VI.1 GENERADOR DE SISTEMAS COMO NÚCLEO DE UN SISTEMA EVOLUTIVO

*Fernando Galindo Soria**

INTRODUCCIÓN

En este trabajo se presenta el proceso para construir y usar una herramienta para el desarrollo industrial de sistemas de información.

Para lo cual, se considera que la arquitectura de un sistema de información típico (nómina, compilador, software educativo, sistema experto, etc.) se basa en tres grandes componentes: datos, procesos y estructura del sistema (figura 1).

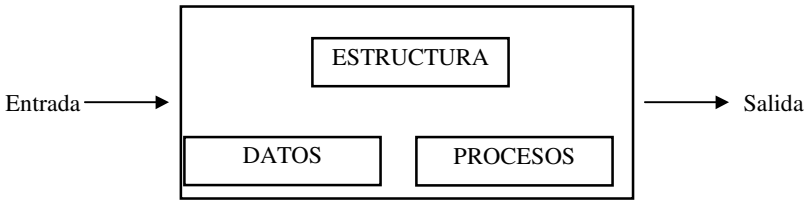


Figura 1. Componentes de un sistema de información

Con la característica de que por lo común cuando cambia algún elemento de una de las componentes, las otras no necesariamente se alteran (si cambian los datos, no necesariamente tienen por que cambiar los procesos o la estructura), por lo que, desde hace tiempo se ha tendido a desarrollar herramientas en las cuales sea relativamente fácil modificar una componente sin tocar las otras (por ejemplo, los sistemas manejadores de bases de datos), en particular aquí se presenta una herramienta de este tipo, conocida como Núcleo de un Sistema Evolutivo (figura 2).

---

* Fernando Galindo Soria escribió este trabajo en enero de 1990 cuando estaba en la UPIICSA del IPN y lo presentó como Conferencia Magistral y publicó en las memorias del Teccomp90, en la Cd. de México en Febrero de 1990.

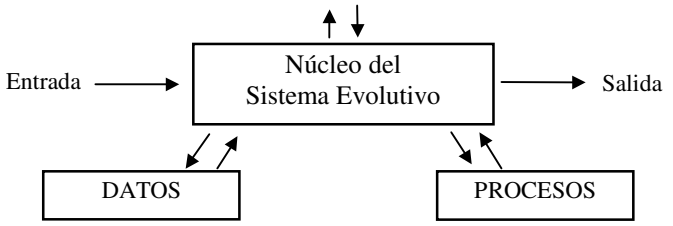


Figura 2. Componentes del Núcleo un Sistema Evolutivo

Donde el Núcleo del Sistema Evolutivo es un programa de propósito general al cual se le dan como entradas los componentes de un sistema de información específico y es capaz de dar solución a los requerimientos, sobre este sistema particular.

Por ejemplo, si se tiene un sistema de información orientado al cálculo de estadísticas (figura 3).

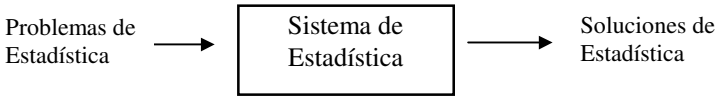


Figura 3. Cálculo de estadísticas

La arquitectura general del sistema se muestra en la figura 4.

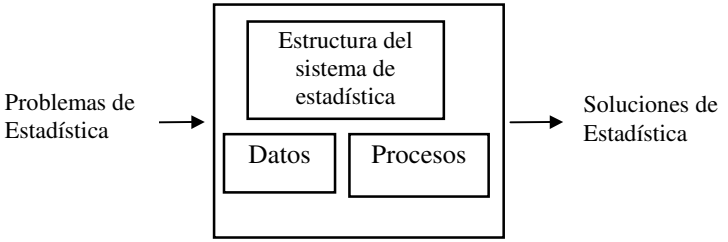
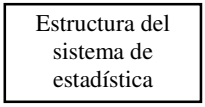


Figura 4. Arquitectura del sistema de estadísticas

y en particular, al integrarlos como Núcleo del Sistema Evolutivo queda como se aprecia en la figura 5.



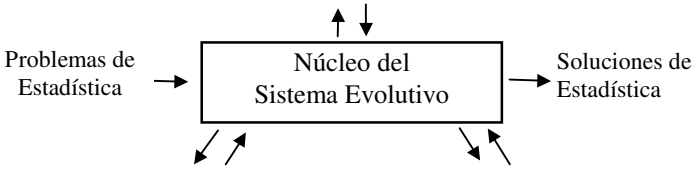


Figura 5. Sistema de estadísticas integrado al Núcleo del Sistema Evolutivo

Las ventajas de un enfoque de este tipo se presenta por el hecho de que al tener el sistema de información distribuido en sus componentes, es relativamente fácil realizar modificaciones y actualizaciones sobre el sistema, ya que, si por ejemplo, cambia el orden en que se resuelve un problema, lo único que se tiene que cambiar es la estructura del sistema, sin tocar ni los datos ni los procesos.

Además y dado que para un área de aplicación específica (nomina, graficación, estadística) las rutinas involucradas son normalmente las mismas y sólo cambia el orden en que se aplican, es relativamente fácil generar una gran cantidad de sistemas concretos, ya que, al cambiar la estructura (orden en que se aplican los procesos sobre los datos) automáticamente se cambian los sistemas, por ejemplo, en el área de estadística existe un conjunto específico de rutinas con las cuales se cubren muchos de los requerimientos, por lo que, utilizando las mismas rutinas pero en distinto orden se pueden obtener resultados diferentes.

Por otro lado, una herramienta de este tipo es de propósito general ya que si se cambian las componentes de entrada, con el mismo núcleo se pueden obtener los resultados para un nuevo sistema, por lo que, sólo es necesario programar una sola vez el núcleo.

Los principales problemas que se presentan con este enfoque son:

1) ¿Cómo se encuentran y/o modifican las componentes de un sistema de información? (diseño lógico).

2) ¿Cómo se construye un Núcleo de Sistema Evolutivo? (diseño físico).

Por lo que, en lo que sigue se describen algunas de las soluciones a estos problemas.

I Diseño Lógico: Obtención de las Componentes de un Sistema de Información

Una primera dificultad que se presenta para desarrollar este enfoque se encuentra en el hecho de que en general se siguen desarrollando los sistemas basándonos en la idea tradicional de Entrada-Proceso-Salida, figura 6.

Figura 6. Modelo tradicional de los sistemas de información

Por lo que cuando se desarrolla un sistema normalmente no nos preocupamos de donde están los datos o cual es la estructura del sistema, por lo que, como primer paso se mostrará como detectar las diferentes componentes de un sistema, de tal manera que exista una independencia relativa entre los datos, procesos y estructura.

Por ejemplo, en la oración:

Dame la regresión de X * 2 + Y, Z

es relativamente fácil detectar estas componentes (figura 7), donde la estructura está dada por las líneas que indican el orden en que se aplican los procesos sobre los datos, las acciones y los datos se indican mediante sus unidades léxicas respectivas (a, d).

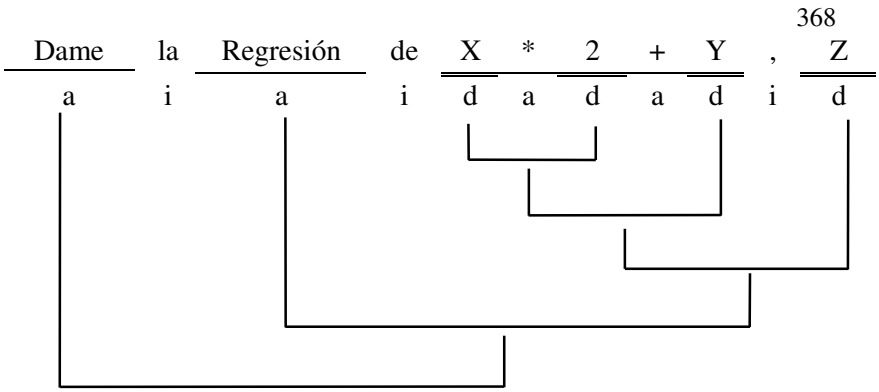


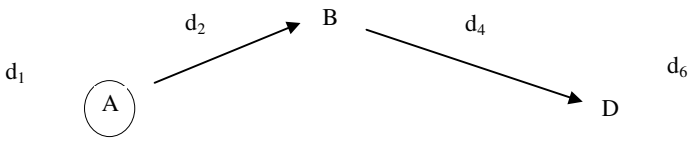
Figura 7. Componentes de un sistema en una oración

En general ya que se acostumbra uno, es relativamente fácil encontrar las componentes de un sistema de información.

A continuación se muestra como dado cualquier sistema de información se pueden obtener sus componentes (procesos, datos y estructura) y en particular se describe como encontrar las componentes de un sistema descritos mediante Diagramas de Flujo de Datos (DFD) (una de las principales herramientas usadas para describir sistemas).

1 OBTENCIÓN DE LOS COMPONENTES DE UN SISTEMA A PARTIR DEL DIAGRAMA DE FLUJO DE DATOS

El DFD es una herramienta utilizada comúnmente para describir el flujo de datos dentro de un Sistema. Por ejemplo, en el DFD de la figura 8, el dato dl entra al proceso A que genera los datos d2 y d3, los cuales entran respectivamente a los procesos B y C, y estos generan a su vez los datos d4 y d5, quienes finalmente entran al proceso D el cual genera el dato d6 .



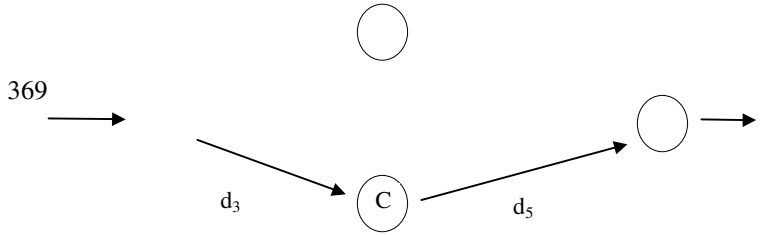


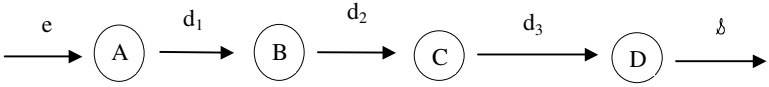
Figura 8. Un diagrama de flujo de datos

Dado un DFD es relativamente fácil encontrar sus componentes como se ve en los ejemplos que siguen:

1) Dado el DFD

tiene como datos (e, ß), procesos (A), estructura (S → e A ß). (El mecanismo para representar la estructura del sistema se puede interpretar como: el sistema S está formado por el dato e seguido del proceso A y terminando con el dato ß).

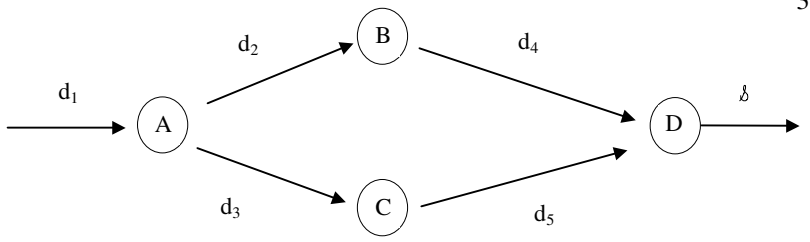
2) Dado el DFD:



sus componentes son: datos (e, d1, d2, d3, S), procesos (A, B, C, D), estructura (S → e A B C D ß).

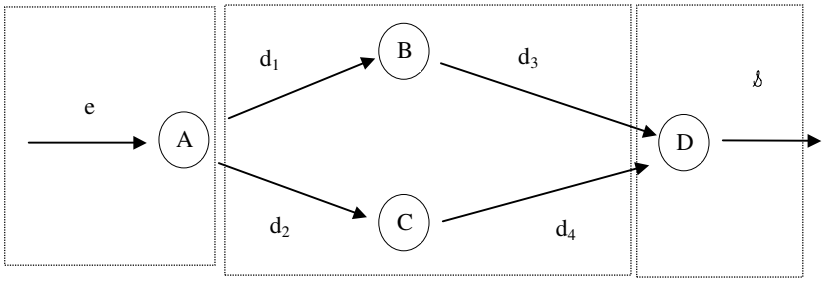
(es decir, el sistema S está formado por el dato e seguido de la ejecución de los procesos A, B, C, D y terminando con el dato S)

3) Dado el DFD:



En este caso el problema de la obtención de la estructura se complica ya que la secuencia no es lineal por lo que se aplica un método para transformar el diagrama (red) en un árbol (existen muchos métodos) y en particular se aplicará el método más sencillo, consistente en analizar la red de izquierda a derecha y de arriba abajo buscando formar "módulos" que engloben diferentes subproblemas.

Por ejemplo; en el problema anterior se puede analizar de izquierda a derecha quedando tres grandes submódulos:



X

de donde:

$S \rightarrow e\ A\ X\ D\ ß$

donde el submodulo X se puede ver como una rutina independiente (que se analiza de arriba-abajo)

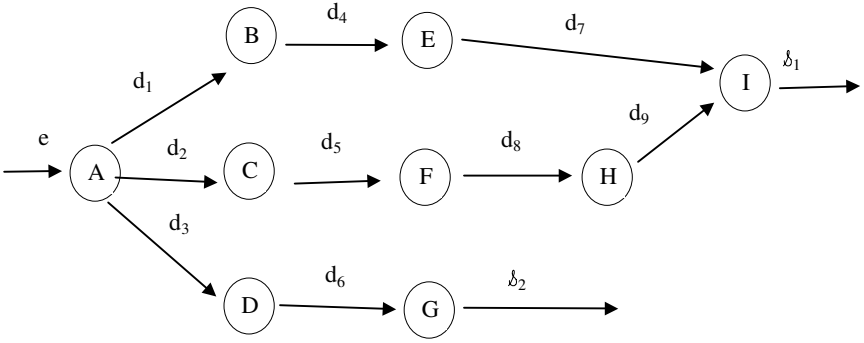
$X \rightarrow B\ C$

de donde la estructura del sistema es:

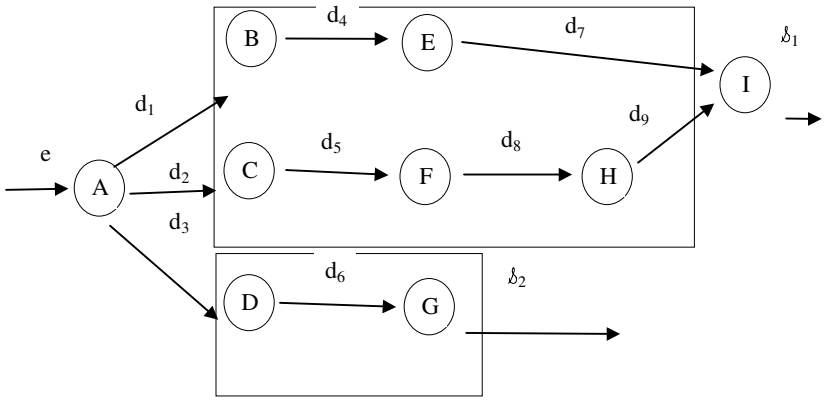
$S \rightarrow e\ A\ X\ D\ \delta$

$X \rightarrow B\ C$

4) Dado el DFD:



X



Y

$S \rightarrow e\ A\ X\ \delta_1 Y\ \delta_2$

$\Downarrow$

$S \rightarrow e\ A\ X\ \delta_1 Y\ \delta_2$

$X \rightarrow X_1\ X_2$

$Y \rightarrow D\ G$

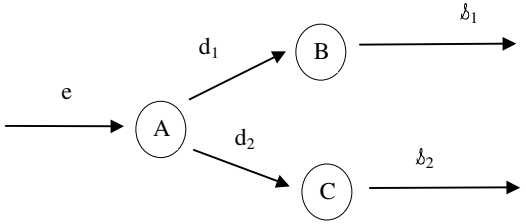
$\Downarrow$

$S \rightarrow e\ A\ X\ \delta_1 Y\ \delta_2$

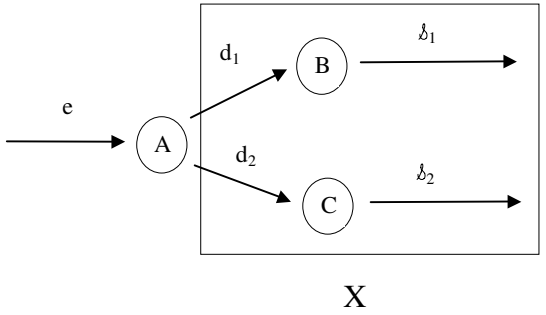
$X \rightarrow X_1\ X_2$
$X_1 \rightarrow B\ E$
$X_2 \rightarrow C\ F\ H$
$Y \rightarrow D\ G$

En los ejemplos anteriores se supuso que en el DFD se tienen que ejecutar todos los procesos y simplemente se transformó la red del DFD en un árbol, sin embargo, existen algunos casos especiales en los cuales se tiene más de un posible camino y se debe ejecutar uno u otro pero no los dos (como en el IF-THEN-ELSE de la programación estructurada).

Por ejemplo, en el DFD siguiente:



la disyunción puede no indicar procesos en paralelo sino que se ejecute el proceso B o C pero no los dos (esto depende del análisis del sistema). En este caso el método para obtener la estructura es parecido, por lo que el DFD anterior se divide como se muestra:



que se representa como:

$S \rightarrow e\ A\ X$

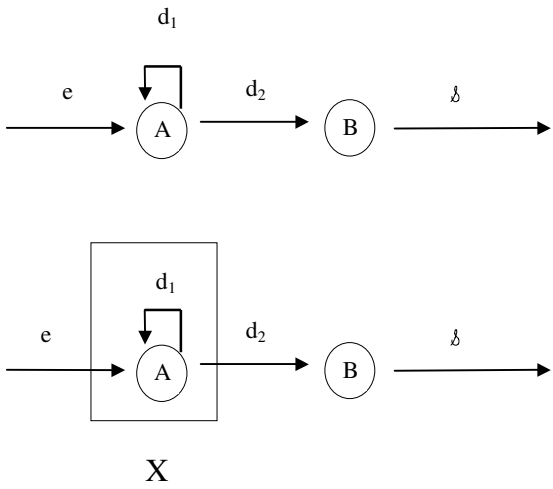
donde en la rutina X se tiene que indicar que existen dos opciones quedando:

$$B \rightarrow B\ \delta_1 \mid C\ \delta_2$$

donde la línea $\mid$ indica ó, por lo que se lee, la rutina X ejecuta B S_1 ó C S_2 de donde la estructura completa es:

$$S \rightarrow e\ A\ X$$
$$X \rightarrow B\ \delta_1 \mid C\ \delta_2$$

Otro problema específico que se presenta es cuando se tienen ciclos dentro del DFD, por ejemplo:



En este caso se sustituye por una estructura de tipo recursivo de la forma:

$$S \rightarrow e\ X\ B\ \delta$$
$$X \rightarrow A\ d_1\ X \mid d_2$$

que se lee, el sistema S toma un dato e llama a X ejecuta a B y genera S, donde X ejecuta A, si continua el dato d_1 vuelve a llamar a X (proceso recursivo) ó termina el ciclo si llega el dato d_2.

Como se puede ver, el problema de encontrar los componentes de un sistema de información a partir de un DFD, se reduce al

problema de encontrar la estructura del sistema (ya que los datos y procesos se obtienen directamente del DFD) y este se reduce al problema de transformar una red a un árbol y en particular a la representación de las estructuras clásicas de la programación estructurada:

1) Secuencia:

$S \rightarrow A\ B\ C\ D$

2) Decisión:

$S \rightarrow A \mid B$

3) Repetición:

$S \rightarrow A\ S \mid B$

En general mediante el método de diseño lógico descrito anteriormente, es relativamente fácil obtener las componentes de un sistema de información (figura 9).

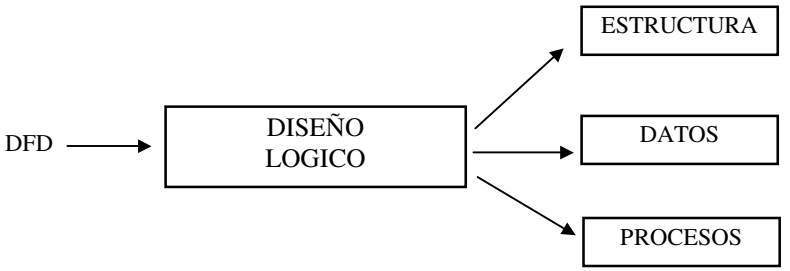
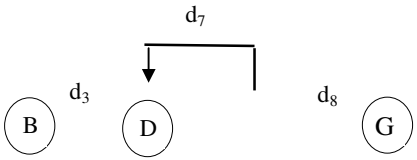
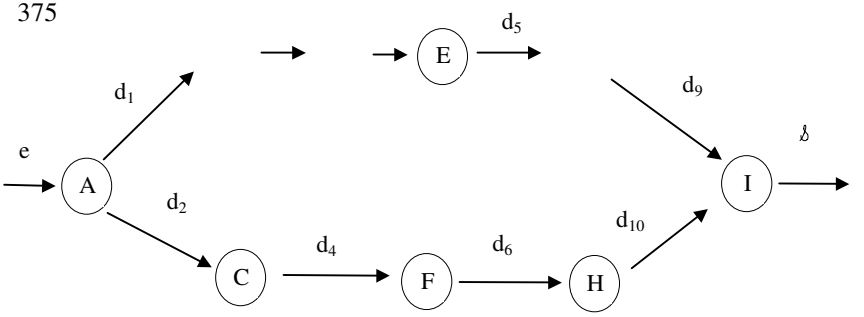


Figura 9. Diseño Lógico para obtener las componentes del sistema de información

Aplicando lo anterior a otro ejemplo más complejo se tiene que dado el DFD:





se obtienen los componentes siguientes:

1) Datos. e, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, S

2) Procesos. A B C D E F G H I

3) Estructura.

$S \rightarrow e\ A\ X\ I\ ß$
$X \rightarrow d_1\ X_1 \mid d_2\ X_2$
$X_1 \rightarrow B\ X_3\ G$
$X_3 \rightarrow d_3\ B\ e\ X_3$
$S \rightarrow C\ F\ H$

2 DISEÑO FÍSICO: CONSTRUCCIÓN DEL GENERADOR DE SISTEMAS

Ahora nuestro siguiente problema es construir un programa X, al cual se le den como entrada los procesos, datos y estructura del sistema de información y sea capaz de responder a los requerimientos de este sistema de información. Programas de este tipo se han construido desde hace bastante tiempo pero con la característica de ser extremadamente complejos y difíciles de desarrollar. En particular a finales de los 70's y principios de los 80's dos trabajos de investigación independientes coordinados por Vicente López Trueba, el primero y por Lino Díaz Bello, el segundo, desarrollaron las bases de la herramienta que se va a presentar a continuación, en el primer grupo se construyeron un conjunto de generadores de programas, cada vez más completos y que en sus últimas versiones son de una simplicidad extrema, por su parte, Lino Díaz, desarrollo un método para construir

compiladores basado en el manejo de gramáticas y en cual para obtener un nuevo compilador, lo único que se requiere es cambiar la gramática.

Esquemáticamente, estos dos modelos se pueden ver en las figuras 10 y 11.

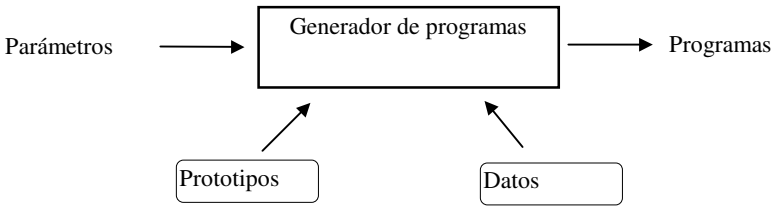


Figura 10. Un modelo de un Generador de programas

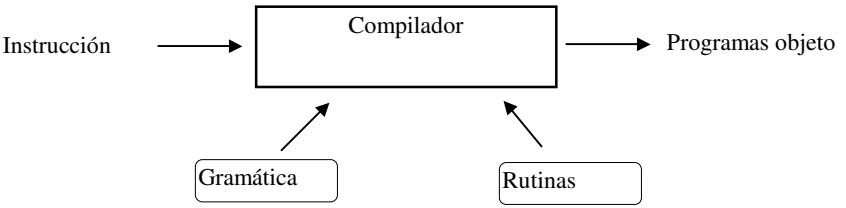


Figura 11. Un modelo de un Compilador Generalizado

En el caso de la figura 10, los prototipos son programas parametrizados de tal manera que sirven como molde para generar un programa específico, y en el de la figura 11, la gramática es un mecanismo con el cual se representa la estructura de un lenguaje.

Integrando las dos herramientas anteriores y generalizando con el fin de representar la estructura de cualquier sistema, se obtiene la estructura de un generador de Sistemas (figura 12).

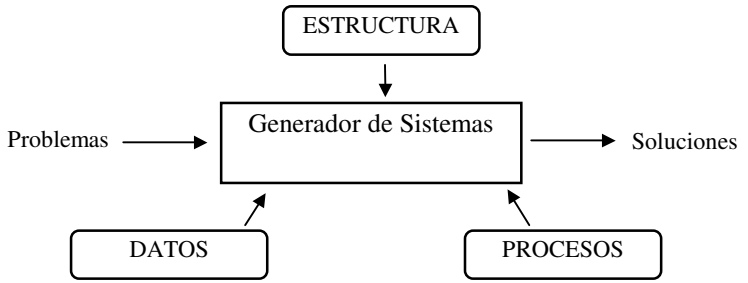
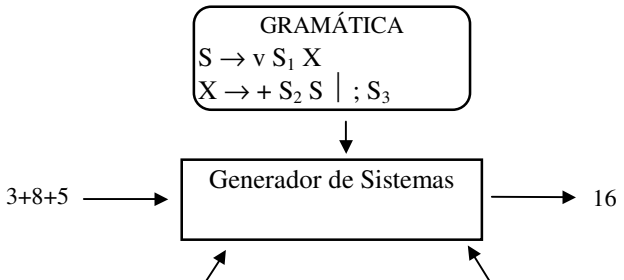


Figura 12. Estructura de un generador de sistemas

La gramática es precisamente la estructura del sistema, por lo que nuestro problema es cómo lograr que la computadora integre un conjunto de procesos y datos de acuerdo a una estructura, con el fin de resolver un problema dado.

Cuando se analiza la estructura de un sistema o gramática se detecta que en ésta se encuentran involucrados los componentes, por lo que, lo único que se necesita construir es un mecanismo capaz de ir analizando los diferentes elementos de la gramática de acuerdo a los elementos de entrada y en su momento llame a los procesos y datos en el orden requerido.

Por ejemplo, si se quiere ejecutar una suma de variables (este es uno de los problemas más trillados) un sistema de este tipo tiene la arquitectura general mostrada en la figura 13.



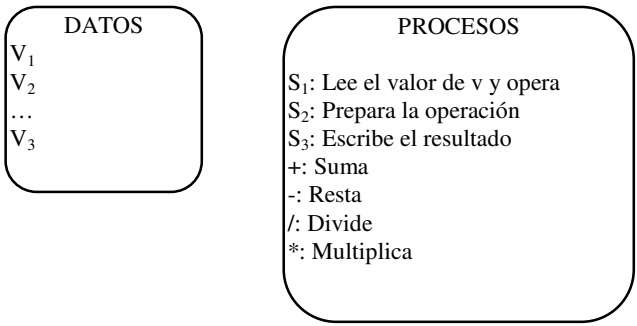


Figura 13. Una instancia del generador de sistemas de suma de variables

Por otro lado, si se quiere ejecutar operaciones aritméticas en general lo único que se cambia es la gramática y el sistema debe de funcionar (figura 14).

Como se puede ver, el gran problema del generador es que debe ser capaz de ir siguiendo la gramática de acuerdo con las entradas y llamando a los procesos y los datos con el fin de obtener los resultados.

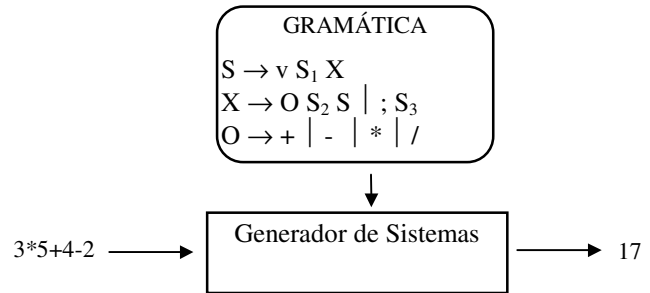
Por lo que el problema se centra en el análisis de la gramática.

Si se observa una gramática típica (como las obtenidas durante el diseño lógico) se encuentra que esta compuesta de tres tipos de elementos:

1) Datos o señales de entrada (también se les conoce como elementos terminales).

2) Llamados a Procesos (también se les conoce como rutinas semánticas)

3) Rutinas de control o llamados a otros componentes de la estructura (también se les conoce como elementos no terminales).



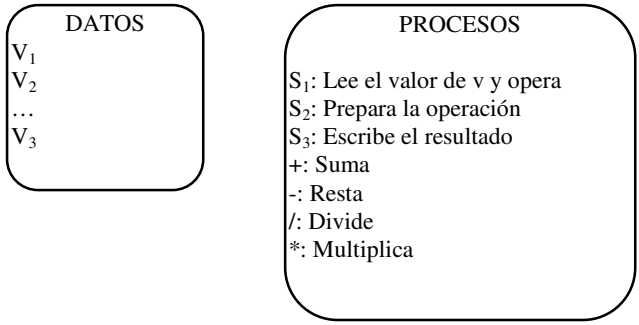


Figura 14. Una instancia del generador de sistemas extendido para varias operaciones

Por ejemplo, en la gramática:

$S \rightarrow v\ S_1\ X$

$X \rightarrow O\ S_2\ S \mid ; S_3$

$O \rightarrow + \mid - \mid * \mid /$

se tiene lo siguiente:

1) Datos o señales de entrada:

v ; + - * /

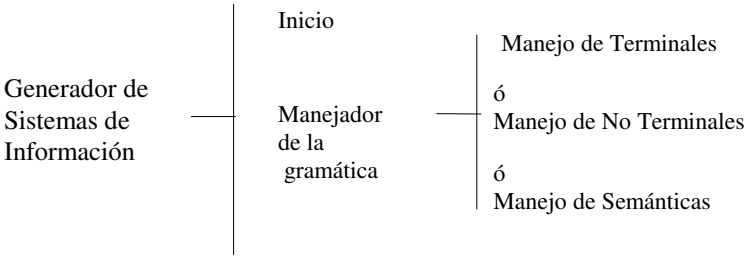
2) Llamados a procesos o Rutinas Semánticas:

$S_1\ S_2\ S_3$

3) Rutinas de Control:

S X O

Por lo que, el esquema general del generador de sistemas es el mostrado en la figura 15.



Termina

Figura 15. Esquema general del generador de sistemas

Y el diseño detallado del generador de sistemas es el mostrado en la figura 16.

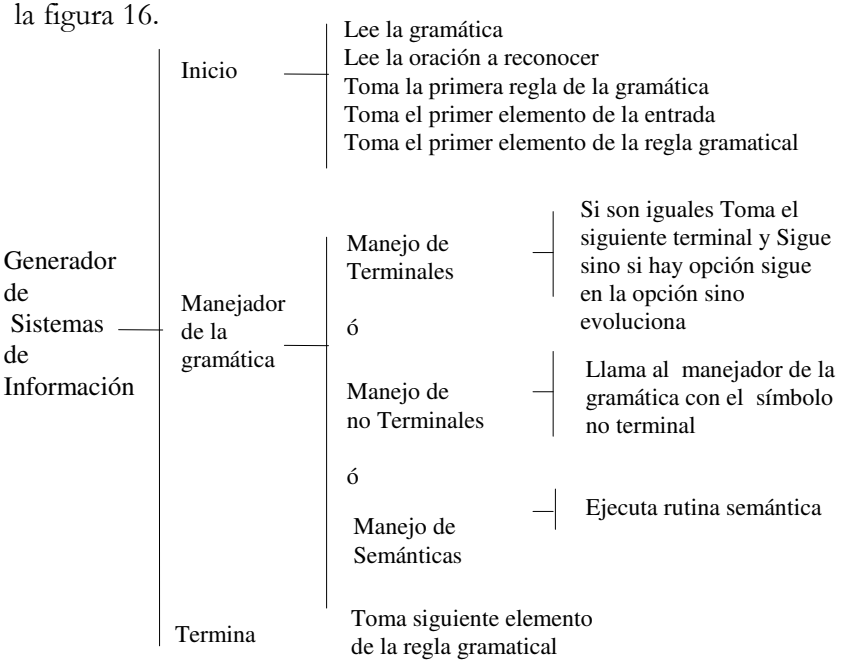


Figura 16. Esquema detallado del generador de sistemas

Como se puede ver, es un programa pequeño (que normalmente se programa en una o dos hojas) pero sumamente poderoso ya que en la mayoría de los casos basta cambiar la gramática para tener una aplicación completamente diferente, sin necesidad de cambiar la lista de procesos en la mayoría de las áreas (por lo común en la mayoría de las áreas de aplicación ya establecidas se cuenta con una gran cantidad de rutinas y cuando "programamos" lo único que hacemos es "pegar" estas rutinas en distintos ordenes).

Lo interesante de este enfoque, es que si por algún motivo se cambia la estructura no es necesario volver a programar el sistema

de información sino, únicamente se cambia el contenido del archivo donde aparece 1a estructura y el sistema sigue funcionando; lo mismo se puede decir si cambian los datos o los procesos; con lo cual, al tener separados los tres componentes y un mecanismo computacional que los integra es relativamente fácil desarrollar nuevos sistemas de información, ya que directamente del análisis se pueden encontrar estos componentes y almacenarse en tablas que va llamando el Generador de Sistemas.

II Diseño Físico: Núcleo de un Sistema Evolutivo

Finalmente se ve cómo se puede integrar esta herramienta como núcleo de un Sistema Evolutivo con el fin de que también se encuentren los componentes en forma automática a partir de una interacción directa con el ambiente o área problema.

Si se observa el esquema de la figura 17.

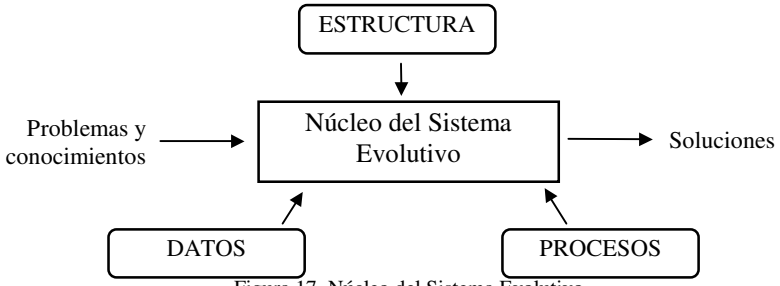


Figura 17. Núcleo del Sistema Evolutivo

La diferencia principal entre el generador de sistemas y el núcleo de un Sistema Evolutivo es que en este último caso, existe una interacción en los dos sentidos entre el núcleo y las componentes del sistema de información, de tal manera, que si por algún motivo, cambia una rutina (proceso), el Sistema Evolutivo debe ser capaz de realizar el cambio en tiempo real y seguir funcionando, y lo mismo se puede decir si cambia la estructura del sistema de información o los datos.

Para lograr lo anterior, ya se cuenta con una gran cantidad de métodos y herramientas, pero en esencia la idea es que cuando el núcleo encuentra un elemento (dato, proceso) o alguna forma estructural que no reconoce en lugar de mandar un mensaje de error léxico o sintáctico, entre en un proceso de diálogo donde busca información que le permita actualizar las componentes o detectar que realmente se presentó un error.

Por ejemplo; si se está pidiendo ejecutar un proceso que no conoce, trataría de que directamente se le "enseñara" a resolver el nuevo problema (pidiendo por ejemplo el código en PASCAL o C del nuevo proceso), por otro lado, si lo que se encuentra es una estructura gramatical no conocida, verifica que sea una estructura valida y en ese caso modificaría la gramática con el fin de aceptar la nueva estructura (ya existen muchos métodos orientados a este fin, englobados en el área de Inferencia Gramatical).

CONCLUSIÓN

El área de los Sistemas Evolutivos es un área relativamente nueva, pero con un campo de acción creciente, sin embargo, aún es poco conocida, por lo que, en este trabajo se describió una de las herramientas que conforman a un sistema evolutivo, pero buscando su generalización sin necesidad de ser un experto en el tema, de tal manera que al contar con una herramienta de este tipo pueda desarrollar sistemas de información en forma industrial, y se motive a continuar investigando sobre este tema y propicie su desarrollo.