objetivos

- Instalar el sistema de producción.
- Comenzar la utilización del nuevo sistema.

4.5.3 Productos

- Sistema vivo en producción.
- Reporte de implantación del sistema.

4.5.4 Tareas a realizar en esta fase.

1 Instalar el equipo, el software ambiental y la red de comunicaiones (opcional)

2 Instalar el sistema LINC

3 Completar la conversión de datos y realizar la transición del sistema.

A N E X O

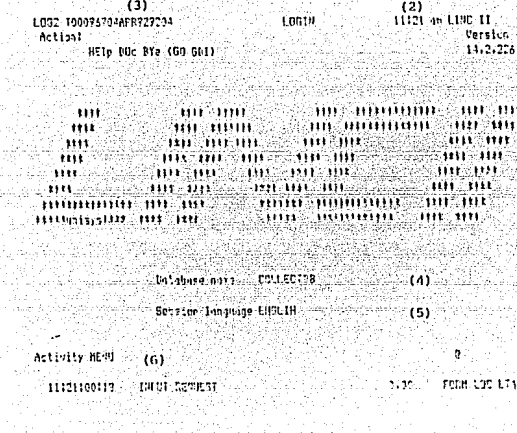
El siguiente grupo de láminas son sólo una muestra de algunas de las pantallas utilizadas en LINC.

El propósito de enunciarlas es mostrar las facilidades que proporciona LINC para la programación de sistemas de información.

LINC es un sistema hecho en LINC, por lo que hace uso de menús, pantallas de ayuda y documentación en linea.

En las siguientes láminas se hará referencia al ejemplo originado por la creación de un sistema de cobranza, cuya base de datos lleva por nombre COLLECTDB.

Así mismo se anexan un par de listados de programas codificados en LINC para poder observar que el grado de complejidad es mínimo.



Esta lámina corresponde a la pantalla de presentación de LINC.

En ella se despliega la versión de que se trata (1), la hora (2) y la fecha (3) de la sesión.

En ésta se indica el nombre de la base de datos que se desea (4), para este ejemplo se considera como nombre COLLECTOR. También se requiere el idioma que se utilizará para dicha sesión (5).

En el campo de Activity se le pide la actividad a realizar (6). En este caso se está solicitando mostrar el MENU de LINC.

Lámina No. 1

```
OSFECT0007690400R729204                OSTEC            11:25 on LINC II
 Action!                                                          Version
         HOme HElp DOc BYe GO                                     11.2.226
         L I N C   I I   P R I M A R Y   D A T A B A S E   O P T I O N S
                * COLLECTDB is an existing database

(1) LINC system nameCOLLECTSYS Description    MODULO DE COBRANZA  (2)
    Default pack    DMSLAG    Fireup ispec   SE400                (3)

    Decimals keyed          Y Decimal character . Separator characterN
    Leading zeros suppressed Y Full suppression  N Leftfill numerics  N
    Numeric CLV character   0 Alpha CLV character   Enforce dictionary N
    Convert to UpperCase    N Use 2 Bytes chs.  N 2 bytes CLV chs.

    Production system       N Adhoc inquiry reqd  N System uses ROC   N
    Integrity system        N Module master       N Module of

    Date format IN UK or US IN Active month number01 LINC security set  N
    Primary languageESPANOL    Global work size 0313
    Allowedcore        00080000 Base year         1987

   ActivityMENU
         (AS, FN, 11, US, GO DEC for system options and pack declaration)?

  11:25:22:43    INPUT REQUEST                          0.00     FORM RCV LIM
```

Esta pantalla es presentada siempre que la base de datos en la pantalla principal de LINC no existe. En ella se dan las características generales de la base de datos.

Se explican solo algunas de estas.

Como se mencionó, para seguir estas pantallas se trabajará con la base de datos llamada COLLECTDB.

En esta pantalla se le define el nombre del sistema que se creará (1) así como una descripción general del mismo (2).

Todos los sistemas hechos en LINC, tienen una primera pantalla que es mostrada al momento de accesar el sistema desarrollado en LINC, esta pantalla lleva por nombre para nuestro ejemplo SE400 y se le indica en el campo FIREUP ISPEC (3).

Los demás campos tienen como función definir las características de edición de los campos de la base de datos, como pueden ser:

Caracter para separador de decimales, caracter numérico o alfabético nulo, etcétera.

Lámina Nov. 2

```
DBSSET0007700AAPD729204            DBSSE              11:26 am LINC II
Action!                                                        Version
     HOme HElp DOc BYe GO                                      14.2.226
                 D A T A B A S E     C H O I C E S
             This format requests the database name
                 and your choice of functions!

                   LO    Log on to database
             (1)   DE    Delete database
             (2)   GE    Generate database
             (3)   OP    Option change
             (4)   PK    Pack specification
             (5)   TX    Edit comment text
                   AS    A Series system options
                   US    U Series system options
                   11    1100 Series system options
                   80    System 80 system options
                   UP    U Series LINC pack definition
                   LP    1100 Series LINC pack definition

     Name   :  COLLECTOR
     Choice :  LO

 11:26:17:77   INPUT REQUEST                      0.00      FORM FOR LOAD
```

Como toda base de datos, es necesario darle mantenimiento. Es en esta pantalla en donde se le indica el tipo de cambio que se requiere hacer.

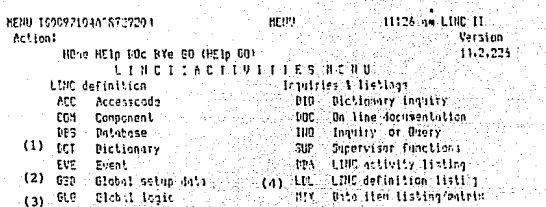
Por ejemplo se puede pedir que se borre (1) o bien que se genere ó compile el sistema (2).

Que se hagan cambios a los atributos definidos a la base (3) (los mencionados en la lámina 2).

Definir los discos en donde quedarán residentes el diccionario de datos, los objetos, las auditorias, etcétera (4), o bien tener documentación acerca de esta base de datos (5).

Las demás opciones son especificas del equipo en el que se trabaja

Lámina No. 3



En esta lámina se muestra el menú general de LINC.

En él se exhiben todas las funciones que se pueden realizar con LINC.

Por ejemplo:

(1) Se puede definir un diccionario de datos.

(2) Se pueden definir variables globales.

(3) Se pueden definir bloques de código que sean comunes a varios programas.

(4) Se pueden obtener listados de programas en una gran variedad de formas.

(5) Se puede hacer un respaldo de la base de datos.

En el campo de choice se le indica que opción se desea, en este caso se le está requiriendo que vaya a la opción de estructuras (COM).

Lámina No. 4

```
I3PSET0099730IAFR727204              ISFSE          11:27 am LINC II
Action:                                                      Version
       HOme HElp DOc BYe GO                                  11.2.225
              C O M P O N E N T   C H O I C E S

            This format requests the component name
            and your choice of functions:

          (1)  PA    Paint screen format

               LS    Edit main LINC logic
          (2)  PC    Edit pre-screen logic
               PL    Edit pre-LINC logic

          (3)  DE    Delete component
          (4)  OP    Option change
          (5)  TE    Edit teach screen
          (6)  TX    Edit comment text
          (7)  TP    Set/reset trace option

      Name   :  CF015
      Choice :  LS
       TE page     0

  11:27:35:35    INPUT REQUEST                      0.00     FOUR LOC LTAI
```

Esta es la pantalla en la que se dá mantenimiento a las estructuras de una base de datos. Se cuenta con varias opciones:

(1) Si esta estructura tiene una pantalla asociada, se le pide el editor para diseñarla.

(2) O bien se invoca el editor para teclear código.

(3) Inclusive se puede borrar una estructura.

(4) Se pueden hacer cambios a la definición de la estructura (estas definiciones se mostrarán en la siguiente lámina).

(5) Una opción muy útil es la definición de "ayuda en línea", y ésta se hace por cada estructura si así se desea.

(6) Se puede tener la documentación del código dentro de éste.

(7) En caso de ser necesario, se puede dar el seguimiento de la ejecución de los programas.

Junto con la opción deseada se debe dar el nombre de la estructura a trabajar. En este caso el nombre es CF016

```
ISFC1T00077404AFF727204              ISFC            11:27 am LINC II
 Action:                                                       Version
       HOme HElp DOc BYe GO                                   11.2.226
             L I N C  I I  -  C O M P O N E N T   O P T I O N S
                   CP016 is an existing component
          Brief description        CONFIRMACION DE SALDOS
          Author's name            ALEJANDRA MENDEZ
     (4)  Component type           M           Automatic recall          N
     (1)  Usage Input              Y       (2) Usage Output              N
          Automatic entry          N           Refresh screen            Y
     (3)  Copy from line number    10      (3) Copy to line number       [illegible]
     (3)  Max copies               12          Repeat from line number   [illegible]
          Expected number                      Test factor (%)
          Autonaint profile name               Ordinates downloadable    N
          Cursor field                        Accessible by III          N
          Subsystem name           [illegible]     Integrity                 N
          Pack name                DEFAULT     Updates allowed           Y
          Actmth NOS               Y           V Series widow  (0,2-9) 3
          High frequency           N           V Series section (0-300) 00[illegible]
          LINC security level      0          1100/2200 Tip File number0000
          User privilege level     01          Background color
          Primary profile name                 Foreground color
                Then, Function [illegible]              ([illegible] none for [illegible])

 11:27:47:34   INPUT [illegible]                              [illegible]    [illegible]
```

Esta pantalla es el medio para definir las características de las estructuras dadas de alta.

Entre estas características se pueden citar:

(1) La estructura es de entrada.

(2) La estructura es de salida.

(3) Si esta estructura trabaja con más de un renglón con em mismo formato.

(4) Si la estrutura dá de alta automáticamente, etcétera.

Lámina No. 6

```
NEXT ....*....1....*....2....*....3....*....4....*....5....*....6....*....7..
:****************************************************************
:***** CP016-LG.- CONFIRMACION DE SALDOS                     *****
:****************************************************************
:
:----- FORMACION DEL TITULO DE LA PANTALLA
:
MV; GLB.WORK      GW-WORK
MV; GW-OTROS      GW-WORKR
DW; GLB.COPY      =        GLB.MAXCOPY
    MV; MMAINT             GD-ACCION
    INS;GL-ARMA-TIT
    INS;GL-MUEVE-TIT
    INS;GL-SEG-LG
END;
DW; MMAINT        =        GD-ING
   DW; GLB.COPY   =        GLB.MAXCOPY
       DT; EVERY  [illegible] ([illegible] DATE-CAP)
           MV;CP100.TOT-CTRL     TOT-CTRL
           MV;CP100.TOT-DETAIL   TOT-DETAIL
           MV;CP100.TOT-DIF      TOT-DIF
           MV;CP100.[illegible]    TOT-[illegible]
           MV;CP100.STATUS[illegible]  [illegible]
           INS;GL-[illegible]
  [illegible] COLLECTOR [illegible]                    [illegible]
```

Una vez que se han definido las características de la estructura se pasa al editor, el cual es de tipo CANDE.

En esta lámina se muestra parte de la codificación de la estructura CP01B.

Lámina No. 7

```
                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXPROG.
                         M O D U L O   D E   C O B R A N Z A
XXXXXXXXXXXXXXXXXXXXXX        XXXXXXXXXXXXXXXXXXXXXXXXXXXXX (3)

  :  0000000      TOTALES: CONTROL    0000000000000000000+
                           DETALLE    0000000000000000000+    NUM DOCTOS : 000    XXXXXXXXXXXX          DIFER

TDA        DOC       IMPORTE DOCTO.           IMPORTE M.N.
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+  (2)
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+
00         000       0000000000000000000+     0000000000000000000+

           COMENTARIOS ??          (S=SI, N=NO)
XXXXXX     CERRAR LA RELACION??    (S=SI, N=NO)     XXX?
[illegible]                                  [illegible] NEW LINE
```

Algunas estructuras pueden tener asociada una pantalla. En este caso se muestra el diseño de la pantalla correspondiente a la estructura CP010.

En esta pantalla se tienen "displays" como MODULO DE COBRANZA, NUM. DOCTOS:, también se tienen variables numéricas enteras (1) ó signadas (2), representadas por ceros y variables alfanuméricas dadas por X's (3)

Lámina No. 8

```
CP0161000[illegible]R049204     INDUNISYS DE MEXICO        PROX.
                      M O D U L O   D E   C O B R A N Z A
 MODULO DE COBRANZA                   CONFIRMACION DE SALDOS

RELACION  :   0000300      TOTALES: CONTROL
                                    DETALLE
 NUM DOCTOS :                       DIFERENCIA

     MONEDA         DOC        IMPORTE DOCTO.          IMPORTE M.N.




                   COMENTARIOS ??        N [illegible], [illegible]
                 CERRAR LA RELACION?  [illegible] SI, [illegible]    [illegible]
 11:23:35:56   INQUIRY REQUEST                              [illegible]
```

Una vez que se definen todos los ispecs y se programa el código correspondiente se procede a generar o compilar el sistema.

El sistema ya generado tendrá sus pantallas y programas para ser ejecutados. En el ejemplo que se está trabajando, la pantalla que se diseñó en la lámina número 8 queda como se presenta en ésta.

Lámina No. 9

```
CF016T00064772AFR047201      INDUNISYS DE MEXICO          PROX.
                       M O D U L O    D E    C O B R A N Z A
 MODULO DE COBRANZA              CONFIRMACION DE SALDOS

RELACION   :  139720205        TOTALES: CONTROL          141316.001
                                        DETALLE             1413151.001
 NUM DOCTOS :   3                       DIFERENCIA

     MONEDA            DOC         IMPORTE DOCTO.          IMPORTE M.N.
       92               1               80000.001              30000.001
      200               2               61346.001          [illegible]

                  COMENTARIOS ??          S (S)SI, (N)NO
AUTORIZADO         CERRAR LA RELACION??   N (S)SI, (N)NO     [illegible]
  [illegible]   INQUIRY REQUEST                              [illegible]
```

Una vez que se le pide información al sistema, este procede a ejecutar el código y presentar la información requerida en la pantalla.

Esta lámina es un ejemplo de la presentación de la información.

Lámina No. 10

```
REFSET0009750 JAFR929204              REFSE              11:27 am LINC II
 Action:                                                          Version
        HOme HElp DOc BYe GO                                      14.2.22
                          R E P O R T   C H O I C E S

                    This format requests the report name
                      and your choice of functions:

                    (1) PA     Paint frame

                    (2) LG     Edit logic

                    (3) DE     Delete frame or report
                    (4) GE     Generate report
                    (5) OP     Option change
                    (6) TX     Edit comment text


        Name   : RP002
        Choice : LG
        Frame  : 0
        (for PA,LG or DE)      2

 1112911:30:19    INPUT REQUEST                               FORM LOC LTAI
```

Otra opción del menú general de LINC corresponde a la definicón de reportes (REP). Cuando se invoca esta opción en el menú general, se presenta esta pantalla en la cual se le indica el nombre del reporte y la opción a ejecutar. Entre estas se ecuentran:

(1) Diseñar la salida del reporte.

(2) Edición de código.

(3) Borrar el reporte.

(4) Compilar el reporte.

(5) Cambiar opciones definidas para el reporte (estas se muestran en la lámina 12).

(6) O bien, documentar el código.

En este caso se está dando como nombre de reporte RP002.

Lámina No. 11

```
REP2 T00132517FEB929202                                  2:54  pm LINC II
 Action:                                                          Versinn
       HOme NElp POc BYe GO                                       14.2.226
          L I N C  I I  -  R E P O R T  O P T I O N S
             REP   RP002 is an existing report

                Database                  COLLECTOR
                Brief description         RELACION DE FALTANTES
                Author's name             ALEJANDRA MENDEZ
          (1)   Default device            LP      Video capable     N  (2)
          (3)   Print line length         132
                Default pitch             132
          (4)   Spacing                   SINGLE
                Standard heading          N
          (5)   Extract pack              DEFAULT
                Integrity                 N   Currency sign         $
                Decimal point             .   Separator character   N
          (6)   Database Update allowed    Y   Able to be transferredN
                LINC security level       0
                Critical Point SD Name

                Then, choice    :LG
                     frame no :000

 14:54:16:62    INPUT REQUEST                                  FORM RCV LTAI
```

En esta pantalla se definen las opciones de los reportes.

Algunas de ellas son:

(1) El dispositivo de salida del reporte.

(2) Si es un reporte cuya salida se puede apreciar en video.

(3) La longitud de las lineas de salida

(4) Espaciado.

(5) Si se genera archivo, en que disco dejarlo residente.(6)

(6) Si el reporte puede actualizar la base de datos.

```
NEXT ....*....1....*....2....*....3....*....4....*....5....*....6....*....7..
!**********************************************************************
!*                          RP002                                     *
!*                  RELACION DE FALTANTES                             *
!**********************************************************************

SD;SDFECGR          GROUP
    SD;SDAYCG       EDIN LE;2
    SD;SDMESG       EDIN LE;2
SG;
SD;SDFECHA          EDIN LE;6
SD;SDIZZ            EDIA LE;12 (ZZZZZZZZZZZ)
SD;SDDIA            EDIA LE;10
SD;SDCLOSE          EDIN LE;1  (0)
MV;(RP002)          GD-REPORTE
INS;GL-TITREP
MV;GD-FECHA         SDFECHA
FL;GC-2             SE202.STATUSREP
LU;SE202.NUMCIA (FR100)
MV;FR100.CURR-PER   SDFECGR
MV;SDMESS           GD-MESM
INS;GL-MES
MV;SC-1             GP-DIA-1
MV;SDMESG           GP-MES-1
  11:27:17:00 ** COLLECTOR Normal 16,12  RP002 *                  WORK SCV LTAL
```

Una vez definido el reporte se procede a codificarlo, de la misma forma que un ISPEC (con un editor CANDE).

Un ejemplo de la edición de un reporte se presenta en esta lámina.

Lámina No. 13

MODULO DE COBRANZA

02 RELACION DE FALTANTES
CORRESPONDIENTES AL PERIODO CONTABLE DE NOVIEMBRE DE 1991

USUARIO 138
CREDITO Y COBRANZAS

DIA	TOTAL DE CONTROL	DOCS.	STATUS	
VIERNES	0.00	0	*****	FALTA RELACION DE ESTE DI
SABADO	123456.00	2	DESBALANCEADO	DIA FESTIVO/NO LABORABLE
DOMINGO	10000000.00	1	DESBALANCEADO	DIA FESTIVO/NO LABORABLE
LUNES	0.00	0	*****	FALTA RELACION DE ESTE DI
MARTES	10000000000.00	1	DESBALANCEADO	
MIERCOLES	10000000000.00	1	DESBALANCEADO	
JUEVES	10000000000.00	1	DESBALANCEADO	
VIERNES	0.00	0	*****	FALTA RELACION DE ESTE DI
SABADO	21142317.00	1	AUTORIZADO	DIA FESTIVO/NO LABORABLE
LUNES	100000.00	2	AUTORIZADO	
MARTES	100000.00	2	AUTORIZADO	
MIERCOLES	0.00	0	*****	FALTA RELACION DE ESTE DI
JUEVES	0.00	0	*****	FALTA RELACION DE ESTE DI
VIERNES	2000000.00	3	DESBALANCEADO	
LUNES	0.00	0	*****	FALTA RELACION DE ESTE DI
MARTES	0.00	0	*****	FALTA RELACION DE ESTE DI
MIERCOLES	100000000.00	1	DESBALANCEADO	DIA FESTIVO/NO LABORABLE
JUEVES	2000000.00	7	DESBALANCEADO	
VIERNES	2000000.00	2	BALANCEADO	
SABADO	0.00	0	BALANCEADO	DIA FESTIVO/NO LABORABLE
DOMINGO	1000000.00	1	DESBALANCEADO	DIA FESTIVO/NO LABORABLE
LUNES	1000000.00	2	BALANCEADO	
MARTES	130000.00	5	DESBALANCEADO	
MIERCOLES	1250000.00	1	DESBALANCEADO	
JUEVES	1300000.00	2	DESBALANCEADO	
VIERNES	1500000.00	5	DESBALANCEADO	
SABADO	500000.00	1	DESBALANCEADO	DIA FESTIVO/NO LABORABLE

Al igual que los ISPECS es necesario compilarlos para poder ejecutarlos.

Un ejemplo de los resultados al ejecutar el reporte RP002 es dado en esta lámina.

CONCLUSIONES

En base a los temas expuestos en este trabajo de tesis se puede concluir que los 4GL son realmente una herramienta que mejora la productividad en aquellos que se dedican al desarrollo de sistemas de información, teniendo en cuenta que para alcanzarla es necesario realizar cambios en las técnicas de diseño y manejo de proyectos.

Esta productividad se ve reflejada en varios aspectos, se mencionarán solo algunos de ellos:

1. El tiempo requerido para el desarrollo de un sistema de información con una herramienta 4GL se ve substancialmente reducido en comparación con el utilizado por un 3GL.
2. En algunos casos se tienen beneficios para los programadores, pues con el uso de lenguajes de cuarta generación, la codificación se convierte en un proceso más sencillo ya que el número de lineas de código se ve reducido.
3. Se logra satisfacer al usuario final con tan solo hacer unos cuantos procesos y liberarlos para que este cree y defina sus propios reportes, manejando la información en la forma que más le convenga.

Sin embargo se ha mostrado con los ejemplos de tan solo algunos lenguajes de cuarta generación, que estos difieren en sus capacidades, por lo que la elección del 4GL para la aplicación a desarrollar es de gran importancia para el éxito del producto que se desea obtener.

En resumen:

Como experiencia propia, puedo decir que es mucho más sencillo y rápido trabajar con herramientas de cuarta generación por la gran variedad de opciones que brindan para el desarrollo de sistemas de información, dejando satisfechos tanto al usuario como a los programadores por los resultados que con éstas se puedan obtener.

B I B L I O G R A F I A

-AN INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING AND C++
RICHARD S. WIERNER & LEWIS J. PINSON
Addison-Wesley Publishing Company.

-PROGRAMMING LANGUAGES, DESIGN AND IMPLEMETANTION
TERRENCE W. PRATT
Pretince Hall.

-FOURTH GENERATION LANGUAGES UNDER DOS AND UNIX
JOSEPH ST. JOHN BATE & DINESH B. VADHIA
BSP Professional Books.

-THE STRUCTURE AND DESIGN OF PROGRAMMING LANGUAGES
J. E. NICHOLS
Addison-Wesley Publishing Company.

-PROGRAMMING LANGUAGES, HISTORY AND FUNDAMENTALS
JEAN E. SAMMET
Pretince Hall

-FOURTH AND FIFTH GENERATION PROGRAMMING LANGUAGES VOL. 2
DIMITRIS N. CHORAFAS
Mc Graw-Hill.

-THE COMPUTER COMES OF AGE
R. MOREAU

-CONCEPTOS DE PROGRAMACION DE LOS ORDENADORES
J. DU ROSCAT

-HISTORY OF PROGRAMMING
WEXELBLAT

-PROGRAMMING LANGUAGES
SETHI

-LSA (LINC SYSTEMS APPROACH)
METODOLOGIA LINC PARA DESARROLLO DE SISTEMAS.
UNISYS DE MEXICO

-INFORMIX
POWERFUL PRODUCTS FOR SOPHISTICATED DATA MANAGMENT
UNISYS DE MEXICO

-ORACLE RELATIONAL DATABASE MANAGMENT SYSTEM
UNISYS DE MEXICO

-MAPPER. EL AMBIENTE FLEXIBLE PARA DESARROLLO DE APLICACIONES
UNISYS DE MEXICO

-THE ALLY SYSTEM.
OPEN DISTRIBUTED TRANSACTION-BASED APPLICATION DEVELOPMENT
UNISYS DE MEXICO