DISEÑO DE COMPUTADORAS

Ulises Agüero*
Juan Carlos Gómez*

RESUMEN

*En este artículo se enfoca el diseño de **hardware** y **software** utilizando un mismo modelo abstracto. Sin duda una de las diferencias principales entre el desarrollo de **hardware** y **software** es que, en el primero, el objeto construido es tangible, mientras que en el segundo, el producto es abstracto. Esto provoca que, en **software**, no pueda establecerse una delimitación clara de la frontera entre diseño e implantación. El énfasis se ha orientado principalmente al diseño de **software** por ser considerado de mayor interés nacional.*

INTRODUCCION

La palabra **diseño** está asociada tanto con un objeto como con el **proceso** que genera este objeto. Se ha asegurado que cualquier persona que cambie situaciones existentes en situaciones deseables está diseñando[8]. Tal caracterización se ajusta mejor a nuestra noción de "solución de problemas" que a la de diseño, mientras que el diseño es un caso particular de solución de problemas.

En este artículo, entenderemos el proceso de diseño como **un proceso de especificación de las características de un objeto para que satisfaga ciertos requerimientos**. La especificación generada es el diseño. Esta definición está de acuerdo con el sentir de muchos investigadores por ejemplo Alexander[1], Freman y Waseman[4] y Simon[8]. Nótese que si la especificación es la de un objeto físico, como por ejemplo una computadora, la construcción (o implantación) del objeto no es parte del proceso de diseño.

En el diseño de computadoras, los aspectos por considerar se clasifican en dos grandes áreas: ***hardware*** y ***software*** aunque en ocasiones se agrega una tercera denominada ***firmware***. Estas áreas están íntimamente relacionadas entre sí en lo que concierne al proceso de diseño. Las decisiones de diseño de *hardware* deben tomar en cuenta las implicaciones potenciales con respecto al *software* y viceversa.

En general, el diseño de *hardware* se refiere a la especificación, en un nivel de abstracción dado, de los **sistemas electromecánicos** que conforman una computadora. El diseño del *software* se refiere a la especificación de los sistemas que no son de naturaleza electromecánica y que son conocidos como **sistemas de programación o de aplicación** y finalmente, el término de *firmware* se emplea para denotar sistemas especializados en el control del *hardware* de las computadoras por medio de un tipo de programación de "bajo nivel" llamado **microprogramación.**

En las secciones restantes introduciremos un modelo general de diseño que luego discutiremos con respecto al diseño de *hardware* y principalmente, del *software* de una computadora.

UN MODELO DE DISEÑO

La Figura No. 1 (basada en la referencia 8) muestra los componentes del contexto de un problema asociado con un proceso de diseño.

* Departamento de Computación. Instituto Tecnológico de Costa Rica

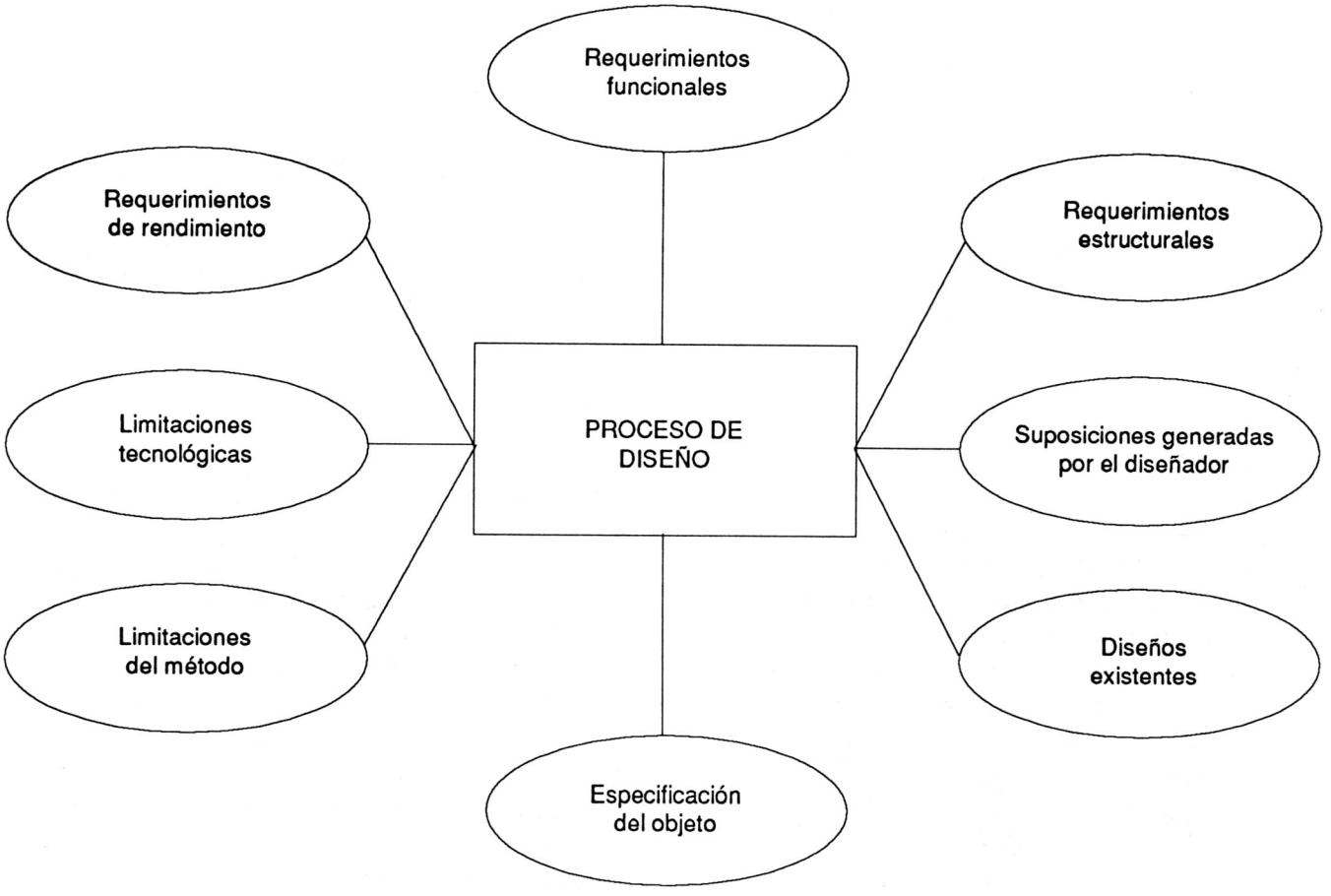

FIGURA No. 1. El contexto de un problema en relación con un proceso de diseño.

Los **requerimientos funcionales** se refieren a la especificación de los resultados que el objeto debe producir, dadas ciertas entradas, mientas que los **requerimientos de rendimiento** se refieren a medidas de eficiencia sobre la forma en que los resultados son producidos.

Las **limitaciones del método** se refieren al marco de referencia que un diseñador debe utilizar cuando se ajusta a un método de diseño particular.

Los **diseños existentes** representan especificaciones de objetos que guardan alguna relación con el objeto que se está diseñando. Estos diseños sirven para reducir el tamaño del universo de enfoques posibles que un diseñador debe considerar.

Los **requerimientos estructurales** especifican demandas con respecto a las características "estáticas" del objeto; es decir, la organización de los componentes del objeto y las interrelaciones entre estos componentes.

Las **suposiciones** que genera un diseñador le permiten simplificar el proceso de toma de decisiones, pero a la vez influyen en la forma en que el objeto puede utilizarse. Si una de tales suposiciones es violada al hacer uso del objeto, esto posiblemente funcionará en forma anormal o inadecuada.

La **especificación del objeto** puede tomar muchas formas diferentes, dependiendo de la naturaleza del objeto y del nivel de abstracción presente en una etapa de diseño dada. Posiblemente, la especificación inicial conste solamente de un conjunto de objetivos descritos informalmente, mientras que las especificaciones posteriores incluyen más detalle, descrito en algún lenguaje especializado.

El **proceso de diseño** en sí puede caracterizarse según se muestra en la Figura No. 2. Los sistemas descritos en la figura no corresponden necesariamente a sistemas computadorizados.

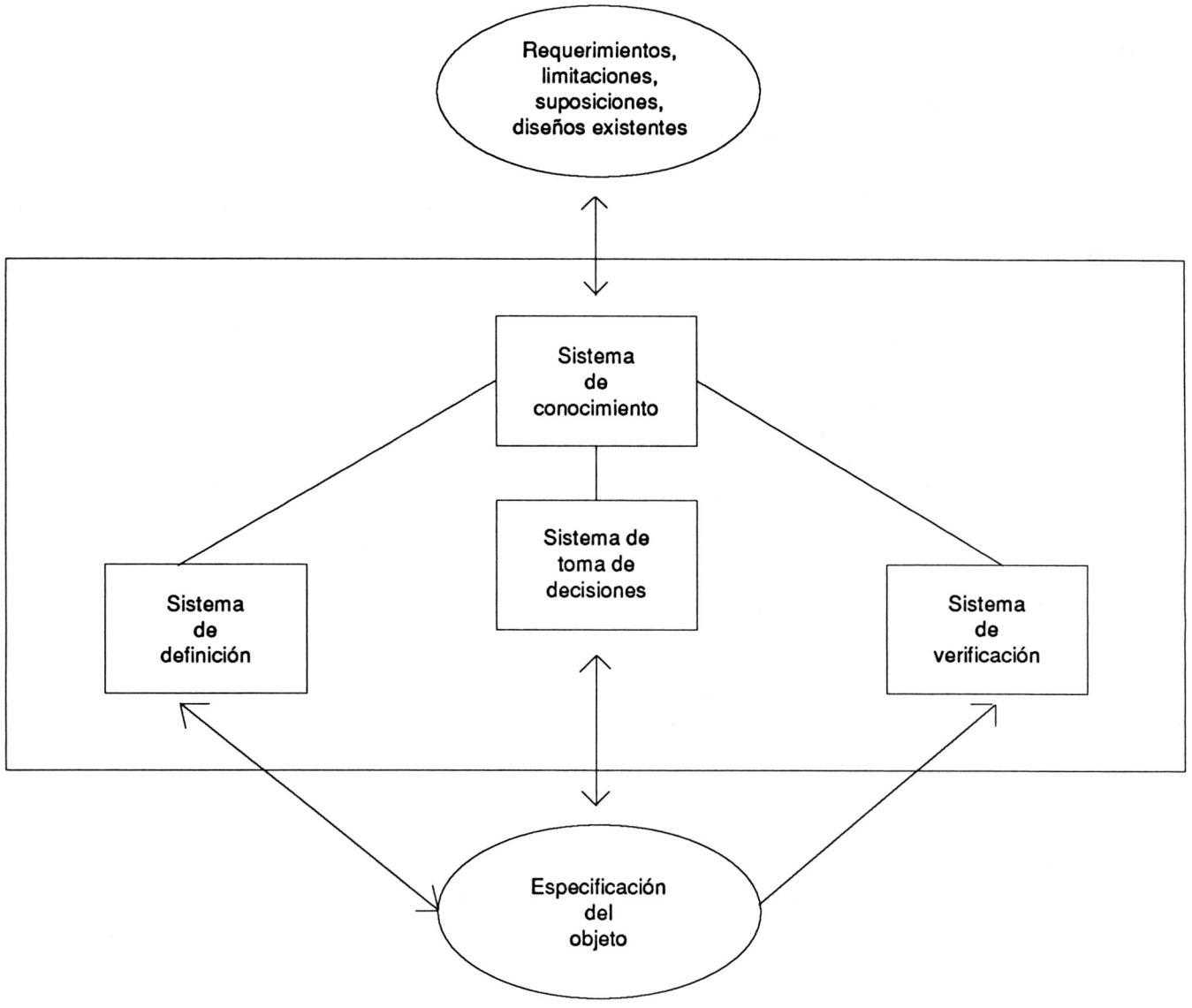

FIGURA No. 2. Un modelo del proceso de diseño.

El **sistema de conocimiento** le permite al diseñador procesar toda la información relevante al contexto del problema, tal como el conjunto de **requerimientos, limitaciones, suposiciones** y **diseños existentes** y organizar el conocimiento generado durante el proceso de diseño.

El **sistema de toma de decisiones** es el motor del proceso de diseño al implementar un método de diseño dado. Basándose en la especificación actual del objeto, este sistema decide si se debe o no activar el sistema de conocimiento para tener una mejor visión del problema o para establecer suposiciones. También decide cuándo activar el **sistema de definición** para modificar o refinar la especificación del objeto. Finalmente, el sistema de toma de decisiones utiliza el **sistema de verificación** para cerciorarse de que la especificación del objeto es aceptable.

EL DISEÑO DEL *HARDWARE* DE UNA COMPUTADORA

Al diseñar el *hardware* de una computadora debemos, al menos, tomar en cuenta los siguientes aspectos interrelacionados:

- Las áreas de aplicación donde se pretende utilizar la computadora

- Las condiciones climatológicas bajo las cuales se pondrá en funcionamiento
- El tipo de *software* que será desarrollado
- Las restricciones y beneficios que, para el desarrollo de *software*, genera una decisión de diseño dada
- Las características de los potenciales usuarios (y programadores) de la computadora
- El costo que representa el que una característica particular esté presente en la computadora
- Las propiedades tecnológicas de los componentes electromecánicos de que se dispone para crear el diseño
- Las características estructurales, funcionales y de rendimiento de otras computadoras que hayan sido diseñadas con objetivos similares
- Los métodos más apropiados para satisfacer los objetivos de diseño.

Las consideraciones arriba mencionadas representan el contexto del problema enfrentado, según se discutió previamente con respecto a la Figura No. 1.

De acuerdo con el modelo de la Figura No. 2 el diseño (es decir, la especificación del objeto) puede ser descrito en lenguajes especializados para la especificación del *hardware*, denominados HDLs (*Hardware Description Languages*)[2]. Estos lenguajes aumentan la precisión con que se plantea el diseño y permiten experimentar con el diseño mismo, previo a la implantación.

En el contexto del diseñador del *hardware*, el sistema de verificación se basa fundamentalmente en técnicas experimentales, sistemas deductivos basados en lógica, matemática y sistemas que permiten al diseñador razonar informalmente sobre las ventajas y desventajas del estado actual de la especificación.

El sistema de conocimiento de la Figura No. 2 se utilizaría para definir el contexto del problema, representar el conocimiento de "expertos" en el diseño de *hardware*, accesar este conocimiento y almacenar el nuevo conocimiento durante el proceso de diseño.

EL DISEÑO DEL *SOFTWARE* DE UNA COMPUTADORA

El proceso de diseño de *software* difiere de los procesos de diseño en otras áreas por la naturaleza de los objetos que se pretenden crear. Podríamos decir que estos objetos poseen una naturaleza intangible en el sentido de que el producto final no es percibido en forma directa. Por el contrario, estos productos los percibimos por sus resultados.

Si se construye una nueva computadora de acuerdo con su diseño propuesto, podemos tocarla, verla. Pero si diseñamos un sistema para control de tráfico aéreo no podríamos decir lo mismo.

El proceso de desarrollo de productos de *software* inicia con una idea de algo que se quiere, que se imagina, que pretende satisfacer un conjunto de necesidades y culmina con la transformación de esa idea en un conjunto de instrucciones –programas–, los cuales, al ser ejecutados por el *hardware* de un computador, transforman la idea en algo "viviente".

A continuación se discuten los elementos del modelo de diseño presentado en la Figura No. 2, desde el punto de vista de *software*.

Sistema de definición

Dado que el producto final del proceso de diseño es un "plano" del objeto que se va a crear, debemos, por lo tanto, identificar los elementos constituyentes de este plano. En el caso del *software*, los principales elementos son:

- Especificación de programas
- Especificación de interfaces
- Especificación de las representaciones de datos

Todos estos elementos conforman la especificación de un sistema.

Las **especificaciones de programas** informan sobre los procesos que serán ejecutados por el computador y que actuarán sobre un conjunto de datos para producir alguna salida. Las **especificaciones de las representaciones de datos** indican la forma en la cual los datos serán recibidos y almacenados en el computador y, por último, la **especificación de interfaces** indica la forma en que deben interactuar los diferentes programas, los programas con el *hardware* y *software* de la computadora y la forma de interacción del usuario con el sistema.

Las especificaciones se pueden lograr ya sea utilizando formas gráficas (e. g., diagramas de flujo y diagramas que muestren relaciones entre programas) o algún tipo de lenguaje, que puede ser infor-

mal (lenguaje natural) o imponer ciertas reglas o formalismos. En el caso de la especificación de programas, el lenguaje de especificación puede ser muy similar a un lenguaje de programación de computadoras.

El nivel de detalle de las especificaciones también es variable, sin embargo, debe ser tal que las otras personas involucradas en las etapas siguientes del proceso de desarrollo de *software* entiendan efectivamente lo que el diseñador desea, basado en los requerimientos definidos por los usuarios.

Hay dos aspectos relevantes en cuanto al sistema de definición. En primer lugar, el sistema de definición debe procurar la comunicación exitosa de las ideas del diseñador a los implementadores, y en segundo lugar, la experiencia ha dictado que es necesario un nivel de estandarización en este proceso de comunicación.

Sistemas de conocimiento y verificación

El proceso de desarrollo de *software* se inicia con una etapa conocida con el nombre de análisis, cuyo objetivo es determinar QUÉ se desea desarrollar mediante un análisis de necesidades potencialmente atendibles con un sistema automatizado basado en la computadora. Los generadores de estas necesidades pueden ser:

- Un grupo de personas o futuros usuarios del sistema
- Otros sistemas que requieren de un nuevo sistema para su funcionamiento; siempre con el objetivo de satisfacer las necesidades de algún usuario.

También es necesario recabar información sobre el medio (contexto) en que estará inmerso el sistema y sobre los problemas que originan estas necesidades.

Como resultado de esta actividad, se obtienen los requerimientos generales del sistema a menudo formulados como las principales entradas de datos necesarios para producir la información deseada, dado que existen un conjunto de limitaciones impuestas por el contexto. En este punto también se identifican las principales funciones que son necesarias para transformar los datos de entrada en la información de salida.

Con esto se establecen los objetivos, restricciones y un modelo conceptual del sistema.

Información valiosa que también debe recolectarse es sobre otros sistemas de naturaleza similar y sobre el nivel en que éstos cubren las expectativas.

El proceso de diseño inicia donde el análisis termina; sin embargo no existe una clara división entre la finalización e inicio de estas reglas. El primer paso en el proceso de diseño es el refinamiento de los resultados obtenidos en la etapa de análisis. Entonces, por medio del sistema de conocimiento, se tienen los mecanismos para almacenar y recuperar la información requerida para el diseño de *software*.

En esta etapa previa, la verificación es lograda confrontando el esquema conceptual del sistema con las necesidades y también comparándolo con sistemas similares.

Los siguientes pasos en el proceso de diseño son:

- Establecimiento de una estructura general del sistema
- Diseño detallado.

Para establecer una **estructura general del sistema** se requiere conocimiento de:

- El modelo conceptual del sistema
- Algoritmos existentes y piezas de *software* que eventualmente puedan ser utilizadas
- Más detalle sobre sistemas similares
- Detalle sobre los otros posibles sistemas con que interactuará el sistema
- Detalles sobre el sistema computacional.

La estructura general del sistema muestra:

- Las partes del sistema (a nivel funcional) y sus relaciones
- Ciertos algoritmos básicos que el sistema usará
- Las formas en que van a ser representados los datos
- La forma en que el usuario interactuará con el sistema (interfaz con el usuario)
- La forma de interacción del sistema con otros sistemas (interfaz con otros sistemas)
- Aspectos relacionados con la utilización de los recursos del sistema computacional

En este punto, también se debe verificar que efectivamente las partes constituyan el todo desea-

do, y que las características y objetivos del sistema se mantengan. Para realizar lo anterior es necesario operacionalizar la descripción del sistema, de manera que sea posible efectuar algún tipo de prueba.

La operacionalización puede ir desde descripciones informales hasta descripciones formales en lenguaje matemático. Para estas últimas se han planteado métodos formales, con fundamento en lógica y matemática, que permiten determinar la correctitud de los programas[5].

Nuevas tendencias para orientar el proceso de desarrollo de *software* han surgido como el desarrollo de prototipos. Por medio de éstos, se trata de involucrar al usuario en forma más directa en las etapas de análisis y diseño. Básicamente, lo que se hace es desarrollar un "cascarón" del sistema, implantado en la computadora, de tal forma que el usuario pueda determinar si sus necesidades están siendo cubiertas. En vista de que se desea minimizar la inversión de esfuerzo en el desarrollo de estos "cascarones", se provee al diseñador con herramientas que le permiten realizarlo en una forma rápida y ágil.

El campo de la verificación es todavía más amplio. Puede incluir la experimentación. Por ejemplo, suponga que se desea desarrollar un sistema computacional que sirva como complemento a un plan educativo. En este caso, la interfaz con el usuario es fundamental. El diseñador puede entonces proponer y realizar experimentos utilizando herramientas estadísticas para refutar o aceptar hipótesis en las que está basado su diseño.

La operacionalización puede estar orientada no solo a los procesos sino también a los datos. Los llamados tipos de datos abstractos son una operacionalización en este sentido. Los tipos de datos abstractos generalizan el concepto de operador que actúa sobre operandos, objetos de algún universo; dicho con otras palabras, un tipo de dato abstracto es un modelo matemático con un conjunto de operaciones definidas sobre ese modelo. Los tipos de datos abstractos pueden ser especificados formalmente, de manera que pueda ser aplicada alguna prueba formal[6].

En el **diseño detallado** se refinan cada una de las partes; subsecuentemente se proponen diferentes opciones y se elige alguna, la cual será implantada. En este punto la información requerida es:

- La descripción estructural del sistema
- Más detalles sobre el sistema computacional.

Como resultado se obtienen las especificaciones de procesos y representaciones de datos que conformarán el sistema, así como la especificación de las interfaces.

Sistema de toma de decisiones

El sistema de toma de decisiones involucra una estrategia para encarar el proceso de diseño y como tal se convierte en el centro motor del mismo.

Varias estrategias han sido planteadas (v. gr. 7, 9); las más significativas son la *Top-Down* y la *Botton-Up*.

En la **estrategia *Top-Down*** el proceso de diseño va de lo general a lo particular. Dado un problema se procede a descomponerlo en subproblemas, buscando soluciones a éstos. Si los subproblemas son todavía muy grandes se continúa con este proceso de descomposición y conforme se avanza en dicho proceso se va logrando paulatinamente una solución más detallada.

La **estrategia *Botton-Up*** va de lo particular a lo general. En las primeras etapas de diseño se atacan problemas específicos y en las últimas se contempla la globalidad del problema.

Ambas estrategias imponen una forma en que se debe recolectar la información (sistema de conocimiento), la forma en que se debe proceder en la verificación (sistema de verificación) y por último la forma en que, por medio del sistema de definición se va obteniendo la especificación del objeto.

La frontera entre el diseño e implantación de *software*

Las especificaciones de programas son descripciones de las instrucciones que conforman un proceso. Estas descripciones se pueden lograr por medio del lenguaje natural o por medio de algún lenguaje o formalismo de especificación, en el cual se imponen ciertas reglas, por consiguiente pueden estar escritas en un lenguaje de programación de computadoras.

Sin embargo, generar las especificaciones de los programas utilizando un lenguaje de programación es parte de la labor de los implantadores de *software*. Por lo tanto, podemos identificar varias etapas para el logro de las especificaciones de programas. En cada una de éstas se incluye más detalle en la especificación y el máximo grado se

obtiene cuando se utiliza algún lenguaje de programación, de tal forma que esa especificación, con un gran nivel de detalle –programa –, pueda ser ejecutado por el sistema computacional.

Con base en lo expuesto se puede llegar a la conclusión de que en *software*, la frontera entre diseño e implantación es difusa. Valdría entonces formular la siguiente pregunta: ¿Si los implantadores de *software* lo que hacen es terminar la especificación de programas, dónde efectivamente, se inicia la construcción del objeto planeado?

En *software*, en el estado actual de la computación, la construcción del objeto se realiza en forma automática por medio de los componentes del sistema computacional. Y estos componentes son el *hardware* del computador y piezas de *software*, conocidas con el nombre de **traductores** que permiten transformar el programa en un conjunto de señales electrónicas, que en última instancia son lo único que pueden "entender" los componentes que conforman el *hardware.*

Esta característica de no poder distinguir en forma clara la frontera entre diseño e implantación puede inducirnos a pensar, en forma errónea, que no es necesario hacer un alto en el camino para evaluar el diseño u obviar el proceso de diseño como tal, concentrando nuestros esfuerzos en una implantación "a ciegas" y relegando a un segundo plano los otros elementos del producto de diseño y todos los aspectos metodológicos y administrativos inmersos en la labor.

En octubre de 1968, en una conferencia sobre Ingeniería de *Software*[3] la comunidad de científicos en computación reconoció la existencia de una crisis en el desarrollo de *software*, así como la importancia del desarrollo de metodologías y formas para administrar el proceso. Los orígenes de esta crisis se dan en los mismos inicios de la computación moderna, cuando lo más importante era el *hardware* y el *software* era relegado a un segundo plano. Sin embargo, los resultados de esta conferencia activan el cambio, haciendo del proceso de desarrollo de *software* una labor más científica, que ahora sobrepasa en formalización al desarrollo de *hardware.*

CONCLUSION

La postulación de un modelo abstracto para el proceso de diseño permitió analizar los aspectos relevantes en el diseño de *hardware* y de *software.*

La principal diferencia entre ambos radica en la naturaleza del objeto que va a ser creado. Mientras que en *hardware* los objetos tienen una naturaleza tangible, en *software* estos tienen una naturaleza intangible, la cual produce que muchas veces los diseños de *software* subestimen la cantidad de esfuerzo para desarrollar un programa o sistema.

Por esta misma intangibilidad del *software*, no son claras las fronteras entre diseño e implementación. Muchas veces no es posible decidir hasta dónde tiene que acabar la especificación de un programa sin que esa especificación se convierta en el programa mismo.

REFERENCIA BIBLIOGRAFICA

1. Alexander, Christofer. **Notes on the Synthesis of Form.** Harvard: Harvard University Press, 1964.

2. **Computer** 18(2). Número dedicado a *Hardware Description Languages.* Febrero 1985.

3. Dijkstra, E. W. *The Humble Programmer.* **CACM** 15, Octubre 1972.

4. Freman, P. y Wasseman, A. I. **Tutorial on Software Design Techniques.** 3 ed. New York. IEEE Computer Society, 1980.

5. Hoare, C. A. R. *An axiomatic basis for computer programming.* **CACM** 12, Octubre 1969.

6. Hoare, C. A. R. *Proof of correctness of data representations.* **Acta Informática** 1, 1972.

7. Parnas, D. L. *On a bussword: hierarchical.* IFIP, 1974.

8. Simon, Herbert. A. **The Sciences of the Artificial.** 2 ed. Massachusetts: The MIT Press, 1981.

9. Wirth, N. *Program development by stepwise refinement.* **CACM** 14, Abril 1971.