UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

TESIS PARA LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN COMPUTACIÓN

QUE PRESENTAN

AROCHE JUÁREZ, CARLOS FEDERICO
LEÓN ESTRADA, MOISÉS ALFREDO
VIDAL GARIBAY, FRANCISCO ARTURO

BAJO EL TÍTULO DE

LA INGENIERÍA DE SOFTWARE ORIENTADA HACIA LOS VIDEOJUEGOS

Cd. Universitaria, México, D.F. 2003

AGRADECIMIENTOS

La culminación de esta tesis y etapa de mi vida es muy importante por lo que quiero agradecer:

A mis abuelos Alicia Castañeda Alderete (q.e.p.d), Jesús Juárez Esparza (q.e.p.d) y Elivira Jiménez Pérez. Quienes siempre me brindaron su amor y apoyo para la realización de mis metas. Aunque algunos ya no estén en este mundo, siempre estarán en mi corazón.

A mis padres Adela Juárez Jiménez y Carlos Federico Aroche Castañeda. Quienes me dieron la vida, su amor, valores y principios, sin los que no podría haber llegado a este punto de mi vida.

A mi hermana Adela Aroche Juárez. Que siempre ha estado a mi lado compartiendo buenos y malos momentos y alentándome a conseguir mis metas.

A todos los maestros. Que durante mi vida escolar han dejado parte de ellos en mí, para formarme como persona y profesionista. Gracias.

De manera muy especial a Santiago Igor Valiente Gómez quien dirigió esta tesis.

A mis amigos Arturo Vidal Garibay y Moisés León Estrada. Con quienes realice esta tesis, compartiendo un sueño y logrando una meta que termina con este documento, para dar paso a nuevos proyectos individuales y compartidos.

A todos los demás amigos. Que no menciono nombres para no cometer omisión, pero saben que aunque no compartamos muchos momentos, siempre contarán conmigo. Mil gracias por su apoyo y amistad.

A todos mis Familiares consanguíneos y por afecto. Que me han apoyado en el transcurso de todos mis proyectos.

Muchas Gracias.

Carlos Federico Aroche Juárez

Estos agradecimientos son para las personas más importantes en mi vida:

A mi esposa Nohemi Quintana García. Esa hermosa mujer que siempre ha estado a mi lado impulsándome día a día, con quien compartiré mi vida, mis logros y mis fracasos, y además, por darme la dicha de ser papá próximamente.

A mi hijo. A quien esperamos con gran anhelo mi esposa y yo, siendo uno de los tantos motivos por el cual sigo adelante.

A mis padres Aurelio León Feria y Catalina Estrada Ramírez. Por ese gran apoyo incondicional en todos y cada uno de mis proyectos tanto en lo personal como en lo profesional, y desde luego, gracias por ese aliento diario el cual se ve recompensado con la culminación de mi carrera profesional.

A mi hermano Aurelio Rafael León Estrada. Quien ha crecido junto a mí apoyándome y ofreciéndome su amistad sincera a lo largo de mi vida y durante mi carrera.

A mis amigos Arturo Vidal, Carlos Aroche, Daniel Hernández, Cintia Quezada, Hugo Cedillo, Raúl Mota y Ana Ortiz. Por su amistad y sus grandes consejos que a través de mi carrera profesional me han servido para seguir adelante, espero que nuestra amistad perdure para siempre.

A mis maestros. Por su gran enseñanza, dedicación y esmero a través de todos y cada uno de los años de la carrera. Y de manera especial a Ing. Santiago Igor Valiente por dirigir la tesis con la que culmino mi carrera profesional.

A mi familia en general. Por todos esos consejos proporcionados a través de mi vida, los cuales me indican el camino para seguir adelante.

Y Finalmente a Dios. Porque me ha dado la dicha de salir siempre adelante en la vida.

Con todo mi cariño.

Moisés Alfredo León Estrada

Quiero agradecer:

A mis padres Arturo Vidal Gutiérrez y Veronica Garibay Mora. Por su amor, su esfuerzo y apoyo incondicional para conmigo, porque sin ellos nada de esto hubiera sido posible. Esto es sólo el fruto de una vida de lucha que construimos juntos. ¡Papás los amo mucho!

A mis hemanas Vero y Pao. Por estar a mi lado en todo momento. Espero que este trabajo les sirva de impulso para que puedan alcanzar todas sus metas y lleven a cabo todos sus sueños. ¡Gracias por su apoyo hermanas, las quiero mucho!

A Jesús y Diego. Por formar parte de mi vida en el momento adecuado. ¡Muchas gracias por ayudarme a entender y comprender junto con mi hermana el concepto de vivir experiencias nuevas!

A mis abuelos Alejandro Garibay González (q.e.p.d), Carmen Mora González y Francisca Gutierrez Alfaro (q.e.p.d). Por todas sus muestras de cariño y palabras de aliento.

A mis tíos María, Olivia y Ernesto. La primera por el ejemplo de valentía, tenacidad y temple para afrontar las cosas, la segunda por enseñarme de alguna manera a ver la vida como ella la ve y finalmente, el tercero por sus muestras de apoyo pero sobre todo por ser mi amigo, ¡los quiero!

A mis tíos y primos Sergio, Javier, Victoria, Sergis, Monse, Moni, Vancho, Caro, Pato y José María. Por ser parte de mi vida. Recuerden que tienen un lugar especial en mi corazón, ¡mi cariño y apoyo siempre!...

A Luis, Carlos, Armando, Sergio (q.e.p.d), Raúl (lobo), Israel y David. Que en su momento fueron parte de la lucha diaria para alcanzar este objetivo. Gracias por su amistad. ¡Sergio, este trabajo es como si fuera tuyo!

A Daniel, Cintia, Raúl, Hugo y Ana. Por su amistad y compañerismo incondicional. Mi agradecimiento por estar conmigo siempre. ¡Chavos, este trabajo también es de su autoría, muchas gracias!.

A Carlos y Moisés. Por su fidelidad como amigos. Sin ustedes este trabajo no hubiera sido posible. ¡Mi cariño, con ustedes hasta el final!...

A todas las personas que en algún momento formaron parte de mi vida. ¡Mi recuerdo y agradecimiento!

Finalmente, pero no menos importante, a Dios. Por permitirme estar en el lugar y momento en donde estoy y por tener a toda esta gente maravillosa a mi lado.

Sinceramente.

Arturo Vidal Garibay.

ÍNDICE

INTRODUCCIÓN

La tecnología, esa que permite el desarrollo del ser humano, es impredecible, en ocasiones cae dentro de la ciencia ficción y es incomprensible para la mayoría.

Hace algunos años, como bien narra esta tesis, la tecnología informática era un proceso que dependía fundamentalmente del Estado para actividades militares y de inteligencia; hoy ya es cotidiano conocer nuevo software con infinidad de aplicaciones, pero sobre todo, que es alcanzable para un porcentaje importante de la población mundial, sin embargo, existe un sector que desconoce las enormes potencialidades que se encuentran en la *bien* encaminada tecnología.

Desgraciadamente en México, como en muchas naciones en vías de desarrollo, aún persisten vicios, como la falta de una normatividad jurídica, apoyo hacendario y de ilícitos en la propiedad intelectual y de fabricación, que han bloqueado la implantación de empresas diseñadoras, productoras y fabricantes de software, principalmente de los llamados videojuegos; fenómeno que hay que detener dado que la industria extranjera y nacional visualizaría a México como una área de producción importante; hacerlo no sólo provocaría la creación de fuentes de empleo, sino también un gran ingreso de divisas que fortalecerían la economía nacional. Como también dicta este trabajo, la creación y fortalecimiento de esta industria atraería rendimientos por más de 2 mil millones de dólares anuales en los próximos 15 años, una oportunidad que estaría latente si se tomara conciencia del rubro.

Por otro lado, la industria del videojuego practicamente está en los inicios de su desarrollo en nuestro país, actualmente somos un mercado netamente consumidor, sin embargo, empiezan a darse los primeros proyectos de desarrollo de videojuegos con sello hecho en México. Debido a ésto la realización de este trabajo no fue fácil ya que existe poca información al respecto lo cual nos permite visualizar la importancia del videojuego, no sólo como un elemento de considerables potencialidades y desarrollo comercial, sino también por su trascendente aplicación en el contexto educacional y pedagógico; característica desconocida puesto que al juego en video se le ha *satanizado*, en muchas ocasiones con suma razón, por la gran cantidad de violencia que presentan, sin embargo, hay otros que cumplen con la función de integrar al individuo al ritmo social y otros que son eminentemente educativos, que de hecho auxilian de manera significativa a la persona que aún desconoce el proceso o uso de la tecnología computacional.

Asimismo, esta tesis nos proporciona los elementos estadísticos para la creación de videojuegos destinados a ciertos sectores de la población, además de los elementos teóricos profesionales para su desarrollo bajo la perspectiva de la ingeniería de software, herramientas trascendentes que nos permiten la construcción de manera ética y con la mínima probabilidad de error logrando con esto un producto de calidad que se reflejará en el gusto del usuario final.

Finalmente, creemos que la implementación de esta industria en México, y por supuesto su regulación, permitirá integrar al país en la ruta de las naciones tecnológicamente preparadas para enfrentar los retos, en todos los rubros, que se proyectan en este siglo XXI.

CAPÍTULO I

INGENIERÍA DE SOFTWARE

INTRODUCCIÓN

Durante las tres primeras décadas de la informática, el principal desafío fue el desarrollo del hardware de las computadoras de forma que redujera el tiempo y el costo en el procesamiento y almacenamiento de datos. A lo largo de ese proceso los avances en microelectrónica han dado como resultado una mayor capacidad de procesamiento y una disminución en sus dimensiones y, por consecuencia, una disminución en los costos de producción.

El potencial de las grandes computadoras en la Era de los años 80 está hoy disponible en una computadora personal. Las enormes capacidades de procesamiento y almacenamiento del moderno hardware representan una gran oportunidad de desarrollo.

Ahora bien, el software es el mecanismo que nos facilita el uso y la exploración de las capacidades del hardware, no obstante, el principal desafío es, y será, mejorar cualitativamente las soluciones basadas en las computadoras, mismas que se implementan con el software.

Cabe señalar que en sus inicios la construcción se realizaba sin ninguna planificación y se desenvolvía en un entorno fundamentalmente personalizado, es decir, el mismo individuo lo escribía, lo ejecutaba y, si cometía un error, lo depuraba. Irónicamente el crecimiento de la industria y su alta movilidad de recursos humanos trajo por consecuencia una *crisis del software* que apareció a raíz de los programas que tenían errores y tenían que ser corregidos, ya sea por el cambio en los requisitos del usuario o porque se debían adaptar a los nuevos dispositivos del hardware, o bien ambas razones. A esta actividad, principalmente correctiva, se le denominó *mantenimiento de software*, sin embargo, la misma naturaleza personalizada de muchos programas los hizo imposibles de mantener al no existir un enfoque único para su solución. De esa problemática surgió, como alternativa, la ingeniería de software que dispuso de métodos más completos para el desarrollo de programas, creó mejores herramientas para automatizarlos, utilizó superiores técnicas para garantizar su calidad e implantó una filosofía para la coordinación, control y gestión del software.

ANTECEDENTES HISTÓRICOS

El término *ingeniería de software* fue acuñado 1968 durante la conferencia de la Organización del Tratado Atlántico Norte (OTAN) en Garmisch, Alemania.

Uno de los objetivos de aquella reunión era estimular el interés en los contextos técnico y administrativo utilizados en el desarrollo y mantenimiento del software. En dicha conferencia se usó por primera vez, como ya mencionamos arriba, el término de ingeniería de software, es decir, para definir el conjunto de conocimientos y técnicas cuya aplicación permitía el uso racional de materiales y recursos naturales para invenciones, construcciones y otros procesos electrónicos para provecho del hombre; y se integró al concepto, la creación o construcción del software.

Desde entonces se ha integrado a nuevos campos de investigación y aplicaciones,

incluso es ya una profesión cuyo estudio superior es esencial para obtener una sólida base teórico-práctica en el rubro de la alta tecnología.

Ahora bien, la necesidad de un desarrollo sistemático y de mantenimiento del software, especialmente para asuntos bélicos, se patentizó durante la década de 1960; fue precisamente durante ese periodo cuando aparecieron las computadoras de la llamada tercera generación en donde se desarrollaron técnicas de programación como la denominada *multiprogramación* y el *tiempo compartido*; con ello las computadoras se hicieron más complejas y por consecuencia la demanda de software creció mucho más que la capacidad de fabricación y mantenimiento, no obstante, esas nuevas tecnologías fueron necesarias para el establecimiento de sistemas computacionales interactivos, de multiusuario, en línea y en tiempo real, apareciendo novedosas y más complejas aplicaciones para la computación, entre ellas las reservadas a la guerra y seguridad nacional.
Actualmente destacan importantes y trascendentes congresos relacionados a esta profesión como el convocado por la *Association of Cumputing Machinary* (ACM) que se llevó a cabo en Zurich, Alemania, en 1997 bajo el nombre de Special Interest Group on Software Engineering; otro fue el convocado por el Institute for Electrical & Electronic Engineers (IEEE) que se denominó *International Conference on Software Engineering* y que se realizó en Boston, EU, en 1998.

En otro contexto, la ingeniería de software tuvo sus inicios durante los años 60, etapa que se caracterizó por la adopción de los mnemónicos y énfasis en el uso del software bélico, pero fundamentalmente por el diseño bien definido y estructurado de éste a través del control de flujo de datos.

Durante sus inicios el software se desarrolló sin ninguna planificación; más bien era de carácter individual, y es que era fundamentalmente un anexo del hardware y por ello existían pocos métodos sistemáticos para la programación de computadoras.
Ahora bien, el software ha evolucionado significativamente, pasó de ser una resolución especializada de problemas y una herramienta de análisis de información, a ser una industria por sí misma; no obstante, la temprana cultura e historia de la programación ha creado un conjunto de problemas (que persisten hoy) y que se han convertido en un factor limitante para la evolución constante de los sistemas informáticos, pero por otro lado, es el único método de afrontar, de forma racional, los cada día más complejos proyectos de software que requiere la sociedad sin correr el riesgo de caer en los errores de un proceso que no tuvo un desarrollo adecuado.

INGENIERIA DE SOFTWARE: DEFINICIÓN

La ingeniería de software tuvo una definición en la conferencia de la OTAN de 1968, la cual hemos mencionado, pero una de las principales fue dada por Frizt Bauer en 1968, en la primera conferencia dedicada al tema: *« La ingeniería del software es el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales.»* **(1)**

Aunque se han propuesto más definiciones todas refuerzan la importancia de una

(1) Pressman Roger S. *Ingeniería de Software, Un enfoque práctico*. Ed. Mc.GrawHill, Estados Unidos. 1997. Prólogo.

disciplina para el desarrollo del software. En general, este tipo de ingeniería abarca tres elementos fundamentales que facilitan el proceso y suministran las bases para construirlo con alta calidad:

1.- Métodos.- Indican cómo construirlo técnicamente. Cubren un amplio espectro de tareas que incluyen: Planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructuras de datos, arquitectura de programas y algoritmos, codificación, prueba y mantenimiento. Los métodos se desenvuelven en un conjunto de criterios que garantizan la calidad del producto final.

2.- Herramientas.- Suministran un soporte automático o semiautomático para los métodos. Cuando se integran de forma que la información creada puede ser usada por una o varias herramientas, se establece un sistema para el soporte del desarrollo de programas llamado ingeniería del software asistida por computadora, y que en inglés se denominan CASE **(2)** y combina software, hardware y bases de datos, es decir, una estructura que contiene la información sobre el análisis, diseño, desarrollo y mantenimiento para crear un entorno análogo al diseño-ingeniería asistido por computadora para el hardware.

3.- Procedimientos.- Son la unión de los métodos y herramientas que facilitan un desarrollo racional y oportuno del software de computadora. Definen la secuencia en la que se aplican los métodos, las entregas y los controles que ayudan a asegurar la calidad y a coordinar las directrices para evaluar el proceso.

Tomando en consideración lo anterior, podemos definir que: *La ingeniería de software es una disciplina que integra métodos, herramientas y procedimientos mediante los cuales se puede desarrollar un software eficiente y cualitativamente óptimo.*

DESARROLLO Y EVOLUCIÓN

Podemos indicar que la ingeniería de software se divide en cuatro importantes periodos o etapas. Durante la primera Era el software era diseñado con carencias que provocaron la individualización en la medida que requería cada aplicación y, por extensión, tuvo una distribución comercial relativamente pequeña y en muchos casos nula.

Durante los años iniciales de desarrollo de las computadoras, el hardware sufrió continuos cambios mientras que el software era contemplado como un anexo, además existían pocos métodos sistemáticos para la programación de computadoras, incluso, el software se construía sin ninguna planificación. Durante ese periodo se utilizaba, en la mayoría de los sistemas, una orientación por lotes, sin embargo, había excepciones como sistemas interactivos y de tiempo real, pero la mayor parte del hardware se dedicaba a la ejecución de un único programa que, a su vez, ejercía una aplicación específica.

La segunda Era se extiende desde la mitad de la década de los años 60 hasta finales de los 70. La multiprogramación y los sistemas multiusuario introdujeron nuevos concep-

(2) Pressman Roger S., *Brereton P. Software Engineering Enviroment*. Ed. Mc. GrawHill, Estados Unidos, 1998. Pág. 18.

tos de interacción hombre-máquina. Estas técnicas abrieron nuevas perspectivas en aplicaciones y hasta en la sofisticación del hardware y del software.

Los sistemas de tiempo real podían recoger, analizar y transformar datos de múltiples fuentes, controlando los procesos y produciendo respuesta en milisegundos, con ello, los avances en los dispositivos de almacenamiento condujeron la primera generación de sistemas de gestión de bases de datos.

Este periodo también se caracterizó por la aparición de empresas dedicadas a la construcción, diseño y comercialización de software no sólo para las grandes computadoras, sino también para las pequeñas. Paralelamente y conforme crecía el número de sistemas informáticos se extendieron también las bibliotecas de software. No obstante del desarrollo del software en esta segunda Era, las empresas desarrollaron proyectos en los que se producían programas de miles de líneas de código fuente, sin embargo, al no utilizarse una metodología en el desarrollo, éstos contenían numerosos errores e inconsistencias, lo que obligaba a un mantenimiento.

Estas continuas modificaciones aumentaron los problemas y por consecuencia se incrementó el costo, a tal grado que frecuentemente era más rápido construir uno nuevo, pero éstos no estaban exentos de aquellos, convirtiéndose en un fenómeno cíclico y dando origen a la llamada *crisis del software*.

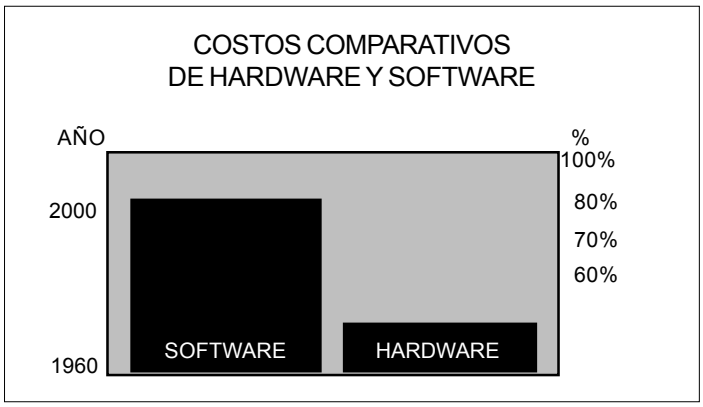
La tercera etapa inició a mediados de los 70, cuando el procesamiento distribuido, denominado generalmente como *redes* (múltiples computadoras, cada una ejecutando funciones y comunicándose con alguna otra) incrementó notablemente la complejidad de los sistemas informáticos. Las redes de área local y global, las comunicaciones digitales y la creciente demanda de acceso a datos, supusieron una fuerte presión sobre los fabricantes de software.

Este periodo también se caracteriza por el surgimiento y el amplio uso de los microprocesadores que pasaron a ser parte integral de un extenso espectro de productos inteligentes y de las computadoras personales y que han sido el catalizador del constante crecimiento de muchas compañías que actualmente venden millones de copias de software.

Actualmente nos encontramos en la cuarta Era que está en pleno auge. Las tecnologías orientadas a objetos están desplazando rápidamente los enfoques de desarrollo convencionales en muchas áreas de aplicación. Las técnicas de hoy están revolucionando la forma en que algunos segmentos de la comunidad informática desarrollan los programas. Hoy los sistemas expertos y el software de inteligencia artificial (IA) son utilizados para aplicaciones prácticas y para un amplio rango de problemas del mundo real, incluso, el software de redes neuronales artificiales abre excitantes posibilidades para el reconocimiento de formas y habilidades de procesamiento de información al estilo prácticamente humano.

En términos generales, la ingeniería de software es una actividad muy reciente, de escasos 50 años, en los cuales ha crecido y evolucionado considerablemente; se pasó de los procesadores de bulbos a los dispositivos microelectrónicos que son capaces de procesar millones de instrucciones por segundo.

En otro contexto y de acuerdo a los antecedentes históricos de esta industria, se debe reconocer que el software en sus inicios estaba directamente ligado al desarrollo del



Gráfica 1

hardware y éste era el elemento principal que determinaba el costo, hoy se ha invertido, es decir, el software es el factor principal del precio, como podemos apreciar en la gráfica 1.

Actualmente el hardware de las computadoras personales se ha convertido en un producto estándar mientras que el software marca la diferencia entre una computadora y otra, de hecho, las tendencias indican que en el campo industrial se ha incrementado más el gasto en un software que la misma computadora que lo ejecuta.

CRISIS DEL SOFTWARE

Hemos mencionado que durante los primeros procesos evolutivos del software se desarrolló un conflicto al que se le denominó *crisis del software*, producto del crecimiento de la industria y su alta movilidad de recursos humanos.

También mencionamos que los primeros productos de software, al ser prácticamente individualizados, tenían errores y por lo tanto era necesario corregirlos, además, los requisitos del usuario cambiaban y por consiguiente la funcionalidad del software, en ocasiones se tenían que adaptar nuevos dispositivos de hardware.

Todo este fenómeno provocó que la industria entrara en una crisis que llevó a adoptar nuevos procedimientos, entre ellos el de mantenimiento, sin embargo, la misma naturaleza personalizada de muchos programas los hacía imposibles de mantener y, al no existir un enfoque único para la solución de esta problemática apareció la ingeniería de software que combinó métodos más completos para el desarrollo de programas, utilizó mejores herramientas para automatizarlos, usó técnicas superiores para garantizar su calidad e implantó una filosofía para la coordinación, control y gestión del software y con ello el conflicto referido tuvo, en parte, solución.

Como vemos, el software es actualmente, dentro de cualquier sistema basado en el uso de computadoras, el componente cuyo desarrollo presenta mayores problemas puesto que es el más difícil de planificar, el que tiene mayores probabilidades de fracaso y el que tiene menos posibilidades de que se cumplan las estimaciones de costos iniciales. Asimismo, su constante demanda y complejidad trae por consecuencia que la problemática se incremente, pero paralelamente, la ingeniería de software se encuentra en vías de soluciones expeditas y cualitativamente eficientes dado que el desarrollo del software es una actividad muy joven.

DEFINICIÓN DE SOFTWARE

En la década de los 70 menos del 1% de la población mundial podía definir lo que significaba el software de computadora, actualmente ya forma parte del entorno educativo, sin embargo, aún persisten definiciones equivocadas y no ha sido asimilado correctamente.

El software, en términos generales, son instrucciones de computadora que cuando se ejecutan proporcionan una función y un comportamiento deseado; también son estructuras de datos que facilitan a los programas manipular adecuadamente la información; y son una serie de documentos que describen la operación y el uso de programas, es decir, el software incluye no sólo los programas de la computadora, sino también las estructuras de datos que manejan éstos y toda la documentación que debe acompañar al proceso de desarrollo, mantenimiento y uso.

CARACTERÍSTICAS DEL SOFTWARE

Para poder comprender lo que es el software es importante examinar sus características que lo diferencian de otras cosas que el hombre puede construir; en el proceso de construcción existen cuatro elementos fundamentales: El análisis, el diseño, la construcción y la prueba, que se traducen finalmente en un objeto físico, no obstante, es un elemento del sistema que es lógico y por lo tanto tiene características muy distintas a las del hardware. A pesar de que existen similitudes en la mayoría de los procesos de desarrollo, en la fase de producción se encuentran diferencias fundamentales; por ejemplo, en la del hardware pueden aparecer problemas que afecten la calidad y que no existen o que son relativamente fáciles de resolver o evitar en el software. Asimismo, en la producción a gran escala de hardware el costo del producto final depende exclusivamente del precio de los materiales empleados y del propio proceso, entre otros aspectos; mientras que en el software el desarrollo es una más de las labores de ésta y la producción (a gran o pequeña escala) no influye en el impacto que tiene en el costo ya que es, como lo mencionamos, un producto lógico, ver gráfica 2.

Cifras en dólares.

	Producción o Desarrollo	Costo U. en 100 unidades	Costo U. en 100,000 unidades
Hardware	50	60	50.01
Software	2000	30	0.03

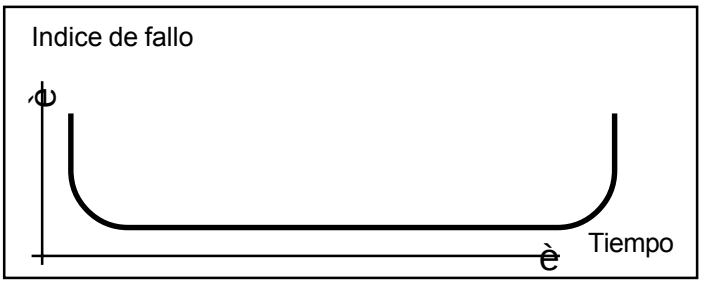
Gráfica 2

Paralelamente la reproducción del software no presenta problemas técnicos, además de que no se requiere de un control de calidad individualizado.

Cabe señalar que el impacto que tiene la ingeniería en el desarrollo del software y del hardware es amplio, los costos del primero se encuentran incluidos en el desarrollo y no en la producción, y es precisamente en ese contexto donde hay que incidir para reducir el precio final del producto.

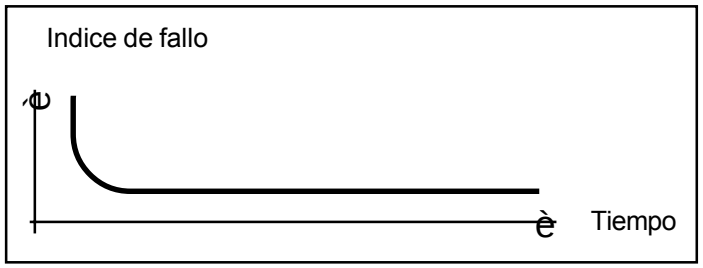
Por otro lado, y en función a las fallas que pueden aparecer en ambos elementos, podemos comparar las curvas de índices del hardware y del software en función del tiempo. En el primer caso se tiene la llamada *curva de bañera*, que indica que el producto presenta relativamente muchas fallas al principio de su vida, debidas fundamentalmente a defectos de diseño o a la baja calidad inicial en la fase de producción.

Una vez corregidos, los costos caen hasta un nivel estacionario y se mantienen así durante cierto periodo de tiempo, posteriormente la tasa vuelve a incrementarse debido al deterioro en los componentes del hardware que van siendo afectados por la suciedad, vibraciones y por otros factores externos, a tal grado que debemos y podemos sustituir los elementos defectuosos o todo el sistema y el índice vuelve a situarse en el nivel estacionario. (ver gráfica 3)



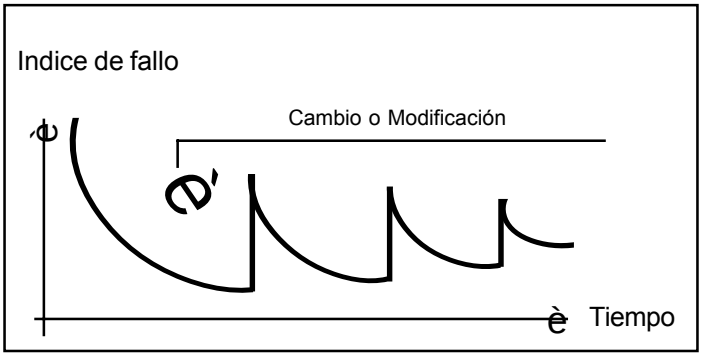
Gráfica 3

En el caso del software, este no es susceptible a factores externos que hacen que, como el hardware, se estropee, sin embargo, inicialmente la tasa de fallas es alta debido a la presencia de errores no detectados durante el desarrollo y a los que se les denomina *errores latentes*, pero que una vez corregidos el porcentaje disminuye hasta alcanzar el nivel estacionario e, idealmente, mantenerse ahí indefinidamente. (ver gráfica 4)



Gráfica 4

Pero esto no es más que una simplificación del modelo real de fallas de un producto de software; durante su vida sufre cambios que se deben al mantenimiento, ya sea correctivo, preventivo o a cambios en los requisitos iniciales del producto, conforme se hacen cambios es probable que surjan nuevos problemas y con ello se producen picos en la curva de fallas, pero esos errores pueden corregirse, no obstante, el efecto de éstos hace que el producto se aleje o se distorsione de las especificaciones originales, además, con mucha frecuencia se requiere un nuevo cambio antes de haber conseguido corregir todos los errores producidos por la permuta anterior, por esas razones el nivel estacionario que se consigue después del cambio es superior al que había antes de efectuarlo, degradándose poco a poco el funcionamiento del sistema. Podemos decir entonces que el software no se estropea pero se deteriora (ver gráfica 5). No obstante,



Gráfica 5

cuando el componente se daña no podemos sustituirlo por otro, como es el caso del hardware, ya que no existen piezas de repuesto. Cada falla del software significa un error en el diseño o en el proceso mediante el cual se transformó el diseño en código-máquina ejecutable y la solución no es sustituirlo sino crear otro con todo el proceso de desarrollo que implica, por lo tanto, el mantenimiento del software tiene una complejidad mucho mayor que el del hardware.

Ahora bien, es importante indicar que el software se construye, no se ensambla con componentes existentes, mientras que el hardware se realiza, en gran medida, a base de componentes digitales que se encuentran en el mercado, cuyas características se comprueban en un catálogo y que han sido exhaustivamente probados por el fabricante y los usuarios. Estos componentes cumplen especificaciones claras y tienen unas interfaces definidas; en el software es distinto, dado que no existen catálogos de componentes, y aunque determinados productos como sistemas operativos, procesadores de palabras y bases de datos se venden en grandes ediciones, la mayoría se fabrica a medida, siendo muy baja la reutilización como parte inicial del desarrollo.

Se puede comprar un software ya desarrollado, pero sólo como unidades completas, no como componentes que pueden ser reensamblados para construir nuevos programas, esto hace que el costo de ingeniería, sobre el producto final, sea muy elevado al dividirse entre un número muy pequeño de unidades producidas.

Con respecto a la reutilización del software, han sido muchos los intentos para conseguir aumentar el nivel de reuso pero con poco éxito. Uno de los ejemplos más típicos, que ha venido aplicándose desde hace tiempo, son las bibliotecas o librerías. Durante los años 70 se empezaron a desarrollar bibliotecas de subrutinas científicas, reutilizables en una amplia gama de aplicaciones científicas y de ingeniería; la mayor parte de los lenguajes modernos incluyen bibliotecas de este tipo, así como otras para facilitar la entrada-salida y más recientemente los entornos de ventanas, sin embargo, esta aproximación no puede aplicarse fácilmente a otros problemas también de uso muy frecuente como puede ser la búsqueda de un elemento en una estructura de datos, debido a la gran variación que existe en cuanto a la organización interna de éstas y en la composición de los datos que contienen.

Existen algoritmos para resolver estos problemas, pero hay que programarlos una y otra vez, adaptándolos a cada situación particular.

Un nuevo intento de conseguir la reutilización se produjo con el uso de técnicas de programación estructurada y modular, sin embargo, se dedica por lo general poco esfuerzo al diseño de módulos lo suficientemente generales para ser reutilizables y, en todo caso, no se documentan ni se difunden ampliamente para extender su uso, con lo que la tendencia habitual es diseñar y programar módulos muy semejantes una y otra vez. La programación estructurada permite diseñar programas con una estructura más clara y que, por lo tanto, sean más fáciles de entender. Esta estructura interna permite el reuso de módulos dentro de los programas, incluso dentro del proyecto que se está desarrollando, pero la reutilización de código en proyectos diferentes es muy baja.

La última tendencia para conseguir la reutilización es el uso de técnicas orientadas a objetos, que permiten la programación por especialización. Los objetos, que encapsulan tanto datos como los procedimientos que los manejan haciendo los detalles de

implementación invisibles e irrelevantes a quien los usa, disponen de interfaces claras, los errores cometidos en su desarrollo pueden ser depurados sin que esto afecte la corrección de otras partes del código y pueden ser heredados y reescritos parcialmente haciendo posible su reutilización aún en situaciones no contempladas en el diseño inicial.

PROBLEMAS DEL SOFTWARE

Hemos hablado de una *crisis del software*, sin embargo, por crisis entendemos normalmente un estado pasajero de inestabilidad, que tiene como resultado un cambio de sistema o una vuelta al estado inicial, en caso de que se tomen las medidas para superarla. Tomando en consideración esto, el software, más que padecer una crisis podemos decir que se encuentra en una fase casi irreversible.

Los problemas que surgieron cuando se empezó a desarrollar el software de una cierta complejidad siguen existiendo actualmente sin que se haya avanzado mucho en los intentos para solucionarlos, estos problemas son causados por las propias características del software y por los errores cometidos por quienes intervienen en su producción.

Entre los errores más comunes podemos destacar el que se presenta en la imprecisión de la planificación y la estimación de costos. A la hora de abordar un proyecto con cierta complejidad, en el ámbito que sea, es frecuente que surjan imprevistos que no estaban contemplados en la planificación inicial, y como consecuencia de éstos se producirá una desviación en los costos del proyecto, sin embargo, en el desarrollo de software lo más frecuente es que la planificación sea prácticamente inexistente, y que nunca se revise durante el desarrollo del proyecto. Sin una planificación detallada es totalmente imposible hacer una estimación de costos que tenga alguna posibilidad de cumplirse, y tampoco se pueden identificar las tareas conflictivas que pueden desviarnos de los costos previstos.

Entre las causas del problema de planificación y estimación podemos nombrar:

1.- No se recogen datos sobre el desarrollo de proyectos anteriores con lo que no se acumula experiencia que pueda ser utilizada en la planificación de nuevos proyectos.

2.- Los gestores de los proyectos no están especializados en la producción de software. Tradicionalmente, los responsables de su desarrollo han sido ejecutivos de nivel medio y superior sin conocimientos de informática, siguiendo el principio de que *un buen gestor puede gestionar cualquier proyecto*. Esto es cierto, pero no cabe duda que también es necesario saber las características específicas del software, aprender las técnicas que se aplican en su desarrollo y conocer una tecnología que evoluciona continuamente.

LA PRODUCTIVIDAD ES BAJA

Los proyectos de software tienen, por lo general, una duración mucho mayor a la esperada, por consecuencia los costos se disparan y la productividad y los beneficios disminuyen. Uno de los factores que influyen en este fenómeno es la falta de metas claras y reales al comenzar el proyecto, además, la mayoría del software se desarrolla a partir de unas especificaciones ambiguas o incorrectas y apenas existe comunicación con el cliente, debido a ello son muy frecuentes las modificaciones de las especificaciones sobre la mar-

cha y hasta cambios de última hora, incluso no se realiza un estudio detallado del impacto de estos cambios y la complejidad interna de las aplicaciones crece hasta que se hacen virtualmente imposibles de mantener y cada nueva modificación, por pequeña que sea, es más costosa y puede provocar fallas en todo el sistema.

Ahora bien, debido a la falta de documentación sobre cómo se ha desarrollado el producto o a que las sucesivas modificaciones han desvirtuado totalmente el diseño inicial, el mantenimiento del software puede llegar a ser una tarea imposible de realizar, pudiendo incluso llevar más tiempo el realizar una modificación sobre el programa ya escrito que analizarlo y desarrollarlo de nuevo.

DEFICIENTE CALIDAD

Como consecuencia de que las especificaciones son ambiguas o incorrectas, y de que no se realizan pruebas exhaustivas, el software contiene numerosos errores cuando se entrega al cliente. Éstos producen un fuerte incremento de costos durante el mantenimiento del producto cuando ya se esperaba que estuviese acabado.

Sólo recientemente se ha tomado en cuenta la importancia de la prueba sistemática y completa y han empezado a surgir conceptos como la fiabilidad y la garantía de calidad.

CLIENTE INSATISFECHO

Debido al poco tiempo e interés que se dedican al análisis de requisitos y a la especificación del proyecto, a la falta de comunicación durante el desarrollo y a la existencia de numerosos errores en el producto que se entrega, los clientes suelen quedar poco satisfechos de los resultados y por consecuencia se tiene que rediseñar el producto, o en el caso más drástico, se produce un cambio de proveedor.

MITOS DEL SOFTWARE

Muchas de las causas de la *crisis del software* se pueden encontrar en una mitología que surge durante los primeros años de su desarrollo.

Hoy, la mayoría de los profesionales consideran a los mitos por lo que son, ya que estos han causado serios problemas tanto a los gestores como a los técnicos.

La gente con responsabilidad sobre el software, como los gestores en la mayoría de las disciplinas, están normalmente bajo la presión de cumplir con los presupuestos, hacer que no se retrase el proyecto y mejorar la calidad; frecuentemente se basa en un mito para disminuir la presión, aunque esto sólo sea temporalmente.

Los mitos más comunes son acerca de los manuales de procedimiento para el desarrollo, los cuales no son útiles si los trabajadores desconocen su uso pero sobre todo su existencia; el gasto excesivo de recursos materiales y humanos con el objetivo de contar con el mejor equipo de hardware (sin la herramienta adecuada) y, el aumento de desarrolladores que a fin de cuentas no beneficia el avance y calidad de cualquier proyecto, por el contrario, lo más común es que genere un retraso.

En muchos casos el cliente cree en los mitos que existen sobre el software, ya que los gestores y desarrolladores hacen muy poco para corregir la mala información. Los mitos conducen a que el cliente tenga una falsa expectativa y finalmente quede insatisfecho con el producto, debido a que se realiza una mala definición del problema a solucio-

nar, puesto que en muchas ocasiones se cree que una sola declaración es suficiente para comenzar a escribir los programas y, por el contrario, las características y los objetivos sólo se pueden determinar después de una exhaustiva comunicación con el cliente. Se puede pensar que los cambios en el producto son fáciles de realizar, ya que el software es versátil, sin embargo, dependiendo de la etapa en la que se encuentre el proyecto los cambios requeridos necesariamente incrementaran los recursos reflejándose en un tiempo y un costo más elevado.

En el contexto del área de los desarrolladores se tienen ciertos mitos que fomentan la improductividad y que trae por consecuencia la falta de calidad del software.

Los desarrolladores piensan que una vez que el programa es escrito y funciona, el trabajo termina, sin embargo, de acuerdo con datos industriales, se indica que el 60% del esfuerzo y dedicación a un programa se realiza después de su entrega.

La forma en que podemos garantizar la calidad del producto es llevando a cabo una adecuada planeación, de modo que tenga como resultado final el programa funcionando y una documentación que compruebe un adecuado desarrollo para que finalmente proporcione la información necesaria para su mantenimiento.

APLICACIONES DEL SOFTWARE

El software puede aplicarse a numerosas situaciones del mundo real. En primer lugar, a todos aquellos problemas para los que se haya establecido un conjunto específico de acciones que lleven a su resolución, para conseguirlo, se emplea el uso de lenguajes de programación procedurales para implementar algoritmos; en segundo lugar, puede aplicarse a situaciones en las que el problema se describe formalmente, por lo general en forma recursiva. En estos casos, no se necesita describir el método de resolución, es decir, cómo se resuelve el problema, basta con describir la problemática en sí, indicando cuál es la solución deseada, para esto, se emplean lenguajes declarativos.

También puede aplicarse en problemas que resolvemos utilizando multitud de reglas heurísticas posiblemente contradictorias e incluso, a problemas de los cuales no tenemos una idea clara de cómo se resuelven, para lo cual se utilizan sistemas expertos. En cualquier caso, es difícil establecer categorías genéricas significativas para las aplicaciones del software. Conforme aumenta la complejidad del mismo se hace más arduo establecer compartimentos separados.

La siguiente clasificación ha venido aceptándose tradicionalmente:

1.- Software de sistemas.- Es un conjunto de programas que han sido escritos para servir a otros, está formado por todos aquellos cuya finalidad es servir al desarrollo o al funcionamiento de otros programas y estos son muy variados: Editores, compiladores, sistemas operativos, entornos gráficos, programas de telecomunicaciones, etc., se caracterizan por ser utilizados concurrentemente por numerosos usuarios y por tratarse de programas de amplia difusión no estando diseñados normalmente a medida, esto permite un mayor esfuerzo en su diseño y optimización, pero también les obliga a ser muy confiables, cumpliendo estrictamente las especificaciones para las que fueron creados.

2.- Software de tiempo real.- Esta formado por todos aquellos programas que miden,

analizan y controlan los sucesos del mundo real a medida que ocurren debiendo reaccionar de forma correcta a los estímulos de entrada en un tiempo máximo prefijado. Deben, por lo tanto, cumplir requisitos temporales muy estrictos y, dado que los procesos que controlan pueden ser potencialmente peligrosos, tienen que ser fiables y tolerantes a fallas. Por otro lado, no suelen ser muy complejos y precisan de poca interacción con el usuario.

3.- Software de gestión.- El procesamiento de información de gestión constituye, casi desde los inicios de la informática, la mayor de las áreas de aplicación de las computadoras en el uso del software. Estos programas utilizan grandes cantidades de información almacenadas en bases de datos con el objetivo de facilitar las transacciones comerciales o la toma de decisiones, además de las tareas convencionales de procesamiento de datos en las que el tiempo no es crítico y los errores pueden ser corregidos *a posteriori*, incluyen programas interactivos que sirven de soporte a transacciones comerciales. Las aplicaciones en esta área reestructuran los datos existentes en orden facilitando las operaciones comerciales o gestionando la toma de decisiones.

4.- Software científico y de ingeniería.- Está caracterizado por los algoritmos de manejo de números. Se encarga de realizar complejos cálculos sobre datos numéricos de todo tipo, en este caso la corrección y exactitud de las operaciones que realizan es uno de los requisitos básicos que deben cumplir. El campo del software científico y de ingeniería se ha visto ampliado últimamente con el desarrollo de los sistemas de diseño, ingeniería y fabricación asistida por computadoras (CAD, CAE y CAM), los simuladores gráficos y otras aplicaciones interactivas que lo acercan más al software de tiempo real e incluso al de sistemas.

5.- Software de computadoras personales.- El uso de computadoras personales y de uso doméstico se ha generalizado a lo largo de las últimas décadas. Aplicaciones típicas son los procesadores de textos, las hojas de cálculo, bases de datos, aplicaciones gráficas, juegos, etc. Son productos de amplia difusión orientados a usuarios no profesionales, por lo que entre sus requisitos se encuentran la facilidad de uso y el bajo costo, de hecho el software de las computadoras personales continúa representando uno de los diseños más innovadores en este campo.

6.- Software empotrado.- Este tipo de software reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. Se instala en otros productos industriales, por ejemplo en la electrónica de consumo, dotando a éstos de un grado de inteligencia cada vez mayor. Se aplica a todo tipo de productos, desde un video doméstico hasta un misil atómico. Realiza funciones muy diversas que pueden ir desde complicados cálculos en tiempo real a sencillas interacciones con el usuario facilitando el manejo del aparato que los incorpora. Comparten características con el software de sistemas, el de tiempo real, el de ingeniería y científico y el de computadoras personales.

7.- Software de inteligencia artificial.- El software basado en lenguajes procedurales

es útil para realizar de forma rápida y fiable operaciones que para el ser humano son tediosas e, incluso, inabordables, sin embargo, es aplicable a problemas que requieren de funciones intelectuales más elevadas, por triviales que nos puedan parecer.

El software de inteligencia artificial trata de dar respuesta a estas deficiencias basándose en el uso de lenguajes declarativos, sistemas expertos y redes neuronales. Dentro de la inteligencia artificial también se interactúa con la red neuronal la cual simula a la estructura del cerebro y a la larga puede llevar a una clase de software que pueda reconocer patrones complejos y aprender de experiencias pasadas.

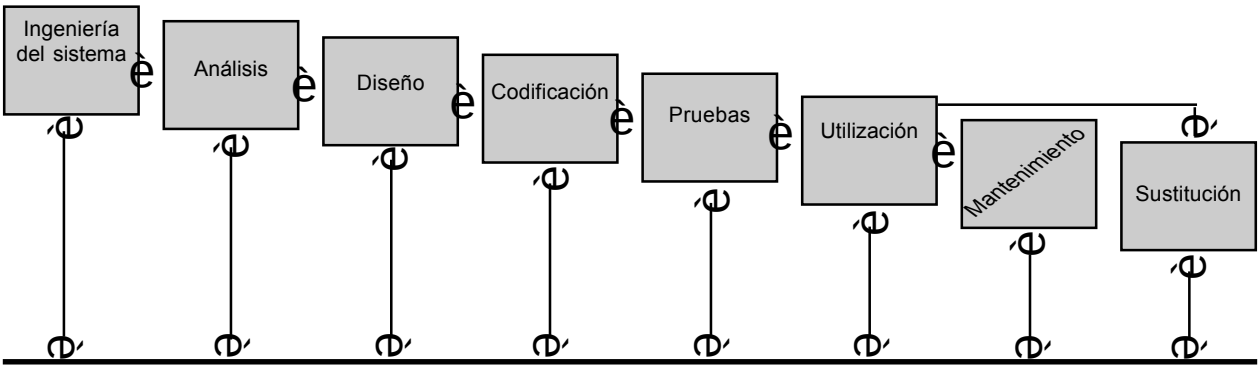
Como sabemos, el software permite aplicaciones muy diversas, pero en todas ellas podemos encontrar algo en común dado que el objetivo es que desempeñe una determinada función y, además, debe hacerlo cumpliendo una serie de requisitos: Corrección, fiabilidad, respuesta en un tiempo determinado, facilidad de uso, bajo costo, etc.

CICLO DE VIDA

Por *ciclo de vida* se entiende la sucesión de etapas por las que pasa el software desde que un nuevo proyecto es concebido hasta que se deja de usar. Cada una de estas etapas lleva asociada una serie de tareas que deben realizarse, y una cadena de documentos que serán la salida de cada una de las fases y servirán de entrada en la siguiente.

Existen diversos modelos de ciclo de vida, y cada uno de ellos va asociado a un paradigma de la ingeniería del software, es decir, a una serie de métodos, herramientas y procedimientos que debemos usar a lo largo de un proyecto. La elección de un paradigma u otro se realiza de acuerdo con la naturaleza del proyecto y de la aplicación, los métodos a usar y los controles y entregas requeridos, por ejemplo, el paradigma denominado en *cascada* o *clásico* que es el más antiguo de los empleados en la ingeniería de software y se desarrolló a partir del ciclo convencional de la ingeniería para dar solución a los problemas más comunes que aparecían al desarrollar sistemas de software complejos. No hay que olvidar que la ingeniería de software surgió como copia de otras ingenierías, especialmente de la de hardware. En la siguiente gráfica (6) se ilustra el paradigma del ciclo de vida *clásico* o también llamado de *cascada*.

Este exige un enfoque sistemático y secuencial del desarrollo de software que comienza en el nivel de la ingeniería de sistemas y avanza a través de fases secuenciales sucesivas.



Gráfica 6

Estas fases son:

1.- Ingeniería y análisis del sistema.- El software es parte de un sistema mayor por lo que siempre va a interrelacionarse con otros elementos, ya sea hardware, máquinas o personas, por esto, el primer paso del ciclo de vida de un proyecto consiste en un análisis de las características y el comportamiento del sistema del cual va a formar parte. En el caso de que se quiera implantar un sistema nuevo se deberán analizar cuáles son los requisitos y la funcionalidad del sistema. Por el contrario, para uno ya existente se deberá analizar el funcionamiento operativo del lugar en que se va a implantar para posteriormente adecuarlo a las funciones que se van automatizar. La ingeniería del sistema comprende, por lo tanto, los requisitos globales del sistema, así como una cierta cantidad de análisis y de diseño a escala superior, es decir, sin entrar en mucho detalle.

2.- Análisis de requisitos del software.- El proceso de recopilación de los requisitos se centra y especifica especialmente para el software. Para comprender la naturaleza de los programas que hay que construir, el analista debe entender el ámbito de la información del software, así como la función, el rendimiento y las interfaces necesarias. Los requisitos tanto del sistema como del software deben documentarse y revisarse con el cliente.

3.- Diseño.- Se aplica a cuatro características distintas del software: La estructura de los datos, la arquitectura de las aplicaciones, la estructura interna de los programas y las interfaces. El diseño es el proceso que traduce los requisitos en una representación del software de forma que pueda conocerse la arquitectura, funcionalidad e incluso la calidad del mismo antes de comenzar la codificación. Al igual que el análisis, el diseño debe documentarse y formar parte de la configuración del sistema.

4.- Codificación.- La codificación consiste en la traducción del diseño a un formato que sea legible para la máquina. Si el diseño es lo suficientemente detallado la codificación es relativamente sencilla y puede hacerse de forma automática usando generadores de código. Hasta este punto podemos observar que estas fases del ciclo de vida consisten básicamente en una traducción: En el análisis del sistema, los requisitos, la función y la estructura de éste se traducen a un documento; el análisis del sistema está formado en parte por diagramas y en parte por descripciones en lenguaje natural. En el análisis de requisitos se profundiza en el estudio del componente software del sistema y esto se traduce a un documento formado por diagramas y descripciones en lenguaje natural.

En el diseño, los requisitos del software se aplican a una serie de diagramas que representan la estructura del sistema, de sus datos, de sus programas y de sus interfaces. Por último, en la codificación se integran estos diagramas de diseño a un lenguaje fuente que luego se compila para obtener un programa ejecutable.

5.- Prueba.- Una vez que ya tenemos el programa ejecutable comienza la fase de pruebas. La prueba se centra en la lógica interna del software y su objetivo es comprobar que no se hayan producido errores en alguna de las fases de traducción anteriores, especialmente en la codificación, para ello deben probarse todas las sentencias, no sólo en

condiciones normales y, además, como parte de esta etapa, se deben probar todos los módulos que conforman el sistema.

6.- Utilización.- Una vez superada la fase de pruebas, el software se entrega al cliente y comienza su vida útil. La fase de utilización se sustenta con las posteriores ejecuciones, el mantenimiento y la sustitución y dura hasta que el software, ya reemplazado por otro, deje de ser útil.

7.- Mantenimiento.- El software sufrirá cambios a lo largo de su vida útil, estos pueden ser debidos a tres causas: Que durante la utilización el cliente detecte errores; que se produzcan cambios en alguno de los componentes del sistema informático, por ejemplo en la máquina, en el sistema operativo o en los periféricos y, que el cliente requiera modificaciones funcionales (normalmente ampliaciones) no contempladas en el proyecto. En cualquier caso el mantenimiento supone volver atrás en el ciclo de vida, a las etapas de codificación, diseño o análisis dependiendo de la magnitud del cambio. El modelo en *cascada*, a pesar de ser lineal, contiene flujos que permiten retomar alguna de las etapas anteriores. Así, desde el mantenimiento se vuelve al análisis, al diseño o a la codificación y también desde cualquier fase se puede regresar a la anterior si se detectan fallas. Estas vueltas no son controladas, ni quedan explícitas en el modelo y este es uno de los problemas que presenta este paradigma.

8.- Sustitución.- La vida del software no es ilimitada y cualquier aplicación, por buena que sea acaba por ser sustituida por otra más amplia, más rápida o más fácil de usar. La sustitución de un software que está funcionando por otro que acaba de ser desarrollado es una tarea que hay que planificar cuidadosamente y de forma organizada. Es conveniente realizarla por fases si es posible, no sustituyendo todas las aplicaciones de golpe, puesto que ésta conlleva normalmente un aumento de trabajo para los usuarios que tienen que acostumbrarse a las nuevas aplicaciones, y también para los desarrolladores, que tienen que corregir los errores que aparecen. Es necesario hacer un traspaso de la información que maneja el sistema viejo a la estructura y el formato requeridos por el nuevo, además es conveniente mantener los dos sistemas funcionando paralelamente durante algún tiempo para comprobar que el sistema nuevo funcione correctamente y para asegurar el funcionamiento normal de la tarea aún en el caso de que el sistema nuevo falle y tenga que volver a alguna de las fases de desarrollo. La sustitución implica el desarrollo de programas para la interconexión de ambos sistemas y para traspasar la información entre ellos, evitando la duplicación del trabajo de las personas encargadas del proceso de datos, durante el tiempo en que estos sistemas funcionen conjuntamente.

Ahora bien, cabe indicar que el ciclo de vida en cascada es el paradigma más conocido y ampliamente usado en la ingeniería de software, sin embargo, ha sido criticado debido a los problemas que se plantean al intentar aplicarlo en determinadas situaciones, por ejemplo:

1.- En realidad los proyectos no siguen un ciclo de vida estrictamente secuencial como propone el modelo. Siempre hay iteraciones.

2.- Es difícil que se puedan establecer inicialmente todos los requisitos del sistema. Normalmente los clientes no tienen conocimiento de la importancia de la fase de análisis o bien no han pensado, en detalle, qué es lo que desean que haga el software.

3.- El ciclo de vida *clásico* requiere la definición inicial de todos los requisitos y no es fácil acomodar en él las incertidumbres que suelen existir al comienzo de todos los proyectos y,

4.- La mayor parte de los errores se detectan cuando el cliente puede probar el programa (no antes) y en ese contexto es más costoso corregir.

Todos estos problemas son reales pero aún así es mejor desarrollar un software siguiendo este tipo de modelo, además describe una serie de pasos genéricos que son aplicables a cualquier otro paradigma.

USO DE TÉCNICAS DE CUARTA GENERACIÓN

Entendemos por ésto a un conjunto muy diverso de métodos y herramientas que tienen por objeto facilitar el desarrollo de software; todos tienen algo en común: Facilitan el desarrollo del software al especificar algunas características del mismo a alto nivel; así, la herramienta genera automáticamente el código fuente a partir de estas especificaciones.

Los tipos más habituales de generadores de código cubren uno o varios de los siguientes aspectos:

1.- Acceso a bases de datos utilizando lenguajes de consulta de alto nivel derivados normalmente de SQL. Con ello no es necesario conocer la estructura de los ficheros o tablas ni de sus índices.

2.- Generación de código. A partir de una especificación de los requisitos se genera automáticamente toda la aplicación.

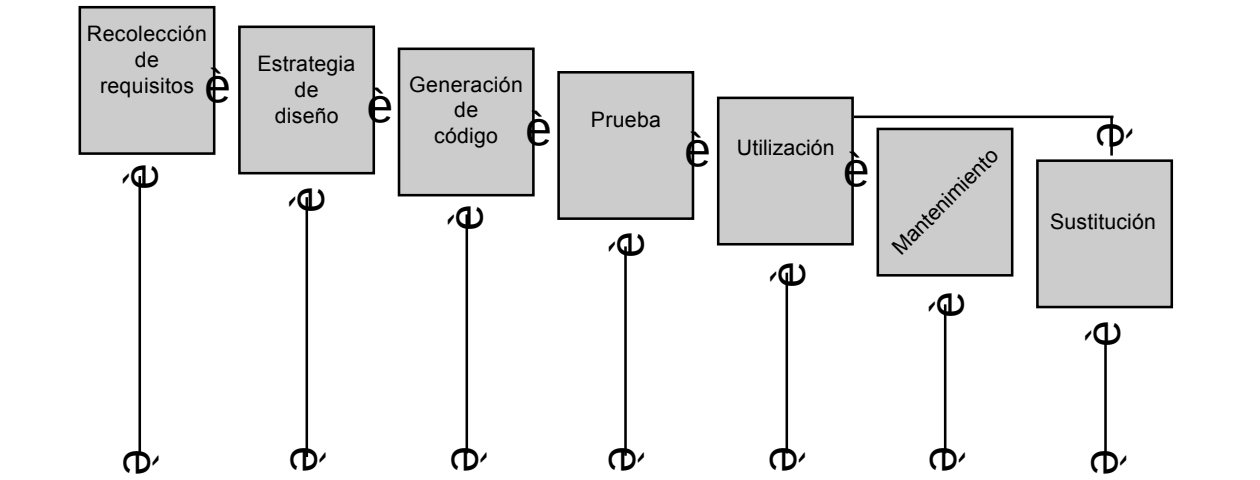
3.- Generación de pantallas. Permiten diseñar la pantalla dibujándola directamente, incluyendo además el control del cursor y la gestión de errores de los datos de entrada.

4.- Gestión de entornos gráficos.

5.- Generación de informes.

Esta generación automática permite reducir la duración de las fases del ciclo de vida clásico, especialmente en la fase de codificación quedando el ciclo de vida según se indica en la gráfica 7 de la siguiente página.

Al igual que en otros paradigmas, el proceso comienza con la recolección de requisitos que pueden ser traducidos directamente a código fuente usando un generador de código, sin embargo, el problema es el mismo que se plantea en el ciclo de vida clásico: es muy difícil que se puedan establecer todos los requisitos desde el comienzo, el cliente



Gráfica 7

puede no estar seguro de lo que necesita o, aunque lo sepa, puede ser difícil expresar la forma que precisa la herramienta de cuarta generación. Si la especificación es pequeña, podemos pasar directamente del análisis de requisitos a la generación automática de código sin realizar ningún tipo de diseño, pero si la aplicación es grande se producirán los mismos problemas que si no usamos técnicas de cuarta generación: Mala calidad, dificultad de mantenimiento y poca aceptación por parte del cliente; por lo tanto es necesario realizar cierto grado de diseño puesto que el propio generador se encarga de una parte de los detalles del diseño tradicional, por ejemplo, de la descomposición modular, de la estructura lógica y de la organización de los ficheros, entre otros.

Las herramientas de cuarta generación se encargan también de producir automáticamente la documentación del código generado, pero ésta es ordinaria y por ello difícil de seguir. Es necesario complementarla hasta obtener una documentación con sentido.

Con respecto a las pruebas, podemos suponer que el código generado es correcto y acorde con la especificación y que no contiene los típicos errores de la codificación manual, pero en cualquier caso es necesaria la fase de pruebas, en primera instancia para comprobar la eficiencia del código generado (la generación automática de los accesos a bases puede producir un código muy eficiente cuando el volumen de información es grande), también para detectar los errores en la especificación a partir de la cual se generó el código y por último, para que el cliente compruebe si el producto final satisface sus necesidades.

El resto de las fases del ciclo de vida usando estas técnicas es igual a las del paradigma del ciclo *clásico* ya que no es más que una adaptación a las nuevas herramientas de producción de software.

Como conclusión, podemos decir que mediante el uso de técnicas de cuarta generación no se han obtenido los resultados previstos cuando estas herramientas comenzaron a desarrollarse a principios de los años 80 (estos resultados incluían la desaparición de la codificación manual y con ello de los programadores, incluso de los analistas, al poder encargarse el propio cliente con pequeños conocimientos técnicos de manejo del genera-

dor), puesto que los avances en procesamiento de lenguaje natural (siempre ambiguo) no han sido demasiado grandes ni se han desarrollado lenguajes formales de especificación con la potencia expresiva necesaria, sin embargo, estas herramientas consiguen reducir el tiempo de desarrollo de software eliminando las tareas más repetitivas y tediosas y aumentan la productividad de los programadores, por lo que son ampliamente utilizadas en la actualidad, especialmente si nos referimos al acceso a bases de datos, la gestión de la entrada-salida por terminal y la generación de informes, y forman parte de muchos de los lenguajes de programación que se usan actualmente, sobre todo en el campo del software de gestión.

CONSTRUCCIÓN DE PROTOTIPOS

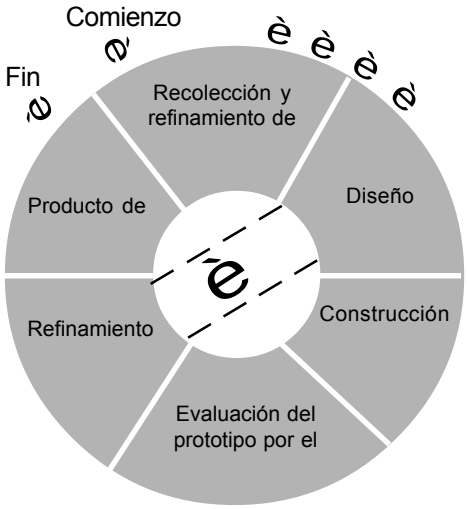
Para resolver las dificultades que presenta el modelo de ciclo de vida en cascada se ha propuesto un modelo basado en la construcción de prototipos. En primer lugar, hay que determinar si el sistema que tenemos que desarrollar es un buen candidato a utilizar el paradigma de ciclo de vida de construcción de prototipos o el modelo en *espiral*. En general, cualquier aplicación que presente mucha interacción con el usuario, o que necesite algoritmos que puedan construirse de manera evolutiva, partiendo de lo general a lo más específico, no obstante, se debe tener en cuenta la complejidad: si la aplicación necesita que se desarrolle una gran cantidad de código para tener un prototipo que enseñar al usuario, las ventajas de la construcción se verán superadas por el esfuerzo de desarrollar un prototipo que al final habrá que desechar o modificar. También hay que tomar en cuenta la disposición del cliente para probarlo y sugerir modificaciones en los requisitos. También es conveniente construirlos para probar la eficiencia de los algoritmos que se van a implementar o para comprobar el rendimiento de un determinado componente del sistema, por ejemplo, una base de datos o el soporte hardware, en condiciones similares a las que existirán durante la utilización del sistema.

Es bastante frecuente que el producto desarrollado presente un buen rendimiento durante la fase de pruebas antes de entregarlo al cliente pero puede ser muy ineficiente o inviable en el momento de almacenar o procesar el volumen real de información que debe manejar, en estos casos la construcción de un prototipo y la realización de pruebas de rendimiento sirven para decidir, antes de empezar la fase de diseño, cuál es el modelo más adecuado para el soporte hardware o cómo deben hacerse los accesos a la base de datos para obtener buenas respuestas en tiempo cuando la aplicación esté ya en funcionamiento.

En otros casos el prototipo servirá para modelar y poder mostrar al cliente cómo va a realizarse la entrada-salida de datos en la aplicación, de forma que pueda darse una idea del sistema final pudiendo entonces detectar deficiencias o errores en la especificación aunque el modelo no sea más que un esbozo. De acuerdo a lo anterior, un prototipo puede tener alguna de las tres formas siguientes: En texto o ejecutable en la computadora, que describa la interacción hombre-máquina y los listados del sistema; uno que implemente algunos subconjuntos de la función requerida y que sirva para evaluar el rendimiento de un algoritmo o las necesidades de capacidad de almacenamiento y velocidad de cálculo del sistema final y, finalmente, un programa que realice en todo o en parte la función deseada pero que tenga características (rendimiento, consideración de casos particulares,

etc.) que deban ser mejoradas durante el desarrollo del proyecto.

La secuencia de tareas del paradigma de construcción de prototipos puede verse en la gráfica 8. Este paradigma se inicia con la recolección de información útil, luego se procede a diseñar el prototipo. Se tratará de un diseño rápido, centrado sobre todo en la arquitectura del sistema y en la definición de la estructura de las interfaces más que en aspectos procedurales de los programas; a partir del diseño lo construiremos.



Gráfica 8

Existen herramientas especializadas en generar prototipos ejecutables a partir del diseño; otra opción sería utilizar técnicas de cuarta generación, en cualquier caso, el objetivo es siempre que la codificación sea rápida, aunque sea en detrimento de la calidad del software generado.

Una vez listo el prototipo hay que presentarlo al cliente para que lo pruebe, sugiera modificaciones y en su caso, lo apruebe. Él puede ver una implementación de los requisitos que ha definido inicialmente y sugerir las modificaciones necesarias en las especificaciones para que satisfagan mejor sus necesidades, a partir de esos comentarios y los posibles cambios que sean necesarios en los requisitos se procederá a construir un nuevo prototipo y así sucesivamente hasta que los requisitos queden totalmente formalizados y se pueda entonces empezar con el desarrollo del producto final.

En resumen, el prototipo es un proceso que sirve fundamentalmente para la fase de análisis de requisitos, pero lleva consigo la obtención de una serie de subproductos que son útiles a lo largo del desarrollo del proyecto.

Gran parte del trabajo realizado durante la fase de diseño rápido (especialmente la definición de pantallas e informes) puede ser utilizada durante el diseño del producto final, además, tras realizar varias vueltas en el ciclo de construcción de prototipos, el diseño del mismo se parece cada vez más al que tendrá el producto final.

Durante la fase de construcción de prototipos será necesario codificar algunos componentes del software que también podrán ser reutilizados en la codificación del producto final aunque deben ser optimizados en cuanto a su tiempo de respuesta. No obstante, hay que tener en cuenta que el prototipo no es el sistema final, puesto que normalmente apenas es utilizable ya que será demasiado lento, demasiado grande, inadecuado para el volumen de datos necesarios, contendrá errores (debido al diseño rápido), será demasiado general (sin considerar casos particulares, que debe tener el sistema final) o estará codificado en un lenguaje o una plataforma inadecuada.

Hay que tener en cuenta que un análisis de requisitos incorrecto o incompleto, cuyos errores y deficiencias se detecten a la hora de las pruebas o tras entregar el producto al cliente, nos obligará a repetir de nuevo las fases de análisis, diseño y codificación, que habíamos realizado cuidadosamente, pensando que estábamos desarrollando el produc-

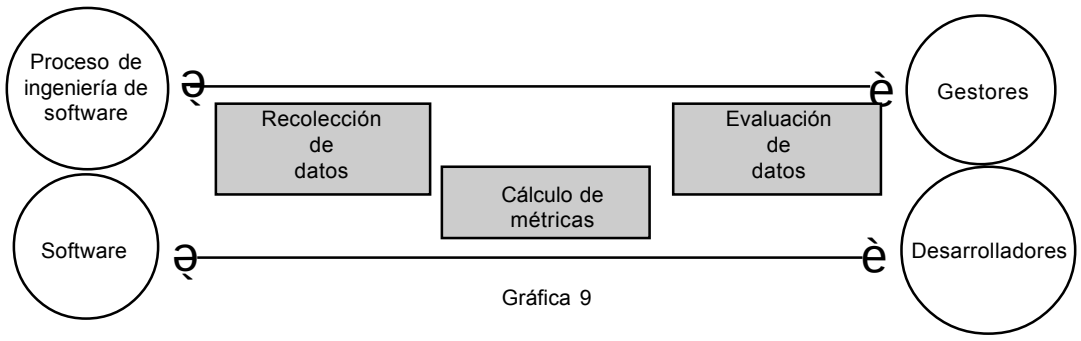
to final.

Uno de los problemas que suelen aparecer siguiendo el paradigma de construcción de prototipos, es que con frecuencia éste pasa a ser parte del sistema final, bien sea por presiones del cliente que quiere tener el sistema funcionando lo antes posible o bien porque los técnicos se han acostumbrado a la máquina, al sistema operativo o al lenguaje con el que se desarrolló el prototipo. Se olvida que éste ha sido construido de forma acelerada sin tener en cuenta consideraciones de eficiencia, calidad o facilidad de mantenimiento, incluso que las elecciones de lenguaje, sistema operativo o máquina para desarrollarlo se han hecho basándose en criterios objetivos que puedan ser adecuadas para el producto final; por ello, utilizar el prototipo en el producto final conduce a numerosos errores latentes: Ineficiente, poco fiable, incompleto o difícil de mantener, en general de poca calidad, y eso es precisamente lo que queremos evitar aplicando la ingeniería del software.

MODELO EN ESPIRAL

El modelo en *espiral* combina las principales ventajas del modelo de ciclo de vida en *cascada* y del modelo de construcción de prototipos; proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos, mucho más realista que el ciclo de vida *clásico* y permite la utilización de prototipos en cualquier etapa de la evolución del proyecto. Este es un modelo relativamente nuevo, fue propuesto en 1988, y no ha sido tan usado como los anteriores, aunque es de esperar que se extienda cada vez más.

Otra característica de este modelo es que incorpora en el ciclo de vida el análisis de riesgos; los prototipos se utilizan como mecanismo de reducción del riesgo permitiendo finalizar el proyecto antes de haber iniciado el desarrollo del producto final, si el riesgo es demasiado grande. En la gráfica (9) se muestra el modelo del ciclo de vida en espiral y en



Gráfica 9

donde se definen cuatro tipos de actividades, representadas cada una en un cuadrante, a saber:

1.- Planificación.- Consiste en determinar los objetivos del proyecto, las posibles alternativas y las restricciones. Esta fase equivale a la de recolección de requisitos del ciclo de vida clásico e incluye la planificación de las actividades a realizar en cada interacción.

2.- Análisis de riesgo.- Una de las actividades de la planificación de proyectos es el análisis de riesgos. El desarrollo de cualquier proyecto complejo lleva implícito una serie de riesgos, unos relativos al propio proyecto (los que pueden hacer que fracase) y otros a las decisiones que deben tomarse durante su desarrollo (los riesgos asociados a que una de estas determinaciones sea errónea). Ahora bien, el análisis de riesgos consiste en cuatro actividades principales:

a) Identificar los riesgos.- Pueden ser: Del proyecto (presupuestarios, de organización, del cliente, etc.), técnicos (problemas de diseño, codificación, mantenimiento), del negocio (riesgos de mercado: Que se adelante la competencia o que el producto no se venda bien).

b) Estimación de riesgos.- Consiste en evaluar, para cada riesgo identificado, la probabilidad de que ocurra y las consecuencias, es decir, el costo que tendrá en caso de que ocurra.

c) Evaluación de riesgos.- Consiste en establecer niveles de referencia para el incremento del costo, de duración del proyecto y para la degradación de la calidad, que si se superan, harán que se interrumpa el proyecto.

d) Gestión de riesgos.- Consiste en supervisar el desarrollo del proyecto de forma tal que se detecten los riesgos tan pronto como aparezcan, se intente minimizar sus daños y debe existir un apoyo previsto para las tareas críticas. (Aquellas que más riesgo encierran)

3.- Ingeniería.- Consiste en el desarrollo del sistema o de un prototipo del mismo.

4.- Evaluación del cliente.- Consiste en la valoración por parte del cliente, de los resultados de la ingeniería.

En la primera interacción se definen los requisitos del sistema y se realiza la planificación inicial del mismo, posteriormente se analizan los riesgos del proyecto, basándose en los requisitos iniciales y se procede a construir un prototipo del sistema. Después el cliente procede a evaluarlo y con sus comentarios se afinan los requisitos y se reajusta la planificación inicial empezando el ciclo. En cada una de las interacciones se realiza el análisis de riesgos, teniendo en cuenta los requisitos y la reacción del cliente ante el último prototipo.

Si los riesgos son demasiado grandes se terminará el proyecto, aunque generalmente se continúa avanzando a lo largo de la *espiral*. Con cada interacción se construyen sucesivas versiones del software cada vez más completas y aumenta la duración de las operaciones del cuadrante de ingeniería, obteniéndose al final el sistema completo.

La diferencia principal con el modelo de construcción de prototipos, es que en éste éstos se usan para perfilar y definir los requisitos, al final, el prototipo se desecha y comienza el desarrollo del software siguiendo el ciclo clásico. En cambio, en el modelo en

espiral, los prototipos son sucesivas versiones del producto, cada vez más detalladas (el último es el producto en sí) y constituyen su esqueleto por lo tanto deben construirse siguiendo estándares de calidad.

RESUMEN: ANÁLISIS

Independientemente del paradigma que se utilice, del área de aplicación y del tamaño y la complejidad del proyecto, el proceso de desarrollo de software contiene siempre una serie de fases genéricas existentes en todos los paradigmas: La definición, el desarrollo y el mantenimiento. Dentro de la fase de definición los pasos concretos dependen del modelo de ciclo de vida utilizado, en general se realizarán tres específicas:

1.- Análisis del sistema.- Define el papel de cada elemento de un sistema informático, estableciendo cuál es el papel del software dentro de ese.

2.- Análisis de requisitos del software.- Proporciona el ámbito del software, su relación con el resto de los componentes del sistema; antes de empezar a desarrollarlo es necesario hacer una definición más detallada de la función de éste. Al respecto existen dos formas de realizar el análisis y refinamiento de los requisitos del software, por una parte se puede hacer un análisis formal del ámbito de la información para establecer modelos del flujo y la estructura de la información, posteriormente se amplían unos modelos para convertirlos en una especificación del software; la otra alternativa consiste en construir un prototipo que será evaluado por el cliente para intentar consolidar los requisitos. Los requisitos de rendimiento y las limitaciones de recursos se traducen en directivas para la fase de diseño. El análisis y definición de los requisitos es una tarea que debe llevarse a cabo conjuntamente por el desarrollador de software y por el cliente, mientras que la especificación de requisitos es el documento que se produce como resultado de esta etapa.

3.- Planificación del proyecto de software.- Durante esta etapa se lleva a cabo el análisis de riesgos, se definen los recursos necesarios para desarrollarlo y se establecen las estimaciones de tiempo y costo. El propósito de esta etapa de planificación es proporcionar un indicador preliminar de la viabilidad del proyecto de acuerdo con el costo y con la agenda que se haya establecido. Posteriormente, la gestión del proyecto durante el desarrollo realiza y revisa el plan de proyecto de software.

DESARROLLO

Dentro de esta fase los pasos concretos dependen del modelo de ciclo de vida utilizado, en general se realizarán cuatro tareas específicas:

1.- Diseño.- Traduce los requisitos a un conjunto de representaciones (gráficas, en forma de tabla o basadas en algún lenguaje apropiado) que describen cómo van a estructurarse los datos, cuál va a ser la arquitectura de la aplicación, cuál va a ser la estructura de cada programa y cómo van a ser las interfaces. Es necesario seguir criterios de diseño que nos permitan asegurar la calidad del producto. Una vez finalizado el diseño es

necesario revisarlo para asegurar el cumplimiento de los requisitos del software.

2.- Codificación.- Se traduce a un lenguaje de programación, dando como resultado un programa ejecutable. La buena calidad de los programas desarrollados depende en gran medida de la calidad del diseño. Una vez codificados los programas deben revisarse, tanto su estilo como su claridad, y se comprueba que tenga una correspondencia con la estructura definida de los mismos en la fase de diseño. El listado fuente de cada módulo (o el programa fuente en soporte magnético) pasa a formar parte de la configuración del sistema.

3.- Pruebas.- Una vez que tenemos implementado el software es preciso probarlo para detectar errores de codificación, de diseño o de especificación. Las pruebas son necesarias para encontrar el mayor número posible de errores antes de entregar el programa al cliente. Es necesario probar cada uno de los componentes por separado (cada uno de los módulos o programas) para comprobar el rendimiento funcional de cada una de estas unidades. A continuación se procede a integrar los componentes para probar toda la arquitectura del software, su funcionamiento y las interfaces. En este punto hay que comprobar si se cumplen los requisitos de la especificación, para ello se puede desarrollar un plan y un procedimiento de pruebas y guardar información sobre los casos y los resultados de las mismas.

4.- Garantía de calidad.- Una vez terminada la fase de pruebas, el software está casi preparado para ser entregado al cliente.

MANTENIMIENTO

Se centra en los cambios que va a sufrir el software a lo largo de su vida útil. Como ya hemos dicho, estos cambios pueden deberse a la corrección de errores, a cambios en el entorno inmediato o a los requisitos del cliente dirigidos normalmente a ampliar el sistema.

La etapa de mantenimiento vuelve a aplicar los pasos de las fases de definición y de desarrollo pero en el contexto de un software ya existente y en funcionamiento.

Ante todo no hay que ser dogmático en la elección de los paradigmas para la ingeniería del software ya que la naturaleza de la aplicación dicta el método a utilizar.

CAPÍTULO II

VIDEOJUEGOS

INTRODUCCIÓN

En este capítulo nos enfocamos hacia la investigación de los videojuegos. Al igual que el anterior, tiene como finalidad entender todos los aspectos teóricos y el entorno en que se desenvuelven.

Dentro de ese contexto nos enfocaremos a su comercialización en todos sus rubros y su aspecto pedagógico, paralelamente estudiaremos teóricamente la estructura de un videojuego, es decir, desde su análisis hasta su desarrollo, sin embargo, en este capítulo en particular, se hará el estudio de manera general y en los subsecuentes se analizará detalladamente cada una de las etapas del ciclo o paradigma.

HISTORIA

En términos generales el videojuego no ha perdido vigencia; en la actualidad se retoma el interés en los juegos clásicos que ganan terreno nuevamente en el mercado, los jugadores de éstos recuerdan la rica historia de la industria, por ejemplo, *Asteroid-64* de Crave es una versión moderna de la que surgió en 1979. Cabe indicar que los inicios de *Asteroid´s* son una versión actualizada del *Computer Space* de Nolan Bushnell, el cual era, a su vez, una copia de la guerra espacial de Steve Russell.

Space Invaders está una vez más compartiendo el mercado con *Centipede*, *Frogger* y *Pong*, y éste último es solamente una variante mejorada del juego de Willie Higinbotham mostrado en osciloscopio.

Ahora bien, no se puede entender el desarrollo de los juegos de video sin contemplar a las compañías que los crearon, por ejemplo *Atari* era una empresa estadounidense pero con un nombre japonés, mientras que *Sega*, japonesa, fue comenzada por un estadounidense; por su parte *Magnavox*, la compañía fundadora de la industria, con el paso del tiempo fue adquirida por *Phillips*, y posteriormente esta parte fue comprada por *Nintendo* (que se traduce como: *Deja la suerte al cielo*), que a la postre sería la que renovó el videojuego y lo popularizó nuevamente. Sin embargo, *Sony*, negocio importante en la producción de electrónicos e inventor de la videocasetera, desarrolló una consola de videojuegos que llegó ser el producto más vendido de todos los tiempos.

Cabe mencionar que en 1889 Fusajiro Yamauchi creó la compañía *Marufuku* para fabricar y distribuir *Hanafuda*, un juego japonés de cartas, y en 1907 comenzó a fabricar juegos de cartas occidentales. En 1951 cambió de nombre a *Nintendo Playing Card Company.*

Por su parte, en 1891, Gerard Phillips estableció una compañía en Holanda para construir lámparas incandescentes y otros productos eléctricos. En 1918, Konosuke Matsushita, creó la *Matsushita Electric Housewares Manufacturing Works*, que posteriormente se diversificaría en una serie de empresas, entre ellas *Panasonic*.

En 1932 la *Connecticut Leather Company* fue establecida por un inmigrante ruso llamado Maurice Greenberg cuyo giro era la distribución de productos de piel para los zapateros. A principio de los años 50, su hijo, Leonard, fabricó una máquina para cortar la

piel y la empresa se transformó en COLECO y a finales de la década, Leonard construyó una máquina de inyección de plástico integrándo a la empresa a esa industria.

En 1945, Harold Matson y Elliot Handler, fabricaban marcos artesanales, poco después crearon la empresa *Mattel*; dos años después, en 1947, Akio Morita y Masaru Ibuka fundarían la *Tokio Telecommunications Engineering Company*, que posteriormente se convertiría en *Sony*. En 1951 Ralph Baer, un ingeniero de la compañía Loral, la cual desarrollaba dispositivos electrónicos sofisticados para aviones militares, es instruido para construir un aparato de TV que se consideraba el mejor del mundo; Baer sugiere que se integre una cierta clase de juego interactivo al aparato para distinguirlo de otros pero la idea fue relegada.

En 1954, David Rosen, veterano de la guerra de Corea, exportó los juegos operados por fichas a Japón, que habían adquirido popularidad en las bases militares estadounidenses durante la guerra; para 1960 fabricó sus propios juegos y adquirió una pequeña empresa en Tokio registrándola con el nombre de *Sega* (Service Games).

En 1958 como un esfuerzo por atraer visitantes al *Brookhaven National Laborotories* de Nueva York, el físico Willy Higinbotham inventó un juego interactivo parecido al tenis de mesa en un osciloscopio, fue mejorado por él mismo un año después adaptándolo para un monitor de 15 pulgadas, sin embargo, y creyendo que no había inventado nada, no lo patentó.

Al inicio de la década de los 60, el estudiante Steve Russell del MIT crea *Spacewar*, el primer juego interactivo de computadora, en un *Mainframe Digital PDP-1.* Limitado por los avances tecnológicos de la época, los gráficos fueron caracteres de texto ASCII, y la gente pudo jugarlo en un dispositivo que ocupaba el espacio de una casa pequeña; para el año de 1962, Nolan Bushnell ingresó a la Escuela de Ingeniería en la Universidad de Utah y expuso por primera vez el *Spacewar* de Russell.

En el año de 1966, Ralph Baer retomó la idea del uso alterno para los aparatos de televisión, es decir, para juegos interactivos; para entonces trabajaba en el despacho jurídico de *Sanders Associates* que se interesaría en la idea y le otorgaría los fondos para desarrollar el proyecto, al año siguiente presentó un juego de persecución y posteriormente desarrolló uno de tenis, además, modificó un arma de juguete que distinguía un punto de luz en la pantalla. En 1968 se patentaron los juegos de Baer.

En 1970 *Magnavox* obtuvo la licencia de los juegos interactivos de *Sanders Associates*, paralelamente, y con la ayuda de Ted Dabney, Bushnell acondicionó una recámara de su casa como un taller para construir una nueva versión de la consola del *Spacewar* que sería conectada a la televisión y a la que denomina *Computer Space Nutting Associates*. Esta consola fue comprada por una empresa y Bushnell fue contratado como supervisor de la fabricación.

En términos generales, hasta 1970, es la antesala que sustentó los juegos en video; pero los años siguientes, de 1971 a 1977 es, propiamente, la etapa inicial.

En 1971 se fabricó el primer videojuego de consola, 1500 máquinas del *Computer Space*, que incluían un aparato de televisión (blanco y negro) de 13 pulgadas, pero el público lo encontró muy difícil de jugar.

Para 1972 *Magnavox* comienzó a fabricar el *Odyssey*, que fue el sistema del videojuego de Baer, e inició su comercialización mostrándolo a los distribuidores de Esta-

dos Unidos. Para el 24 de mayo de ese mismo año, se llevó a cabo una convención en Burlingame, California, Bushnell asistió y reportó que *Odyssey* no era interesante y que no representaba una competencia para *Computer Space*, sin embargo, el juego no se vendió y Bushnell tuvo que renunciar; después se asoció con Dabney para iniciar una compañía de diseño de videojuegos y que otras los distribuyeran. Llaman originalmente a su empresa *Syzygy* (la configuración de tres líneas continuas de tres cuerpos celestes), pero ese nombre estaba siendo usado por un negocio de material para techos, y se inclinaron por el de *Atari*, un término del juego japonés *Go*, cuyo significado es equivalente al jaque mate en el ajedrez. La empresa contrató a Al Alcorn cuya función sería la programación de los juegos, como es inexperto, Bushnell hace que programe un sencillo juego de tenis al que denominan *Pong* por dos razones; primero porque es el sonido que emite una pelota de Ping-Pong cuando es golpeada por una raqueta, y segundo porque el nombre Ping-Pong ya estaba registrado. Este juego se intentó comercializar a través de fabricantes de consolas pero no tuvieron interés, así que ésta la desarrolló personalmente Bushnell; el juego fue puesto a prueba en un bar llamado *Andy Capps* y sería todo un suceso.

Por su parte *Magnavox* vendió el *Odyssey* a través de sus propios almacenes, y a pesar de que el comprador creyó que el juego funcionaba solamente con la televisión *Magnavox*, la empresa vendió 100 mil unidades, sobre todo porque era el juego más parecido, en su versión casera, al famoso *Pong*. Similitud que provocó pleitos legales, *Magnavox* demandó a *Atari* alegando que *Pong* había sido copiado de uno de los juegos del *Odyssey*, sin embargo, tras descubrir que el manual del juego de *Magnavox* del año anterior presentaba la firma de Nolan Bushnell, *Atari* decidió ir a la corte y compró los derechos para la fabricación de los videojuegos llegando al arreglo de que cualquier otra empresa que deseara fabricarlos tenía que pagar a *Magnavox* y a Sanders el derecho respectivo.

En 1973, mientras que *Pong* se convertía en todo un fenómeno, muchas empresas lo imitaron, tanto que para finales de ese año existían más de 25 cuyos juegos competían con él. Esa competencia provocó que Ted Dabney, socio de *Atari*, vendiera su parte a Bushnell.

Al año siguiente se fabricó *Touch Me*, en el que cuatro botones brillaban aleatoriamente y el usuario debía presionarlos en orden subsecuente; este juego fue observado por Ralph Baer quien lo rediseñó con colores aleatorios que permitían distinguir las luces, fue vendido a Milton Bradley quien lo fabricó con el nombre de *Simon* que a la postre llegaría a ser uno de los grandes éxitos de los juegos interactivos de todos los tiempos. Después de este triunfo, *Atari* fabricó una nueva versión de *Touch Me* pero no alcanzó el éxito esperado.

Por su parte, Harold Lee, empleado de *Atari*, propuso un nuevo juego tipo casero de la versión de *Pong*. Bushnell le otorgó a Lee, a Alcorn y al ingeniero Bob Brown libertad para desarrollarlo. Ese mismo año, la empresa japonesa *Namco* adquirió *Atari Japón* y se integraba a la industria del videojuego.

En 1975 Tom Quinn, gerente de compras de *Sears Roebuck*, ofreció comprar todos los juegos *Pong* caseros que *Atari* pudiera fabricar. Esta le informa que podía producir solamente 75.000 unidades anuales, *Sears* (a través de Quinn) ofreció también el financiamiento si *Atari* se comprometía a duplicar su producción y a otorgarle la exclusividad del producto durante el primer año, *Atari* aceptó y *Sears* vendió el juego con su ima-

gen corporativa *(Sears Tele-Games)* logrando ser el de mayor venta durante la temporada navideña de ese año. En este tiempo apareció el primer juego de computadora, *Gunfight*, fabricado por *Midway Games* y desarrollado por Taito. Fue el primer juego que utilizó un microprocesador en vez de circuitos de estado sólido.

En 1976 y atraídas por el éxito de *Atari*, diversas compañías fabricaron consolas caseras de videojuegos, pero solamente COLECO recibió el registro aunque no la aprobación de la FCC dado que el producto hacía interferencia en la radiofrecuencia, COLECO contrató los servicios de Ralph Baer para corregir el problema y *Telstar* hizo su debut.

Atari se fue consolidando y en 1977 abrió el primer pizza time theatre, una combinación de restaurant/consola el cual presentaba animales robotizados con movimiento, juegos electrónicos y comida. La mascota del restaurante era una rata llamada *Chuck E. Cheese*. Paralelamente fabricó e introdujo la consola programable (basada en cartuchos), un sistema informático de video, VCS, conocido más adelante como el *Atari 2600.* Mientras tanto Bally fabricó una consola programable llamada *Bally Professional Arcade*, con un costo de $350.00 dólares, precio que provocó que no tuviera el éxito esperado.

A partir de 1978, comenzó la etapa de la producción y comercialización del videojuego, periodo al que se le denomina Edad de Oro (1978-1981), inició cuando Bushnell dejó la presidencia de *Atari* (Ray Kassar sería el nuevo presidente) y firmó un contrato por cinco años que lo comprometía a no lucrar contra la compañía que él inició, pero adquirió los derechos de *Pizza Time Theatre*. En marzo de ese año *Nintendo* de Japón fabricó su versión de *Computer Othello*, un juego muy sencillo basado en el de mesa del mismo nombre, mientras que *Atari* fabricó su versión de *Foot Ball* que ofrecía un revolucionario control llamado *Trackball*. Por su parte *Midway* importó *Space Invaders* de Taito.

Ahora bien, tanto *Foot Ball* como *Space Invaders*, rompieron todos los records de ventas con casi las mismas ganancias, sin embargo, *Foot Ball* fue perdiendo interés conforme la temporada de fútbol llegaba a su fin, mientras que la de *Space Invaders* continuaba.

Atari inició una nueva etapa en el mercado y comenzó a vender su línea de computadoras modelo 400 y 800 para competir con *Apple*, pero el público asociaba a *Atari* con juegos de video y las computadoras nunca las tomaron con seriedad.

Magnavox fabricó el *Odyssey 2*, una consola programable que tenía un teclado de membrana incorporado. *Cinematronics* produjo *Space Wars*, un juego similar al *Space Computer* de Bushnell, pero éste ofrecía gráficos de vector, forma muy básica de esquemas de polígonos y que aparecieron por primera vez en aplicaciones de videojuegos, sin embargo, para 1979 *Atari* desarrolló *Cosmos*, una máquina programable interactiva que ofrecía holográficos dentro de los gráficos, que son elementos de estética y que no intervienen en la funcionalidad del juego y lanzó el *Lunar Lander*, su primer juego de vectores-gráficos. A pesar de la popularidad de este juego se decidió detener la producción para iniciar la fabricación del *Asteroid's* en las consolas de *Lunar Lander*; *Asteroid's* fue un juego originalmente diseñado por Lyle Rains y Ed Logg para el sistema *Cosmos* y que se convertiría en el producto de mayor venta de *Atari* de todos los tiempos.

Los *Asteroid's* crearon nuevas características para las consolas, por ejemplo, permitieron introducir tres caracteres al final del juego para identificar a la persona que logra altos puntajes; fue tan popular que cerca de 80,000 unidades se vendieron en los Estados

Unidos, pero fue poco popular en otros países.

También en 1979, *Sega* fabricó *Monaco GP,* un juego de manejo; posteriormente apareció *Pro Monaco GP* en 1980 y el *3D Super Monaco GP* en 1989.

Por su parte *Milton Bradley Electronics* fabricó *Microvision*, una unidad interactiva programable que incluía su propia pantalla, mientras que *Atari* produjo, en exclusiva para VCS, la versión casera de *Spacer*, paralelamente *Mattel Electronics* introdujo la consola de juegos *Intellivision*, que sería el primer competidor serio del VCS, ésta tenía mejores gráficos y un precio más accesible, además, prometía fabricar un periférico opcional que mejoraría el *Intellivision*.

Poco después varios programadores de VCS dejarían *Atari* por una disputa en los créditos del juego, crearon *Activision*, que se convertiría en el rival del software casero de VCS, cabe señalar que mientras *Atari* no daba crédito a sus programadores, *Activision* los reconocía incluyendo sus nombres en el empaque y en la comercialización del juego.

En 1980, el diseñador de *Atari*, Ed Rottberg, creó *Battlezone*, que sería el primer juego tridimensional de primera-persona. Consiste en manejar un tanque en una batalla virtual, seleccionando los blancos en un escenario de guerra. A la postre este juego sería importante en el entrenamiento militar ya que el gobierno estadounidense fabricó una versión similar con propósitos de entrenamiento militar.

Por su parte *Namco* fabricó *Pac-Man*, que llegaría a ser el juego más popular de todos los tiempos. Más de 300,000 unidades se vendieron en todo el mundo, un poco más de 100,000 solamente en los Estados Unidos. Originalmente se llamó *Puck-Man*, pero fue retitulado después de que los ejecutivos de la empresa consideraron que el nombre podía sufrir mutilaciones en la letra P, lo que podía desalentar a los padres para permitir que sus niños lo jugaran.

En otro contexto, *Sega* obtuvo los derechos para fabricar y manufacturar la versión japonesa del *Missile Command*. Paralelamente, Minoru Arakawa, yerno del jefe de *Nintendo Japón*, Hiroshi Yamauchi, abrió *Nintendo América* cuya sede fue Nueva York, después la trasladaría a Seattle, Washington. Las ventas disminuyeron considerablemente y obligaron a la aparición del ordenador *Othello*; a la par, la consola *Astrovision* cambió de nombre a *Astrocade*, una vez que Bally, su diseñador, la vendiera.

Casi a finales del año de 1980, Williams, fabricante de máquinas de *Pin Ball* de Chicago, construyó la primera versión en videojuego; fue diseñado por Eugene Jarvis, y se tituló *Defender* que sería el primer juego de manivelas impulsoras.

Prácticamente la carrera por el control y dominio de los videojuegos comenzaba. Para 1981 la empresa Nintendo tuvo la oportunidad de convertir una gran cantidad de videojuegos fracasados en algo exitoso y que ganaría mucho dinero. Shigeru Miyamoto crea el *Donkey-Kong*. El juego consiste en que un pequeño carpintero intenta salvar a su novia *Pauline* de un enloquecido gorila; el personaje originalmente se llamaba *Jumpman* y posteriormente se le llamó *Mario* por el parecido físico con su creador, Mario Segali.

Dado el *boom* de los videojuegos, diversos programadores renunciaron a sus empresas, como sucedió con los de *Atari* e *Intellivision* que se asociaron y constituyeron *Imagic*, una compañía que prometía desarrollar software para VCS e *Intellivision*. *Atari* negoció los derechos para desarrollar más los títulos de chispas como *Pac-Man* para VCS y paralelamente fabricó *Tempest*, un videojuego de vector de colores basado en una tec-

nología de gráficos todavía inestable y que era propensa a fallas, pero que reuniría a jugadores devotos.

Es importante señalar que para finales de 1981 las ventas de consolas en los Estados Unidos alcanzaron los 5 mil millones de dólares, dado que los estadounidenses pasaban más de 75,000 horas en los videojuegos. Bajo ese parámetro apareció *Electronic Games*, la primera revista dedicada enteramente al videojuego, cuyos fundadores fueron Arnie Katz y Bill Kunkel.

Los siguientes dos años sería de gran impacto para la industria.

En 1982 COLECO fabricó *Colecovision*, una consola basada en juegos de cartuchos, aparato que no solamente era superior en gráficos y sonidos, sino que también soportaba juegos de la empresa *Nintendo*. Ésta concedió a COLECO la licencia de fabricación y uso de *Donkey-Kong* y de *Donkey-Kong Junior*. COLECO construyó una excelente interface de *Colecovision*, los puertos del *Atari VCS*; el *Intellivision* fabricó un adaptador para el *Colecovision* que permitía jugar con cartuchos VCS y se alía con *Sega, Konami* y *Universal*.

Magnavox fabricó un juego llamado *K.C. Munchkin* para el *Odyssey 2*, sin embargo, *Atari* consideró que era muy similar a *Pac-Man*, demandó y ganó el juicio, por lo que *Magnavox* tuvo que retirar el juego del mercado. Apareció entonces la versión anticipada de *Pac-Man* para el VCS, que desafortunadamente no se asemejaba al juego de consola, el público se desencantó con *Atari*. Fabricó en tan sólo seis semanas el *E.T.*, un juego programado por Howard Scott Warshaw que no logró el éxito esperado, posteriormente fabricó la consola 5200 para competir con *Colecovision* (aunque había sido diseñada originalmente para competir con el *Intellivision*), cabe señalar que de acuerdo a los gráficos y a los chips de audio encontrados en las computadoras personales de *Atari*, los juegos para el 5200 fueron esencialmente refabricados para correr en la consola VCS ó 2600, sin embargo, la máquina fue incompatible con los cartuchos del 2600 hasta que se introdujo un adaptador en donde éstos pudiesen utilizarse en la consola 5200, conjuntamente fue un éxito ofrecer un Joystick.

Por su parte *General Consumer Electronics* (GCE) fabricó el *Vectrex*, que sería la primera y única consola basada en una tecnología de gráficos de vectores. Incluía un juego (*Minesweepers*, una copia improvisada de *Asteroid's*) y un Joystick analógico de cuatro botones.

Durante ese año *Midway* creó a *Ms. Pac-Man*, convirtiéndose en el videojuego más grande de consola en la historia; se vendieron más de 115,000 unidades solamente en los Estados Unidos, pero *Namco*, que no estaba implicado con *Ms. Pac-Man*, desarrolló el *Pac-Man* mejorado, juego radicalmente diferente y que sería estupendo para los consumidores japoneses. A la par apareció un chip para acelerar el *Pac-Man* original y para cambiar sus características y laberintos.

En 1983 Nolan Bushnell entró de nuevo a la industria; unificó a *Videa* y renombra a la compañía *Sente Games*, se asoció con *Midway Games* para fabricar títulos de consola, como el simple, pero adictivo juego del hockey *Hattriz*. Desafortunadamente nunca se pudo establecer en el mercado. En marzo, *Atari* anunció un nuevo proyecto (que hasta entonces era secreto) llamado: *El proyecto falcón*. Que sería una división de la empresa *Ataritel*, cuyo objetivo era incorporarse al mercado de las telecomunicaciones.

Cinematronics fabricó *Rick Dyer´s Dragon´s Lair* animada por Don Bluth, que sería el primer juego de consola en ofrecer la tecnología de disco láser.

Por su parte *Comodore* construyó la *Comodore 64*, una computadora de bajo costo y de gran alcance que superó a cualquier consola videojuego.

Nintendo fabricó en Japón la computadora familiar llamada *Famicom*, diseñada como un juguete, incluía el *Donkey-Kong, Donkey-Kong Junior* y *Popeye*. Ahora bien, debido al dominio de *Atari* en el mercado, *Nintendo* no pretendía vender *Famicom* fuera de Japón, pero ofreció a *Atari* los derechos de distribuirlo en otros países, y se firmó el contrato en junio de 1983. Por su parte COLECO presentó su computadora *Adam* en ese mismo mes, que incluía el *Donkey-Kong*. Atari, que tienía los derechos de *Nintendo*, acusó de violación al contrato y amenazó con no distribuir el *Famicom* y demandar penalmente. Como era de suponer *Nintendo* amenazó a su vez a COLECO porque ésta tenía solamente los derechos del videojuego pero no para introducirlo en computadoras.

La controversia golpeó a *Atari* cuando se revela que Ray Kassar vendió $250,000 dólares en piezas del inventario el 6 diciembre de 1982, un día antes de que *Atari* hiciera el anuncio de la causa del faltante en el inventario, Kassar fue sustituido por James Morgan el 6 de septiembre del año siguiente .

A partir de entonces se va desarrollando una crisis en las empresas productoras de videojuegos, muchas de ellas cierran ya que no pueden competir con juegos más baratos.

En 1984 Milton Bradley comenzó a distribuir el *Vectrex* después de adquirir GCE. Una de las primeras políticas de la empresa fue reducir el precio del *Vectrex* para hacerlo más competitivo, pero no logró fortalecerse y finalmente canceló el producto.

COLECO utilizó todos sus recursos para fabricar el *Adam* paralelamente al *Colecovision*, pero el 60% del producto fue regresado por defectuoso. Otra empresa que cerró por pérdidas fue *Mattel Electronics*, pero fue comprada por Terry Valeski, vicepresidente de *Mattel*, quien la renombró *Intellivision Inc.*

Mientras que la industria del videojuego comenzó a derrumbarse, *Nintendo* anunció que podía fabricar *Famicom* en Estados Unidos y *Atari* introdujo nuevos productos en ese verano, entre ellos el 7800, una consola avanzada que aceptaba cartuchos del 2600, y el *Mindlink*, un controlador de manos libres que se sujeta en la cabeza.

Haciendo frente a las pérdidas, *Warner Comunication* vendió la división de *Atari* a Gack Tramiel, el hombre que había fundado *Comodore* y que había sido obligado a salir de esa compañía a principios del año. *Warner Comunications* conservó la división de consolas y la retitula como *Atari Games*, mientras que Tramiel hace lo mismo y la denomina *Atari Corporation* (la cual incluía la división de videojuegos y computadoras) y anunció inmediatamente que la nueva empresa no tenía ninguna intención de vender las consolas de videojuegos y que comercializaría una nueva línea de computadoras de 16-bits.

A pesar de la crisis en el mercado de los videojuegos, comenzaría una nueva etapa entre los años de 1985 y 1988 que sentaría las bases para reintegrar a la industria al desarrollo.

El año de1985 fue prácticamente el repunte a partir de las pruebas de mercado de *Nintendo Entertainment System* (NES) en Nueva York, quien negoció con los comerciantes para comprar todo el inventario no vendido, no obstante, y estableciendo una gran cantidad de títulos originales desarrollados por ellos, NES fue un impacto en la produc-

ción. Otro episodio que marcó el repunte y siguiendo el liderazgo de *Apple* en la fabricación de *Macintosh*, *Atari* adquirió un desafío con el 68000 de Motorola de 16-bits basado en el 520 ST, internamente llamado *Jackintosh*.

Por su parte el programador ruso Alex Pajitnov diseñó *Tetris*, un juego simple pero adaptable del rompecabezas que se puede jugar en las PC.

En 1986 Nintendo fabricó NES para Estados Unidos, fundamentalmente por el éxito obtenido en Nueva York, y su debut se llevó a cabo con el *Super Mario Brothers*, que a la postre sería un rotundo éxito. Después de la introducción del NES, *Sega* construyó su *Sega Master System* (SMS) en los Estados Unidos. *Atari* revaluó la popularidad de los videojuegos y decidió impulsar nuevamente la fabricación de la consola 7800, sin embargo, *Nintendo* que era la empresa con mayor consolidación, vendió en proporción 10 a 1 con relación a sus competidores en el mercado estadounidense, y en Japón dio a conocer un periférico disk-drive para el *Famicom*, junto con los títulos *Zelda, Golf* y *Fútbol*. Fue tal el éxito de *Nintendo* que varias compañías firmaron un contrato con ella como desarrolladoras asociadas, mientras que la mayoría de los viejos socios de *Atari*, como *Namco*, hacían mejores juegos para el sistema de *Nintendo*.

En 1987 *Nintendo* conservó el mercado que iba creciendo, manteniendo fuera de él a *Sega* y a *Atari*. Este último fabricó juegos para el 2600, los cuales fueron ignorados por la prensa, y construyó puertos de acceso para la 7800; mientras que *Nintendo* fabricó *Zelda* en cartucho para los Estados Unidos, pero decidió no importar el costoso periférico disk-drive.

Los juegos como *Kid Icarus* y *Metroid* ofrecían gráficos fabricados para NES. La compañía *Tonka*, de carros de juguete, compró los derechos de distribución en los Estados Unidos a SMS, colocándolos en más almacenes que *Sega* y con ello permitiendo que compitieran mejor contra el NES.

Por su parte *Atari* fabricó el sistema de juego *Atari XE* (XEGS), que era básicamente empaquetar su vieja computadora 800. El XEGS utilizó cartuchos compatibles con la línea de computadoras XE de 8-bits de *Atari* e incluía dos juegos, el *BarnyardBlaster* y el *Flight Simulator II,* una pistola de luz y un teclado desmontable, pero fracazó rápidamente. NEC contruyó la PC-Engine en Japón (máquina 16-bits). La consola ofrecía un procesador de 16-bits para gráficas.

Para 1988 *Atari Games* estableció *Tengen*, que era una subsidiaria que producía los juegos para consolas. Esta empresa inició como desarrollador asociado de *Nintendo* de los juegos compatibles con NES. *Atari Games* llevó a Nintendo a la corte demandando que tenía un monopolio ilegal en la industria del videojuego, llevando a cabo prácticas prohibidas como, determinación de precios y uso de una tecnología no patentada para el desarrollo del software lógico de NES; *Tengen* descubrió una manera de producir juegos compatibles con NES sin la aprobación de *Nintendo* y anunció que los desarrollaría, fabricaría y distribuiría sin la conformidad de ésta.

Por su parte, e incapaz de recuperarse del desastre de *Adam*, la mayoría del catálogo de archivos de COLECO pasó a Milton Bradley y a Parker Brothers.

Debido al gran crecimiento de *Nintendo* con su consola NES, la industria del videojuego, entre 1989 y 1992, se expandió en el mercado casero.

En 1989 *Tengen* adquirió los derechos de *Tetris* y comenzó a vender el popular jue-

go, sin embargo, se descubre que *Tengen* ha comprado los derechos a *Microsoft* pero no sobre los originales, *Nintendo* aprovechó la circunstancia y adquirió rápidamente los derechos legítimos de *Tetris* para fabricarlo bajo su nombre, mientras que la versión de *Tengen* es retirada del mercado.

Nintendo fabricó su *Gameboy*, cuyo sistema viene con *Tetris* y contenía una pequeña pantalla monocromática, con ello inició una carrera histórica de ventas. La versión del *Super Mario* para Gameboy *(Super Mario Land)* y un juego de béisbol fueron fabricados de inmediato.

NEC trajo el *PC-Engine* a América y lo llamó *TurboGraf X-16*, también fabricó el lector de CD's portátil asociado al *Turbo-Graf X-16*, por primera vez los juegos fueron almacenados en discos compactos.

Sega, después del éxito en Japón, fabricó el *Genesis* de 16 bits, que es empaquetado con una versión del juego *Altered Beast.* Los primeros esfuerzos de comercialización lo colocaron como una experiencia que sustancialmente mejoraría las consolas caseras de videojuegos anteriores.

Casi al finalizar el año, *Epix* introdujo una consola portátil llamada *Handygame*, *Atari* compró los derechos y fabricó el *Lynx*. Después de publicitar los juegos comenzó a desarrollar un sinnúmero de títulos compatibles con el sistema 7800, pero más costosos que el *Gameboy*. El *Lynx* tenía bajas ventas y se rumoraba que *Atari* detendría la producción.

En 1990 *Nintendo* contruyó *Super Mario 3*, el videojuego más vendido de todos los tiempos, y a pesar de la competencia del *Genesis* y de *TurboGraf X-16*, el NES gozó de su mejor año. *Nintendo* de Japón dio a conocer su *Super Famicon*, un sistema de 16- bits con mejor audio y gráficos 3D y *Super Mario 4 SM World* fue dado a conocer a los jugadores japoneses.

SNK, un desarrollador de *Nintendo* y fabricante de juegos como *Ikary Warriors* y *Crystalis*, y *Nintendo*, fabricaron la consola de 24 bits llamada *Neo-geo* y las versiones caseras de dichos juegos. Los gráficos y sonido superaron al *Genesis* y al *TurboGraf X-16* pero el precio limitaba las ventas.

Por su parte *Sega* seguía fortaleciéndose con las ventas de consolas; aseguró los derechos para *Genesis* del videojuego *Strider* de *Capcom*, un juego desconocido pero asombroso el cual se consideraría el juego del año. *After Burner II, E-Swat* y otros éxitos de *Sega* llegaron a Estados Unidos.

NEC fabricó el *Turbo Express*, que fue la versión del *TurboGraf X-16* portátil, y que fue la primera máquina de juegos portátiles con resultados óptimos en juegos de consola.

La empresa *Commodore* anunció su CDTV (Comodore Dynamic Total Vision), que fue básicamente una computadora sin teclado, fue el primer sistema interactivo casero con software educativo y fue vendido en discos compactos y en cartuchos.

Para 1991 *Nintendo* construyó el *Super Famicom* en Estados Unidos y lo llamó el *Super NES* (SNES), mientras que *Sega* introdujo *Sonic*, del cual se esperaba que compitiera contra el NES y el SNES. Comenzó una lucha entre *Mario* y *Sonic* y que finalmente ganaría *Super Mario World.*

Las compañías *Sony* y *Nintendo* anunciaron planes para que la primera desarrollara un lector de CD's para trabajar con el SNES.

Apareció *Genie* de *Galoob Toys*, un juego cuyo dispositivo permitía ganar de una

forma sencilla en los de NES. *Nintendo* vio el juego *Genie* como herramienta para reducir el valor, a largo plazo, de sus juegos y procuró limitar las ventas de éste.

Capcom fabricó *Street Fighter II.* Mientras que *Atari* anunció el desarrollo del *Panter*, un nuevo sistema de 32-bits que competiría contra *Sega* y *Nintendo*.

En 1992, y aunque *Atari* tenía contratos con *Nintendo*, con *Capcom* y con *Konami* hablaron con *Sega* sobre el desarrollo de *Genesis*. Fabricaron los juegos pero nunca dedicaron sus mejores esfuerzos y equipos al trabajo del software de *Sega* y éste preparó apresuradamente el *Sonic 2.* Por su parte *Sonic* llegó a ser un desafío serio para el éxito de *Mario*.

Sega fabricó el *Segacd*, pero le negó a los desarrolladores el fácil acceso a las herramientas que les permitían utilizar capacidades especiales para gráficos. *Sega* de América se enfocó al desarrollo de películas interactivas.

Por su parte JVC introdujo el *Wondermega* en Japón que fue una combinación del *Genesis* y del *Segacd*.

Sony y *Nintendo* abandonaron sus planes para desarrollar un periférico de CDs, cuando la primera ya tenía terminado el prototipo. Se rumoró que los abogados de *Sony* tenían la información necesaria para publicar las cifras del SNES o *Superfamicom* que utiliza CD's, datos que *Nintendo* mantenía en secreto. Ésta anunció sus planes para trabajar con *Phillips* y crear un cd-rom compatible con los CD-I de *Phillips*, mientras que *Sony* trabajaba en los detalles finales del SNES desarrollado para *Nintendo* y que fue retitulado *PlayStation,* paralelamente un grupo de ingenieros trabajaba sobre una máquina de videojuegos de 32-bits para desbancar a *Nintendo* en Japón y en Estados Unidos.

Apareció *3DO*, una compañía iniciada por *Electronics-Arts* fundada por Trip Hawkins, que anunció la construcción de una nueva consola de 32-bits. Esta compañía recibió ofertas de *Panasonic*, *Time-Warner* y *MCA*, pero no planeaba fabricar la consola por sí misma dado que el sueño de Hawkins era que ésta se convirtiera en el estándar que fabricaran diferentes empresas.

Entre 1993 y 1997 la tendencia en el manejo de gráficos fue de 32-bits, e inicia una nueva etapa o Era.

En 1993 *Panasonic* fue la primera compañía en comercializar la consola *3DO* pero se enfrentó a la desventaja del precio; por su parte *Atari* decidió brincar la Era de los 32-bits y fue directamente a la de 64, lanzó *Jaguar* (fabricado en Estados Unidos por *IBM*) y proclamó ser la primera consola de juegos de 64-bits (realmente tenía dos coprocesadores de 32-bits).

Nintendo y *Sega* anunciaron sus sistemas de la siguiente generación. El proyecto de *Nintendo* fue un sistema de 64-bits desarrollado por *Silicon Graphics*, mientras que el *Saturn* de *Sega* sería un sistema de 32/64-bit.

En 1994 la *Entertainment Software Rating Board* (ESRB) fue establecida para la clasificación de videojuegos. Este organismo impuso normas, entre ellas, recomendaba las edades mínimas para ciertos juegos, asimismo clasificó al juego como violento o de riesgo.

Nintendo fabricó *Super Metroid* y comenzó su impulso para recuperar el control del mercado de 16-bits; los nuevos juegos *Super-FX*, como *StarFox*, ayudaron en los esfuerzos de la compañía contra *Sega* y sus máquinas de 32/64-bits. Fabricó también *Donkey*

Kong Country y, en una exhibición comercial, demostró que aún la lentitud del CPU del *Super NES* pudo competir con el *3DO* y el *Jaguar*, de hecho, *Donkey Kong Country* fue el videojuego mejor vendido de ese año.

Por su parte *Sega* fabricó *32X*, un periférico que permitía al *Genesis* ejecutar un nuevo conjunto de juegos de 32-bits en un intento por restar ventas al *Jaguar* de *Atari* y al *3DO* de *Panasonic*. Apareció simultáneamente *Virtual Racing* y *Star Wars* que fueron recibidos favorablemente al igual que la versión del *DOOM* de *id Software*, pero extrañamente las licencias de *Sega* no cumplieron con los requisitos, provocando incertidumbre sobre el futuro de la máquina, sobre todo porque no tenía la capacidad de producción en Japón.

Nintendo construyó *Super Gameboy,* un adaptador que permitía jugar en el SNES los cartuchos del *Gameboy* con las gráficas a color y características adicionales.

El *Saturn* de *Sega* y el *PlayStation* de *Sony* se lanzaron en el mercado japonés y al final del año los críticos señalaron al *PlayStation* como la mejor máquina.

En 1995 se anunció que el *Saturn* sería fabricado en los Estados Unidos, pero las compañías distribuidoras fueron sorprendidas con el lanzamiento y no dieron todo el apoyo. Sega y *3DO* estuvieron listos para notificar el surgimiento de una empresa conjunta que utilizaría la tecnología *3DO M2* de 64-bits, sin embargo, y aunque el trato se rompió al poco tiempo, las negociaciones continuaron durante el año.

Por otro lado el desarrollo de *3DO* fue lento con respecto al aviso del 64-bits; *Panasonic* adquirió, en última instancia, la tecnología del *M2* para uso en los juegos caseros y otros dispositivos.

Nintendo fabricó el *Virtual Boy*, una consola de juego portátil de 32-bits, ahora llamada *Ultra 64*. Los conocedores criticaron el sistema, hasta que *Nintendo* indicó que las ventas del *Gameboy* habían sido exitosas a pesar de sus limitaciones; las críticas cesaron hasta que las ventas bajaron dramáticamente con respecto a las proyecciones.

Sony contruyó el *Play Station* en los Estados Unidos y las ventas fueron exitosas; la colección de buenos títulos recibió sensibles comentarios de los consumidores, mientras que las ventas del *Jaguar* de *Atari* continuaban bajando a pesar del periférico CD, el cual había levantado las esperanzas de los partidarios del juego, no así las expectativas de los ejecutivos de *Atari*.

Nintendo retrazó el lanzamiento de su sistema de 64-bits, *Ultra 64*, argumentando que los seguidores de sus productos apoyaban los 16-bits producidos algunos meses después, y mostró eventualmente el *Nintendo 64* (el nuevo nombre para los Ultra 64). *Super Mario 64* impresionó a los usuarios, pero surgieron rumores que señalaban que el software era muy pequeño con respecto al desarrollo de la máquina.

Por su parte el público se confundió después de la introducción del sistema *Saturn* de *Sega* con varios periféricos para el *Sega Genesis* por lo que desistió en sus planes del *Neptum*, un sistema que combinaba la tecnología del *Genesis* con el 32X y los periféricos del *Sega CD.*

El lanzamiento del *N64* en Japón provocó alborotos y debido a la improvisación en la distribución la gente pudo adquirir la máquina en almacenes locales sin problemas. *Nintendo* tuvo ventas record y rápidamente agotó su inventario inicial del hardware, pero después de algunas semanas las ventas del *N64* pararon prácticamente debido a la ca-

rencia del software. Los consumidores rechazaron comprar el tercer título del lanzamiento del *N64* fundamentalmente por los rumores que se desataron que indicaban que el desarrollo del software no era correcto, provocando que nuevos títulos tardaran en aparecer varios meses.

Al inicio del año de 1996 *Sega* bajó el precio de sus productos, pero el prestigio de sus desarrolladores continuaba en descenso, los rumores persistían en torno a que la compañía detendría su desarrollo en hardware y que se enfocaría a la traducción casera de otros sistemas. Por su parte *Panasonic*, la cual poseía la tecnología *M2* de *3DO*, no hizo ninguna demostración pública de la máquina. Mientras tanto, los juegos en CD parecían ser la única opción para el futuro y aparecieron grandes dudas sobre la viabilidad de los cartuchos.

Sega fabricó *Virtual Fighter III* en Japón y en Estados Unidos; y una versión del *Saturn* se anunció inmediatamente.

Un sinnúmero de juegos de simulación comenzaron a gozar de popularidad, incluyendo el *Esqui*, *Snowboarding* y el *Jet Sky* de *Namco* y *Sega*, mientras que los juegos de lucha saturaban el mercado, previamente saciado por los de tiro y aventuras, así las consolas se enfocaron a combinaciones más costosas de entretenimiento.

Las ventas japonesas del *Saturn* fueron altas, y las americanas fueron decepcionantemente bajas.

Sony bajó el precio del *PlayStation* y anunció una larga variedad de promocionales excitantes.

La corporación *Atari* se unió a JTS, un fabricante de discos duros para computadora, y anunció oficialmente el cese en la fabricación de la línea *Jaguar*, un tema que había sido discutido de manera no oficial por meses.

Después de diversas noticias dando a conocer los planes para un dispositivo portátil de 32-bits a color, *Nintendo* reconoció que *Atlantis* era una máquina de videojuegos con arquitectura RISC que había estado bajo desarrollo por sus filiales de Europa y Japón pero los planes del lanzamiento fueron eclipsados por el surgimiento del *Nintendo 64*.

Nolan Bushnell resurgió en la industria como el presidente de *Aristo Games*, compañía que hacía las estaciones de la Internet para las consolas.

El *N64* fue fabricado en los Estados Unidos y más de 1.7 millones de unidades fueron vendidas en tres meses, mientras las ventas de *Sony* superaron los $12 millones de dólares por día en la temporada navideña y el *PlayStation* se aferró a su lugar mundial como el número uno en los juegos de consola.

La industria del juego de video tuvo un año altamente provechoso y los precios del software de 32-bits comienzaron a mostrar una volatilidad excepcional. Apareció el 15 de junio en Pittsburgh el *Videotopia*, un lugar que se enfocaba a exhibir la historia de los videojuegos, específicamente en el centro de la ciencia de Carnegie.

A finales del año *Nintendo* detuvo todo el desarrollo del *Virtual Boy*, debido a una falla. La empresa culpó del incidente a su diseñador, Gumpei Yokoi, empleado por 30 años que era también responsable del éxito del *Game boy*; Yokoi dejó *Nintendo* en deshonra y comenzó su propia compañía.

En 1997 *Sony* anunció una base instalada de 3.2 millones de unidades en los Estados Unidos, un tercio de éstos había sido vendido durante la temporada navideña de 1996,

mientras que *Nintendo* dio a conocer que el *N64* pudo haber vendido cerca de 2.5 millones de consolas durante la temporada si se hubiera podido fabricar esa cantidad.

Sony dio a conocer las estadísticas sobre el *PlayStation* que fue el sistema más popular en el mundo, éstas demostraron que fueron vendidas 5 millones de unidades en Japón, 4 millones en los Estados Unidos y 2.2 millones en Europa; estos números se duplicaron casi cuatro meses después, cuando se vendió la unidad 20 millones. Los analistas creyeron que la popularidad de *PlayStation* continuaría al menos hasta 1998.

Sony fabricó *Yaroze* en los Estados Unidos, que permitía a los usuarios diseñar juegos compatibles con *PlayStation* en sus computadoras. Por su parte *Nintendo* construyó una nueva consola y una versión compacta del SNES. Curiosamente, el desarrollo del SNES surgió cuatro meses después de que la empresa anunciara que no desarrollaría más juegos de 16-bits y señaló, en el verano, que su juego esperado desde hace mucho tiempo, el *64DD*, *Legend of Zelda 64*, sería fabricado primero en cartuchos, eso provocó rumores de que el *64DD* nunca sería construido, pero *Nintendo* lo negó y señaló que las versiones en cartucho serían fabricadas al mismo tiempo, es decir, a finales de 1997. Algunas semanas después anunció que el lanzamiento sería pospuesto hasta marzo de 1998. La fecha del lanzamiento de *Zelda* no se cambió, en noviembre, *Nintendo* mostró la primera versión del *Legend of Zelda 64* e indicó que el *64DD* sería retrasado hasta junio de 1998.

Muchos analistas indicaron que *Sega* renunciaría al *Saturn* a favor de una consola de 64-bits que tendría un módem incorporado y un lector de 6 u 8 velocidades. *Sega* desmintió los rumores y dijo que no había ninguna en 1997, pero los rumores demostraron ser correctos y se reveló que Lockheed Martin había revisado varios planes para una consola nueva. Al final del año decidieron diseñar su propia consola; el nuevo sistema, llamado *Black Belt*, tenía que ser construido alrededor de un subsistema de gráficos, el *Voodoo Graphics* de *3dfx Interactivo* y trabajaría con el procesador de *Hitachi*, que prometía tener una velocidad de 200MHz y una capacidad de procesamiento de 350 millones de instrucciones por segundo. El *Black Belt* tendría un lector de CD, sin hacerlo compatible con DVD.

Sega de Japón también estaba trabajando en un sucesor del *Saturn*, al cual llamaron *El Dural* que utilizó un chip de PowerVR; después que decidió ir con este sistema, la mayoría del equipo de diseño americano salió de la compañía. *3dfx Interactivo* inició un pleito de contrato contra *Sega*. Antes de finalizar el año, la compañía renombró a la consola como *Katana*.

Los rumores dentro de *Sony* sugerían que la empresa estaba planeando una nueva consola de 64-bits con un RAM adicional y que el sistema emplearía un chip R4000 y un lector de CD de cuatro velocidades. Los ejecutivos de la compañía no confirmaron el rumor y señalaron que el *PlayStation 2* no estaría disponible hasta 1998.

Nintendo de Japón anunció que bajarría el precio del *N64*; por su parte *Sony* anunció que también lo haría con el *PlayStation* en el Reino Unido y Australia. Una semana después lo hizo.

George Harrison advirtió que *Nintendo* no tenía planes de bajar los precios, pero 3 días después, un periódico japonés, dio a conocer que lo haría con el *N64*, no obstante, su filial de América indicó que el aviso se había traducido mal en los Estados Unidos y reafir-

mó que mantendría los precios, finalmente lo hizo dos semanas después. Por su parte *Sega* bajó el precio del *Saturn* en junio; pero continuaba siendo tan popular en Japón que anunció que vendería el software exclusivo de *Saturn* a través de sus consolas; el primero sería un disco llamado *Digital Dance Mix*, el cual ofrecía las animaciones en 3D con canciones del japonés Namie Amuro.

Sega fabricó su segunda consola con tecnología *Super-Polygon*, *Super GT Scud Race*, en Japón y en América. *Capcom* construyó su esperado *Street Fighter III* en Japón, llamado simplemente *Three* en Estados Unidos y comunicó que se uniría con la compañía japonesa de juguetes *Bandai* la cual se denominaría *Sega Bandai*, sin embargo, diversos problemas abundaron ya que ésta planeaba continuar desarrollando el software para el *PlayStation*. *Bandai* reprobó la unión y explicó que la compañía no necesitaba a *Sega* para aumentar sus ventas y fabricó el *Tamagotchi* en Japón, que fortificó las ganancias de la empresa; el juguete se convirtió rápidamente en una obsesión en Japón, incluso, lo fabricó en los Estados Unidos. El primer almacén que lo ofreció logró vender 30,000 unidades en tan sólo tres días. *Bandai* anunció versiones para la PC y *Game Boy,* después de poco tiempo, otras compañías, como *Tiger Electronics*, fabricó sus propias mascotas virtuales y construyó un sistema portátil monocromático, llamado *Game.com* para competir con el *Game Boy.* El primero ofreció varios suplementos incorporados tales como solitario, una calculadora, una agenda telefónica y un calendario; también incluyó una tecnología touch-screen y podía conectarse a un módem de PC para el acceso a un servicio de e-mail.

Nintendo anunció que estaba listo para reasumir el desarrollo de su sistema portátil de color, el *Atlantis*. Su desarrollo había sido detenido debido al éxito inesperado del *Game Boy* y sería un sistema de 32-bits que permitía hasta 30 horas de juego sin necesidad de baterías nuevas.

Telegames sorprendió a muchos fabricando seis nuevos juegos, incluyendo el *BreakOut 2000*, para el *Jaguar* de Atari y dos para el *Lynx*, pero no recibieron mucha distribución y desaparecieron rápidamente.

Un periódico japonés señaló que durante una entrevista un ejecutivo de *Matsushita* dijo que el *M2* había sido terminado y que la división entera para la fabricación de consolas había sido cerrada, la compañía publicó inmediatamente una declaración en donde explica que habían entendido mal y señalaron que el *M2* estaría listo para lanzarlo junto con diez juegos más, y abundó, que fue detenido su lanzamiento debido a que había muchos productos en el mercado para competir. Poco tiempo después, el presidente de *Matsushita*, Yoichi Morishita, indicó que no se fabricaría como una consola de videojuegos y dijo que la tecnología del *M2* podía ser empleada como parte de una consola de uso múltiple para multimedia, el tipo de consola en que el *3DO* y *Phillips CD-I* fueron originalmente construidas.

VM Labs anunció oficialmente que estaba trabajando en una consola de videojuegos y que estaría disponible a finales de 1998, su presidente, Richard Miller, dijo que *Project X* era una realidad, pero no señaló las características de la nueva consola sólo que sería fabricada por más de una compañía.

Por otro lado, la etapa considerada como Edad Moderna comprende desde 1998 a la actualidad.

En 1998 *Sega* reconoció oficialmente su nuevo sistema 128-bits pero el nombre continuaba siendo desconocido durante la mayor parte del año, aunque se sabía su nombre original en código: *Dural and Black Belt,* pero se renombró como *Katana*. Al mismo tiempo dio a conocer que la nueva consola utilizaría el sistema operativo *Windows CE* de *Microsoft*, que significaría conversiones más fáciles del juego y de la PC. *Katana* fue dado a conocer en mayo, y un aspecto que resaltaba fue su sistema visual de memoria (VMS), un dispositivo que se conectaba a la consola y que también podía utilizarse como un dispositivo independiente del juego compatible con los *Tamagotchis*. *Sega* anunció en noviembre que el *Katana* sería fabricado en Japón y que no sería dado a conocer en Estados Unidos hasta 1999; la filial de América comenzó la planificación para gastar $100 millones de dólares para su lanzamiento. A mediados del año anunció otro cambio en el nombre, esta vez el sistema se convirtió en *Dreamcast*, finalmente salió a la venta en Japón el 27 de noviembre. Los 150,000 sistemas iniciales que se ofrecieron para la venta se agotaron inmediatamente, junto con 132,000 copias del *Virtual Fighter III.*

El *Naomi*, prometía ser una nueva consola que tenía las mismas capacidades que el *Model 3* de *Sega*, pero más económica. Dado que el *Naomi* y el *Dreamcast* utilizaban el mismo chip, la conversión de títulos entre las consolas sería muy simple, mientras más capacidad era ofrecida; las ranuras de VMS fueron proveídas en las máquinas de *Naomi* para la transferencia de datos para y desde el *Dreamcast*.

Por otro lado un estudio de mercado indicaba que la marca no era muy importante para los consumidores de videojuegos, así que *Sega* decidió no incluir su nombre en el *Dreamcast*, irónicamente, *Majesco*, una nueva compañía de New Jersey, determinaba que el nombre de *Sega* era importante y compró los derechos del *Genesis* y fabricó el nuevo *Genesis 3* con la marca como parte central del empaquetado.

Majesco planeó construir las versiones baratas del *Game Gear, Saturn* y *Pico*, también el nuevo software para el *Super Nintendo* y el *Genesis*. Uno de los sorprendentes títulos fue *Frogger* del cual *Majesco* había adquirido los derechos de *Hasbro Interactivo.*

Después de la fusión cancelada entre *Sega* y *Bandai*, el presidente de la primera, Hayao Nakayama, se retiró y fue sustituido por el presidente de *Sega* de América, Shoichiro Irimajiri. Otro de los problemas que mantienía a la empresa preocupada era la desaparición de tres divisiones: *Sega of América*, de *SegaSoft* y *Sega Entertainment*; junto con la desaparición anunció que estaba tirando la toalla y paró la distribución del *Saturn* en Estados Unidos.

Los rumores iniciaron, se decía que *Sony* trabajaba en el *PlayStation 2*. Las piezas que conformaban el nuevo sistema comenzaron a ser fabricadas por desarrolladores independientes; a mediados del año se admitió que la nueva consola estaba en desarrollo y que podía ser basada en tecnología DVD. La incógnita más interesante era que el nuevo sistema estaría disponible en el año 2000. A la mitad del año los rumores se hicieron más fuertes, de hecho, *Sony* hizo equipo con *Toshiba* para desarrollar el chip de la nueva consola; las predicciones indicaban que el procesador del RISC tendería a 250MHz, ligeramente más rápido que *Dreamcast Sega.*

Cuando el *64DD* de *Nintendo* no aparecía se asumía que nunca se daría a conocer. La compañía fabricó *Nintendo 64 Expansion Pack*, el cual se suponía sería el complemento del *64DD*. Este sistema se conectaba al puerto de expansión del *N64* y duplicaba la

memoria del sistema a 8MB. Paralelamente fabricó *The Legend of Zelda* para el N64 en noviembre y reportó 325,000 anticipos para *Zelda* haciéndolo uno de los cartuchos más esperados de todos los tiempos. Los que reservaron el juego recibieron una edición especial.

Entre su fecha de lanzamiento y el final del año, *Nintendo* vendió 2.5 millones de copias del juego, ganando $150 millones de dólares en ventas y anunció que *Pokemon* (abreviatura de *Monstruos de Bolsillo*) vendría a Estados Unidos. Juego que recibió atención mundial cuando provocó ataques epilépticos a casi 700 espectadores japoneses, pero las caricaturas serían editadas en los Estados Unidos por lo que no tendrían el mismo efecto entre los espectadores.

Los juegos fueron fabricados para el *Game Boy* en dos ediciones en septiembre, se convirtieron en el producto de *Nintendo* más rápidamente vendido. La compañía también fabricó un dispositivo tipo *Tamagotchi* llamado *Pikachu* de bolsillo, el más popular de los *Pokemones*.

En abril de 1999 *Nintendo* de Japón construyó el *Game Boy Light,* una versión de la popular unidad portátil que ofrecía un contraste, pero no especificaba si el dispositivo se comercializaría en Estados Unidos. Inmediatamente después de su producción fabricó sus primeros periféricos para el *Game Boy*: Una cámara fotográfica y una impresora. A pesar de las imágenes de baja resolución, las dos unidades se conviertieron en éxitos para el juego de bolsillo, sin embargo, firmó la sentencia de muerte para el *Game Boy* de bolsillo cuando contruyó el *Game Boy Color*; aunque originalmente se intentó hacerlo en un 100 % compatible con el primero, los desarrolladores comenzaron a fabricarlos en versiones blanco/negro y a color, después de poco tiempo, los juegos trabajaban solamente en el *Game Boy Color.*

El *Tiger* continuaba con su *Game.com* y fabricó un modelo más pequeño llamado *Game.com Pocket Pro* y *Bandai* introdujo el *WonderSwan*, un sistema que fue desarrollado en parte por Gunpei Yokoi, el diseñador del *Game Boy* de *Nintendo*. El *WonderSwan* podía mostrar juegos horizontal y verticalmente en una pantalla de alta resolución en blanco y negro y podía también mostrar pequeños videos. *Bandai* esperó dar a conocer el *WonderSwan* antes de finales de 1998 y predijo que vendería entre 3 y 4 millones de unidades en 1999.

Por su parte SNK anunció el *Neo-Geo Pocket,* un sistema portátil que trabajaba independiente y en conjunto con *Sega Dreamcast.* La unidad sería compatible con los VMS de *Dreamcast* pero podía hacer otras cosas; ofrecía un pequeño Joystick y una pantalla monocromática tan grande como la del *Game Boy,* aunque originalmente no estaba planeado fabricarlo en los Estados Unidos, SNK anunció una versión doméstica para abril de 1999.

Después del éxito de *Frogger* para la PC y el *PlayStation*, *Hasbro Interactive* compró un par de compañías relacionadas con el videojuego, la primera *Tigger Electronics,* fabricante de juegos y juguetes electrónicos, incluyendo la unidad portátil del *Game.com. Hasbro* anunció la compra de los derechos de las librerías de *Atari* por $5 millones de dólares. Los rumores comenzaron a circular sobre que *Hasbro* produciría el *Jaguar* y el *Lynx* o combinaría, de alguna manera, el *Lynx* con el *Game.com.* La empresa negó inmediatamente cualquier plan para un nuevo hardware e insistía que deseaba fabricar las versiones ac-

tualizadas del catálogo de *Atari*. El primer juego que fabricó fue *Centipede* para PC, aunque se anunció una versión para el *PlayStation*, no se pudo colocar en el mercado en 1998.

Activision construyó un compendio de 30 juegos del 2600 para el *PlayStation* pero recibió una respuesta tibia aunque el público apreciaba los esfuerzos de *Activision* para fabricar juegos clásicos; también construyó una versión actualizada de *Asteroid's* para PC y *PlayStation*. Unos meses después un Cd-rom llamado *Intellivision Lives* fue fabricado, contenía una colección de 50 juegos que se podían jugar en PC o MAC.

Sobre la primera mitad de 1998, en *VM Labs* estaban tranquilos sobre sus planes para un nuevo sistema de videojuegos, el cual tenía el código *Project X.* Todos los reportes señalaban que era un dispositivo de tipo multimedia, como el *3DO* y el *CD-I* que tendría la capacidad para ejecutar tanto juegos como software educativo. Cuando se anunció que el *Project X* no sería una consola, sino un chip instalado dentro de las consolas DVD y permitiría que los juegos fueran interactivos. A mediados del año notificaron que el *Project X* sería incluido en todos dispositivos de DVD fabricados por *Thompson* (RCA y GE) y *Toshiba*. *VM Labs* planeó recuperar su inversión cobrando por las licencias a los desarrolladores que proyectaban producir software para *Project X;* los que destacaban eran *Activision*, *Capcom, Hasbro* y *THQ*. Al final del año, *VM Labs,* hizo oficial el nombre del nuevo sistema: *NUON*.

La *Asociación de Software Interactivo Digital* (IDSA por sus siglas en inglés) anunció que 1998 fue un año de bandera para la industria del entretenimiento electrónico, únicamente durante los primeros seis meses las ventas superaron 30% a las de 1997. Desafortunadamente las noticias no fueron todas atractivas; la IDSA también señaló que la industria casera del juego de video estaba prosperando a expensas de la industria de las chispas; una víctima fue *Acclaim*, quien señaló que saldría del mercado en marzo.

Para 1999 *Nintendo* anunció una nueva consola, llamada *Dolphin* que sería construida con un microchip de 400MHz y una tecnología llamada *Gekko* y que sería fabricado por *IBM*. Esperaba enviar la nueva consola antes de la temporada navideña del 2000, y señaló que el *Game Boy Advance* sería un sistema portátil de 32-bits, el cual podía ser combinado con un teléfono celular para accesar a Internet. La compañía prometió que la nueva unidad sería compatible con el *Game Boy* y con su software de color. Paralelamente, Howard Lincoln, comunicó que planeaba retirarse como director de *Nintendo of America.*

JTS, la compañía que absorbió *Atari Corporation*, se declaró en bancarrota.

La primera *Classic Gaming Expo* abrió sus puertas en Las Vegas, entre los asistentes estaba Ralph Baer, inventor del videojuego.

Microsoft reveló que estaba trabajando en una consola casera llamada *X-Box* que, al igual que el *Dreamcast* de *Sega*, utilizaba una versión del *Windows CE* como sistema operativo.

Después de años del secreto, *VM Labs* dio a conocer su cabina en *E3*. Estaba claro que *VM Labs* no fabricaría una nueva consola de juegos, en su lugar planeó colocar su tecnología dentro de nuevos DVD y con ello ganar dinero sobre las licencias de los juegos compatibles con *NUON*.

Aunque no fue construida la unidad de bolsillo monocromática *Neo-Geo* en los Esta-

dos Unidos, *SNK Corporation of America* tuvo diversos planes para la implantación de la unidad a color. La compañía comenzó a ofrecer el sistema de 16-bits vía e-mail, posteriormente continuó distribuyendo las unidades en los almacenes de Estados Unidos.

Hasbro Interactive seguía creciendo y adquirió los derechos de *Namco* para distribuir 11 títulos clásicos de videojuegos para PC, también compró los derechos para fabricar éstos para consola.

La información sobre el *PlayStation 2* se dio a conocer a través del año y se reveló un microprocesador nuevo de Toshiba/Sony a 250MHz, doblando al *Emotion Engine. Sony* comunicó oficialmente el *PlayStation 2* en septiembre, además de jugar con éste la nueva unidad sería compatible con todos los juegos originales, así como con los CD´s de audio y con los DVDs. Se planeaba construir el *PlayStation 2* en Japón en el 2000 y en los Estados Unidos y en Europa a finales del mismo año.

Iomega anunció que produciría Zip Drive el cual sería diseñado como periférico para el *Dreamcast. Sega Japón* notificó sus planes para desarrollar y vender juegos para las unidades portátiles como el *Game Boy* de *Nintendo* y el *WonderSwan* de *Bandai*, pero no consideraba ser competidor de estos sistemas puesto que no tenía proyectos para entrar al mercado de sistemas portátiles. Poco después bajó el precio del *Dreamcast* japonés.

Debido a los disturbios en la preparatoria en Littleton, *Sega* decidió no fabricar la pistola de *Dreamcast* en Estados Unidos, por lo que fanáticos del *House of the Dead* se consideraban defraudados cuando comprendieron que el juego sería operado por el regulador estándar de *Dreamcast* usando los controles periféricos. Las compañías intermediarias prometieron fabricar pistolas de luz en lugar de *Sega*.

Sega de América reportó ganancias por $98 millones de dólares dentro de las primeras 24 horas de lanzamiento del *Dreamcast* en los Estados Unidos.

*Connectix Corporatio*n introdujo la *Virtual Game Station,* la cual emularía los juegos de *PlayStation* en *Macintosh*. *Sony* demandó a la *Connectix*, pero el proceso es denegado.

Una compañía llamada *Bleem LLC* introdujo con *IBM* un emulador de *PlayStation* para la computadoras personales.

El 4 de marzo del año 2000, *Sony* lanzó el *PlayStation 2* en Japón, vendiendo un millón de equipos marcando un nuevo record de ventas en ese país. En el mismo año se anunció el *X-Box,* un equipo *Intel 733 Mhz Pentium III* con acelerador gráfico a 250 Mhz, 64 MB en RAM y 8 MB en disco duro. Además, introdujo el *PSOne*, una versión compacta del *PlayStation*, el cual es ligeramente más grande que el *Discman* de *Sony*.

Game Boy anunció su *Songboy* que tocaba hasta 60 minutos de MP3, éstos pueden ser obtenidos por Internet a una PC y transferido al *Songboy* por el puerto USB.

Sega comenzó el servicio de Internet para el *Dreamcast Network*, en marzo. El servicio cambió de nombre a *Sedanet*, al mismo tiempo inició su servicio de ISP.

En el año 2001, surgieron rumores sobre el futuro de *Sega*, se decía que dejaría de fabricar el *Dreamcast* para marzo; otro indicaba que se dedicaría al desarrollo de software para competir con otras compañías del mismo ramo. *Sega* de Japón respondió que el primer rumor no era cierto y referente al segundo, se estaba negociando para distribuir software para *PlayStation 2* y el *Game Boy* avanzado.

Después de dos años de batallas legales, *Sony* y *Connectix Connect* sobre su *Vir-*

tual Game Station, un emulador para PC y Mac, la primera adquirió toda la tecnología del emulador.

Los videojuegos debutan en el cine, *Lara Croft, Tomb Raider, Final Fantasy,* etc. *Sony* de Japón lanzó el disco duro de 40 GB para PlayStation 2 en julio.

El ataque a las torres gemelas de la ciudad de Nueva York afectaron a la industria del videojuego, provocó que los diseñadores cambiaran sus productos, por ejemplo: *Spider-Man 2* para el *Play Station*, *The Grand Theft Auto III* y *Smugglers´Run 2* y surgen cambios para *Flight Simulator 2002* por las escenas en el World Trade Center.

IMPORTANCIA PEDAGÓGICA

El creciente uso de los medios electrónicos en la educación, particularmente las tecnologías derivadas de la informática, han propiciado el desarrollo de una nueva visión acerca de los procesos de aprendizaje. Hoy en día se está enfocando hacia los ambientes de enseñanza diseñados para crear condiciones pedagógicas, donde el conocimiento y sus relaciones con los individuos es el factor principal.

En la rama de los videojuegos la orientación de las actividades, desde una visión pedagógica, se encamina hacia la solución de problemas, a la formación de habilidades y aprender a pensar críticamente fomentando el aprendizaje a lo largo de la vida. Otro de los elementos fundamentales que se debe tomar en consideración es la formación en la producción del conocimiento, es por ello que los videojuegos se deben ubicar en escenarios enriquecidos, es decir, deben cambiar a medida que se diseñan para el aprendizaje con determinados contenidos. En ese sentido, el ideal aplicativo de la tecnología informática es propiciar modelos que permitan un aprendizaje individual y colaborativo.

Es innegable que actualmente estamos presenciando cambios importantes en todos los ámbitos de la sociedad, los cuales tienen una relación estrecha con el desarrollo de la tecnología. Hoy hay diversas maneras de concebir un ambiente de aprendizaje, que en el caso de los videojuegos, deben existir al menos cuatro componentes principales que lo conformen: El contenido como tal, el usuario, los elementos educativos y los medios.

Todos estos componentes deben estar diseñados de modo que el aprendizaje se desarrolle con un mínimo de tensión y un máximo de eficacia. Por supuesto que éstos no son exclusivos de los ambientes de aprendizaje, pero cualquier enfoque que se le de a los videojuegos tiene como base estos elementos y la estrategia didáctica es la que permite una determinada dinámica de relación entre los componentes.

Asimismo los ambientes de aprendizaje deben tener las características para responder a diversas necesidades: El individuo que aprende en su propio espacio, el aprovechar las herramientas tecnológicas y los conocimientos en la dinámica de una interrelación directa entre los componentes principales. Por lo anterior, los ambientes de instrucción pueden ser desarrollados en formas muy diversas, desde los ambientes totalmente reales hasta los virtuales; contextos que pueden prescindir total o parcialmente de la intervención de un asesor; escenarios abiertos o cerrados, de acuerdo al software, y la interacción de redes que se conectan a él; ambientes unimediales o multimediales, dependiendo de los elementos que participan.

Una de las clasificaciones más útiles que han surgido para estudiar los ambientes de aprendizaje se refiere a las posibilidades de interacción que presentan. Así, se habla

de medios de una y dos vías, para diferenciar a aquellos que operan basados en el flujo de información del usuario al videojuego, pero no a la inversa, y aquellos que permitan esa reversibilidad, sin embargo, la rápida convergencia de los videojuegos está rompiendo este esquema clasificatorio, lo que hace que en el futuro todo ambiente de aprendizaje sea interactivo.

A lo largo del tiempo la tecnología educativa ha sido concebida de diferentes formas de acuerdo a la etapa de desarrollo en que se encuentra, no obstante, al vincular las nuevas tecnologías con la educación han surgido algunas confusiones con respecto al término. En algún momento la tecnología educativa fue entendida como la introducción de los medios o recursos audiovisuales e informáticos con el objeto de apoyar la enseñanza (refiriéndose mas a una concepción de tecnología en la educación y no a una educativa), sin embargo, la tecnología por sí misma no puede tener un efecto sobre el proceso de aprendizaje si no se cuenta con un enfoque metodológico que le de sustento dentro del proceso de enseñanza.

Los videojuegos, como parte de una tecnología educativa, implican una manera sistemática de diseñar, llevar a cabo y evaluar todo proceso de aprendizaje y enseñanza en términos de objetivos específicos, basados en la investigación del aprendizaje y la comunicación, empleando una combinación de recursos humanos y materiales para conseguir un objetivo más efectivo.

En resumen, la tecnología educativa nos remite a todos aquellos elementos que en conjunto buscan hacer más eficiente el proceso de enseñanza-aprendizaje.
La tecnología como una herramienta relevante debe utilizarse para enseñar y aprender, inclusive, para diseñar experiencias y ambientes de aprendizaje y aprovechar sus posibilidades. Actualmente la tecnología comienza a revelar su potencial al producir mejoras en la enseñanza.

En estos momentos las nuevas tecnologías, como los videojuegos, poseen ciertas características que responden a un nivel más general, entre estas destacan: La interactividad, la programabilidad, la retroalimentación, la utilización de elementos de texto, imagen y audio, la posibilidad de trabajo en grupo, la conexión con usuarios distantes y el acceso inmediato a la información.

Pero veamos, desde el punto de vista de los videojuegos, cada una de estas características por separado. La interactividad en los videojuegos radica en que permite tener un flujo bidireccional o diálogo entre el usuario y la computadora. Evidentemente depende de una serie de factores como lo es la programabilidad, ya que el usuario puede diseñar, por medio de instrucciones, la funcionalidad del videojuego de modo que consiga adaptarlo a la forma más adecuada para él.

Por otra parte, la retroalimentación que recibe el usuario es importante debido a que de manera inmediata recibe la información y temática que se busca como objetivo en el videojuego. La integración de los elementos visuales y sonoros modifica de forma determinante la relación que el usuario tiene con los elementos didácticos, ya no se trata de una serie de imágenes estáticas, sino que ahora puede reforzar su aprendizaje y formación por medio de la imagen y el sonido que apoyan los contenidos.

Con esta nueva tecnología y los avances en comunicaciones, la posibilidad de establecer sesiones de juego con usuarios distantes aumenta la posibilidad de interacción de

modo que el trabajo en grupo se refuerza, logrando compartir el conocimiento de una forma inmediata y en retroalimentación.

En una investigación llevada a cabo sobre las actitudes que manifiesta la sociedad mexicana hacia la computadora, se incluyó un estudio referente a ciertas etapas de adopción de la tecnología informática, en ésta se aplicó una escala sobre el grado de adopción de la tecnología informática en ocho estados de la república, en la cual se establecen en orden progresivo seis etapas: Conciencia; aprendiendo el proceso; entendimiento y aplicación del proceso; familiaridad y confianza; adaptación a otros contextos y aplicación creativa a contextos nuevos.

Los resultados mostraron que, en general, la gente se percibe entre las etapas de adopción, esto es, entre la de entendimiento y aplicación del proceso y familiaridad y confianza, sin embargo, en otro análisis se observa cómo la distribución porcentual divide aproximadamente a una mitad de la muestra en las primeras tres etapas y la otra mitad en las siguientes tres. Es debido a ésto que se distinguen bloques bien definidos: Los que están aprendiendo y entendiendo el funcionamiento y uso de la computadora y los que se encuentran adaptando y aplicando la tecnología en su trabajo.

Una parte fundamental que puede caracterizar a cada uno de los miembros que integran los dos bloques es la experiencia que van adquiriendo con el uso de la tecnología, enfocándonos al primer grupo bajo la consideración de ampliar la población del segundo bloque, los videojuegos pueden ser una herramienta fundamental para lograr el objetivo, ya que la utilización de los mismos con ambientes amigables puede facilitar de manera sustancial la transición del primer grupo hacia el segundo, es decir, que el usuario adquiere una familiaridad y confianza con la computadora para posteriormente adaptar todas las herramientas tecnológicas hacia contextos de la vida cotidiana.

La pedagogía señala que para mejorar el proceso educativo e impulsar el desarrollo de los sujetos, la organización de los objetivos de la enseñanza debe adecuarse a su desarrollo psicológico y social actual; para que suceda es necesario tener en consideración cómo se aprende y por consecuencia se desarrolla el proceso de conocimiento, es decir, señala la importancia de la díada epistemológica y enseñanza. Estos dos supuestos consideran la idea de que la adquisición de conocimientos se basa en la experiencia que ofrecen los sentidos (empirismo) o que se fundamenta en la razón, basada en los conocimientos innatos del individuo (racionalismo).

Una herramienta básica en la orientación cognoscitiva es el videojuego, los procesos cognoscitivos son la adquisición de conocimientos del organismo sobre el medio ambiente en donde aquel es un procesador de información, a manera de explicación los videojuegos se sustentan como ambientes de aprendizaje que retroalimentan procesos cognoscitivos internos útiles para aprender, como son: Atención, memoria y otros asuntos básicos, así como los contenidos de lo aprendido (la representación del conocimiento), como determinantes esenciales de la conducta.

De acuerdo con Piaget, padre de la teoría cognoscitiva, existen varias etapas del desarrollo: El sensorio motor e inteligencia preverbal; el pensamiento simbólico y las funciones de representación; el pensamiento intuitivo; el pensamiento lógico concreto y el pensamiento lógico formal. Los videojuegos de acuerdo a su nivel de complejidad fomentan el desenvolvimiento de estas etapas con excepción de la primera.

A manera de resumen, los videojuegos bien encaminados (lejos de aspectos de violencia) son, y deben ser, parte fundamental para el proceso de educación de una sociedad, por tal motivo debemos considerar que el diseño de éstos tiene que ser, sin lugar a dudas, sustentado en bases pedagógicas para que cumplan con los métodos de aprendizaje con el objetivo de formar la personalidad del usuario, logrando que éste dicte y se riga por las normas de una sociedad que en su conjunto debe estar bien definida, sin olvidar el objetivo principal de los videojuegos, que es la diversión.

NIVEL SOCIOCULTURAL

Durante los últimos años la tecnología informática y las telecomunicaciones han tenido un desarrollo acelerado que ha marcado la forma en que se complementan las relaciones sociales. Con la integración de los medios masivos de comunicación, se han formado distintos sistemas tecnológicos cada vez más sofisticados que involucran de una manera activa e interactiva la vida del ser humano. Hablar de esto nos remite a las nuevas tecnologías de comunicación e información; éstas se han infiltrado poco a poco en casi todos los aspectos del quehacer humano, en los ambientes laborales, en los hogares e inclusive en los sistemas escolares, que al incorporarlas se han visto en la necesidad de replantear las formas tradicionales de formación social.

Tomando en cuenta la influencia y el impacto del desarrollo tecnológico sobre la sociedad, hoy en día diversas instancias se están valiendo del conjunto de nuevas tecnologías en comunicación e informática y están descubriendo sus potencialidades para eficientar las estrategias pedagógicas. Evidentemente esa integración como elementos de diversificación y mejoramiento de los entornos de aprendizaje, ha exigido, a su vez, un replanteamiento del proceso de enseñanza-aprendizaje y de las relaciones entre los miembros de la sociedad y el entorno.

La rápida convergencia de los medios está rompiendo con el esquema tradicional de formación, que se basaba en una directa retroalimentación de los padres hacia los hijos, actualmente la base fundamental es ésta, sin embargo, la tecnología viene a complementar la formación del ser humano como tal.

Con los avances tecnológicos que surgen y que traen como consecuencia una revolución en la interacción entre los miembros de la sociedad; ejemplos tales como el Internet, el correo electrónico, la videoconferencia etc., benefician la construcción de ambientes de formación, logrando que no solamente se rompa con la visión que tradicionalmente se tiene acerca de la comunicación de masas, en cuanto al acceso a grandes grupos poblacionales, sino que además posibilita la incorporación de múltiples puntos de vista y quienes son informados se convierten a su vez en informadores.

Por otro lado, la tecnología debe tomar en cuenta una sociología de las comunicaciones ya que necesariamente se involucra el uso de los medios de comunicación con la adaptación a la sociedad, ya que ésta ve a la tecnología como una herramienta para alcanzar sus objetivos, por ejemplo, un investigador asignado a un proyecto cuyo tema es del ambiente local, se verá enriquecido al utilizar la Internet y otras fuentes tecnológicas, como el correo electrónico.

Existe una clasificación de acuerdo a la función que desempeñan las nuevas tecnologías en la formación de los miembros de una sociedad; como ayudas instructivas que no

facilitan la interacción, son lineales y se utilizan para promover la eficacia de los mensajes sociales. Se usan para la enseñanza colectiva (ejemplos de estos son: El cine, el video, las grabaciones y los materiales multimedia); y los sistemas instruccionales, que son interactivos, se usan para la enseñanza individual (ejemplos son: Los sistemas interactivos, hipertextos y entornos virtuales).

Los videojuegos como herramienta de formación social cumplen con estas características de forma no excluyente, ya que abarcan de una manera combinada las dos clasificaciones, dependiendo de las necesidades de enseñanza, es decir, pueden utilizarse como ayuda o como sistema de instrucción.

Retomando la idea final del punto anterior, donde se menciona que los videojuegos son y deben ser parte fundamental para el proceso de educación de una sociedad. Se llevó a cabo una investigación acerca de la forma en que la sociedad sabe y entiende el videojuego para que de esta forma se tenga la información necesaria que sustentará los puntos tratados más adelante.

La siguiente investigación, así como los datos que la conforman, fueron tomados de un estudio realizado por el *Instituto Latinoamericano de Comunicación Educativa* (ILCE).

DATOS GENERALES

El tipo de muestreo fue no probabilístico ni intencional por cuotas, debido a que no se empleó el aleatorio, y la selección de los sujetos debió cumplir con ciertas características asignadas de antemano, es decir, un cierto grado de escolaridad y/o conocimiento de computación, así como haber establecido un número especifico de participantes. La muestra estuvo compuesta por 172 personas (94 hombres y 78 mujeres) y fue realizada en puntos estratégicos de las 16 delegaciones del Distrito Federal; las personas participantes fueron seleccionadas por contar con experiencia en el manejo de videojuegos ya sea en computadora o en las llamadas *maquinitas*.

Se elaboró un cuestionario en donde se contemplaban ambientes de aprendizaje. La finalidad de cada una de las preguntas fue recabar información que permitiera conocer las preferencias a fin de obtener datos útiles para el capítulo concerniente a la etapa de análisis. Las preguntas que conformaron el cuestionario fueron:

1.- Menciona en orden de preferencia los 3 videojuegos que más te gustan.

2.- ¿Qué es lo que te parece más atractivo de cada uno de ellos?

3.- Describe detalladamente las formas en que se presentan los contenidos.

4.- Describe la dinámica de los juegos.

5.- Describe detalladamente los personajes de los videojuegos.

6.- Menciona los 3 videojuegos que menos te gustan.

Debido a la naturaleza de este estudio, en el cual se trataba de recoger el punto de

vista del sujeto encuestado, el análisis de la información fue muy sencillo. Se conformó una base de datos por cada pregunta, en el caso de los encuestados se diferenció según el sexo y se reunió la información de cada uno de los cuestionarios.

RESULTADOS

En total las personas del sexo masculino mencionaron 180 videojuegos diferentes, mientras que las del femenino reportaron conocer 129.

Algunos de los juegos eran de conocimiento común entre ellos, como se verá más adelante.

Se establecieron las siguientes categorías, de acuerdo al tipo de videojuego señalado:

1.- Luchas o peleas.- Combate cuerpo a cuerpo, en donde la estrategia principal combina golpes, agilidad y fuerza de los personajes.

2.- Carreras.- De autos, motos, aviones, naves espaciales, etc.

3.- Estrategia simple.- Videojuegos que hacen pensar y decidir sobre diferentes opciones con la finalidad de acumular puntos para ganar. No contienen ambientes gráficos y estrategias de juego complejas o sofisticadas.

4.- Infantiles.- Juegos cuyos personajes o ambientes se dirigen a la población infantil menor de 10 años, además de contar con una estrategia simple.

5.- Misiones y aventuras.- Con ambientes gráficos y estrategias de juego complejos y cuya temática se relaciona con vencer al enemigo utilizando armas de fuego. Además de cierta historia de interés, pudiendo o no contar con batallas, lucha cara a cara, estrategias militares, etc.

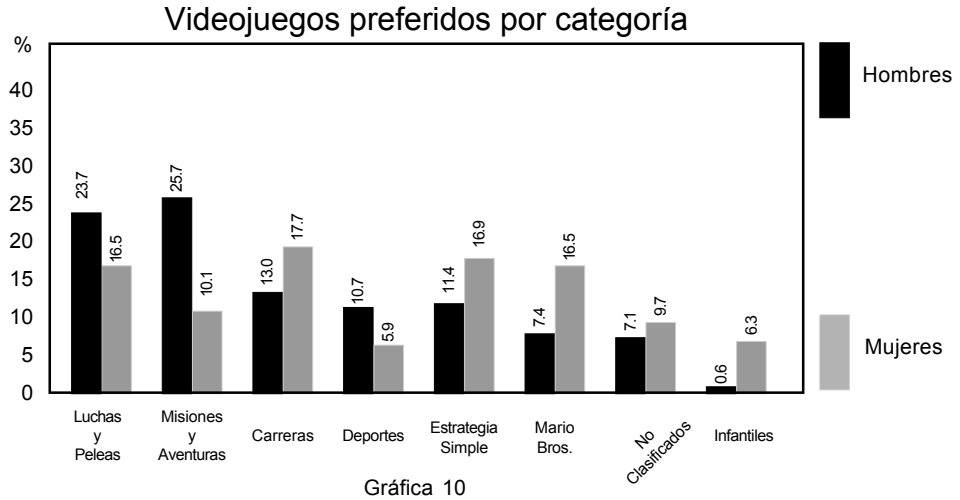
6.- Deportes.- Competencias tales como fútbol, fútbol americano, béisbol, etc.

7.- Serie Mario Bros.- Dada la alta frecuencia del uso de los videojuegos relacionados con esta serie como *Mario Kar, Dr. Mario, Mario Bros, Mario Científico, Mario Party,* etc., fueron ubicados dentro de una sola categoría.

8.- No clasificados.- Cuando las personas no especificaron claramente el contenido o temática de los videojuegos.

Al organizar las respuestas de acuerdo a estas categorías, la frecuencia total de videojuegos preferidos, muchos de ellos mencionados más de una vez, fue de 307 para el sexo masculino: Luchas y peleas, 73; Misiones y aventuras,79; Carreras, 40; Deportes, 33; Estrategia simple, 35; Mario Bros, 23; No clasificados, 22 e Infantiles, 2. Mientras que para el sexo femenino fue de 236: Luchas y peleas, 39; Misiones y aventuras, 24; Carreras, 42; Deportes,14; Estrategia simple, 40; Mario Bros, 39; No clasificados, 23 e Infantiles,15 como podemos apreciar en la gráfica 10.

Los porcentajes de respuesta: n = 94 hombres / n = 78 mujeres. Ver la siguiente gráfica (10)



Gráfica 10

Se observó que los juegos favoritos de los hombres son los de Misiones y aventuras con el 84% y Luchas y peleas con el 78%. Mientras que las mujeres distribuyeron sus preferencias con mayor amplitud: Carreras 54%, Estrategia simple 51 %, Luchas y peleas 50% y la serie Mario Bros 50%.

De acuerdo con estos resultados, vemos que es en Luchas y peleas donde coinciden las preferencias de hombres y mujeres, sin embargo, si tomamos en cuenta la similitud de las barras, tenemos los juegos de Carreras y Estrategia simple como preferencias comunes entre ambos.

Es necesario enfatizar la importancia que tienen los gustos de los hombres ya que se concentran principalmente en dos tipos de videojuegos, mientras que las mujeres parecen ser más eclécticas en su selección, esto pudiera significar que existe una preferencia diferenciada hacia los videojuegos entre hombres y mujeres.

Entre las razones del porqué las mujeres prefieren los videojuegos de Luchas y peleas, ellas indicaron que era debido a que su contenido no era sangriento, además de que muchas veces el personaje principal era mujer, situación que le otorga un grado de identificación de género, mientras que los juegos de Estrategia simple les parecen atractivos debido a que no presentan demasiadas dificultades para su uso, sin embargo, ambos géneros criticaron, en ocasiones duramente, los videojuegos por diversas razones.

En el caso de los hombres, los argumentos fueron:

Luchas y peleas.- *Tienen poca duración; los personajes son espantosos; los retos son difíciles; la temática es simple y las estrategias son sencillas. No se pueden escoger a los personajes y algunos otros suelen ser poco reales y hay mucha violencia.* Ejemplos de estos juegos son: *Mortal Kombat, Dragon Ball, Doom, Killer Instinct,* etc.

Estrategia simple.- *Son tontos, el tema es simple y no presenta gran dificultad porque no tienen retos, las gráficas son malas y el objetivo del juego es aburrido; algunos de*

ellos son muy infantiles y otros desesperan y presentan poca acción. Algunos ejemplos son: *Pokemon, Tetris, Solitario, Pac-Man, Carta Blanca, Prince, Simpsons,* etc.

Sobre el resto de los juegos en general indicaron que no son muy divertidos, les falta dinámica y son muy simples o tienen retos demasiado fáciles *(Serie Mario Bros, FIFA, Rugrats, Resident Evil, Frog, Mulan, etc.)*

Por otro lado, y con relación al por qué las mujeres no prefieren algunos juegos de video, comentaron que les parecen aburridos y tontos; que son demasiado sangrientos; que son muy violentos o que son muy difíciles de usar y se aburren pronto.

En primera instancia, las respuestas hasta aquí vertidas nos arrojan información importante para la construcción de un ambiente de aprendizaje computarizado que incluya juegos didácticos. A simple vista, una de las características para diseñar un videojuego para mujeres es que no sea demasiado sangriento y que no implique altos niveles de violencia; que contenga uno o varios personajes femeninos y que las estrategias sean fáciles de operar. Mientras que para los hombres, los juegos no deben ser demasiado simples, aburridos o infantiles, tienen que ser divertidos y con la posibilidad de configurar los personajes que se utilizan al jugarlos.

LO MÁS ATRACTIVO DE LOS VIDEOJUEGOS

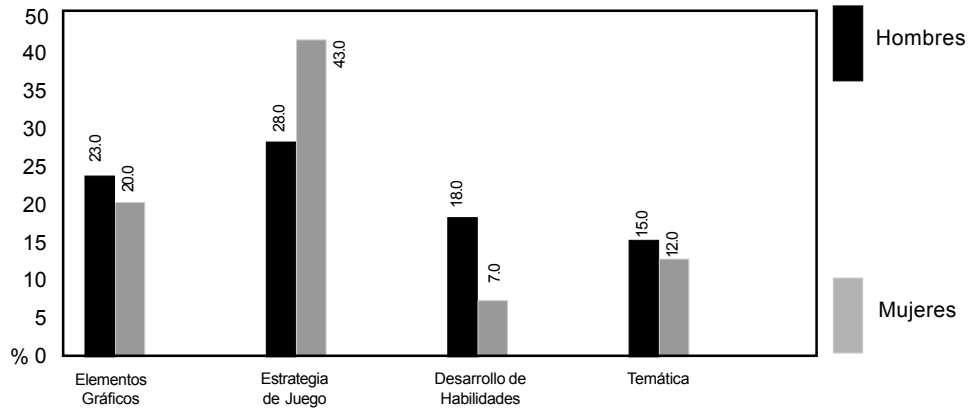
Para el análisis de lo más atractivo de los videojuegos, la información se dividió en las siguientes categorías: Elementos gráficos, elementos sonoros, estrategia del juego, características de los personajes, desarrollo de habilidades, interactividad y temática.

En total, los hombres mencionaron 394 particularidades que les parecen atractivas en los videojuegos: Elementos gráficos, 89; elementos sonoros, 19; estrategia del juego, 112; características de los personajes, 32; desarrollo de habilidades, 69; interactividad, 15 y temática, 58.

Mientras que las mujeres mencionaron 271 atributos atractivos de los juegos: Elementos gráficos, 55; elementos sonoros, 13; estrategia del juego, 117; características de los personajes, 31; desarrollo de habilidades, 18; interactividad, 4 y temática, 33.

En la gráfica 11 se presentan los porcentajes de respuesta para cada una de las categorías antes señaladas. n = 394 hombres / n = 271 mujeres.

Atracción de los videojuegos



Gráfica 11

De acuerdo a la opinión de ambos, la categoría cuyos elementos les parece más atractiva es la estrategia del juego u oportunidades de juego y sus herramientas. Dentro de esta categoría los hombres y mujeres mencionaron elementos como: Que el videojuego tenga trucos; que sea interactivo; con herramientas para pelear; que tenga variedad de juegos; con diferentes estrategias militares; con diversos competidores en red; con posibilidad de jugar en equipo; que contengan niveles; diversas formas de pelear y dificultad de niveles, etc. Es notorio que las mujeres se hayan referido a esta categoría en mayor medida que los hombres.

Le sigue en importancia la categoría elementos gráficos, ambos grupos opinan que las imágenes, los escenarios, los paisajes, los efectos, los colores utilizados, las imágenes tridimensionales, el realismo de las imágenes, etc., son también atractivos. Se debe observar que, aunque la diferencia es pequeña, un mayor número de hombres le dio importancia a este punto.

Por otra parte, los hombres también consideran que son atractivos los elementos en las categorías de desarrollo de habilidades (te hace pensar, agiliza tus reflejos, etc.) y a la temática (que es de peleas, deportes, misterio, guerra o terror, viajes, carreras, etc.)

Puede observarse que estas dos categorías también son más importantes para los hombres que para las mujeres.

En el caso de la categoría de desarrollo de habilidades, además de las diferencias en el número de menciones, también existen discrepancias en el tipo de habilidades que cada grupo percibe, por ejemplo, mientras que los hombres dicen que: *«...te hace pensar, se adquiere destreza y agiliza tus reflejos...» las mujeres opinan que: «...aprendes a manejar el joystick y el teclado, adquieres velocidad, se aprenden golpes y agiliza mi modo de pensar...»*

Finalmente, un menor número de hombres y mujeres le dio importancia a las categorías de caracterización de los personajes, efectos sonoros e interactividad. A continuación se presentan algunas de sus respuestas:

Caracterización de los personajes.- Al respecto los hombres dijeron: *«...me gusta el vestuario, que los personajes se transformen, la variedad de jugadores, etc...»* Mientras que las mujeres se refirieron a su simpatía, al físico, al porte de la muchacha, la rapidez con que se mueve y a sus habilidades físicas.

Efectos sonoros.- Hombres y mujeres generalizaron en la música: Que tiene buen sonido; con detalles de sonido; sonido real; sonido innovador y fuerte y con efectos de sonido.

Interactividad.- En este caso opinaron principalmente los hombres e indicaron que les atraía la facilidad de movilidad; el manejo de los personajes; la facilidad en el manejo de autos, así como la maniobrabilidad de las naves, etc.

FORMA DE PRESENTACIÓN EN CONTENIDOS

Para el análisis de la pregunta referente a la manera como se presentan los contenidos de los programas, las respuestas se agruparon en cuatro categorías:

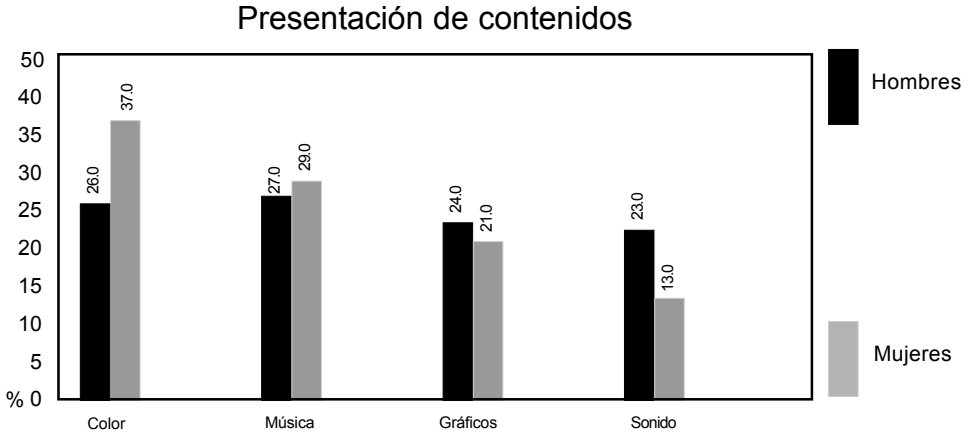
1.- Gráficos.- Se incluye todo lo referente a la escenografía de los juegos y el diseño de los personajes, así como a los detalles especiales de los mismos.

2.- Colores.- Manejo y aplicación de los colores según los juegos.

3.- Sonido.- Todo tipo de sonidos para ambientar los juegos.

4.- Música.- Qué tipo de música se presenta y cómo es utilizada en cada juego.

En la gráfica 12 se observa que ambos presentan diferencias en cuanto a la forma que perciben la presentación del contenido. Por un lado los hombres le dan más importancia a la música, al color, a los gráficos y al sonido, en ese orden, mientras que las mujeres se enfocan más hacia el color, seguidos por la música, los gráficos y el sonido. Asimismo las categorías con porcentajes más elevados son las de color y música, con un promedio mayor de mujeres en la primera y uno mayor de hombres en la segunda.



Gráfica 12

Respecto a los colores, los hombres (77%) reportaron que son vistosos y atractivos, reales y alegres, e indicaron que son acordes al escenario, a la temática y al ambiente de cada juego, es decir, si el contexto es de misterio o de noche, los colores son oscuros, si es de día son alegres y vistosos. Las mujeres (85%) dicen que los colores son vivos, bonitos, variados, nítidos, atraen la atención, son realistas y bien manejados, se pueden configurar, etc.

De la música los hombres (85%) dijeron que tiene la función principal de ambientar la temática del juego, mientras que las mujeres (67%) indicaron que es bonita, alegre, prendida, varía con la acción, tiene cambios, se puede regular el volumen, crea el ambiente adecuado, te relaja, te pone alerta, etc.

En relación a los gráficos, los hombres (74%) hablaron de aspectos conexos con los escenarios, su diversidad, el realismo, la calidad de las imágenes, la definición y detalles y el tipo de colores. Por su parte las mujeres (49%) mencionaron que los personajes están

bien hechos, crean el ambiente requerido, hacen buen uso de videos y gráficos, usan 3D, etc. Dentro de esta misma categoría los hombres y mujeres se refirieron a los personajes, generalmente les gusta que estén bien diseñados, que tengan apariencia real, con cuerpos bien elaborados y detallados, así como la posibilidad de opciones.

En cuanto a los sonidos, los hombres (64%) dicen que se escuchan claros y reales, muy acordes con el momento de la acción, es decir, esperan que haya sincronía o concordancia entre los elementos gráficos y el sonido, por ejemplo que se escuche el disparo de una pistola al ser accionada; el grito del personaje al recibir un golpe o realizar algún festejo, etc. Para las mujeres (31%) los efectos de sonido deben corresponder a la realidad, que se escuche lo que haces, que sean fuertes, rápidos, agradables, diferentes, que ayuden a la comprensión de la acción y creen el ambiente requerido, etc.

DESCRIPCIÓN DE LA DINÁMICA DEL JUEGO

Con esta pregunta los encuestados describieron cómo se juegan los videojuegos que conocen. De acuerdo con sus respuestas se pudieron identificar las 3 categorías siguientes: Personalización del juego, sistema de recompensas y castigos y estrategia del juego.

A continuación se describen los elementos que componen cada una de ellas, así como algunas respuestas:

1.- Personalización del juego.- Se refiere a la posibilidad que se tiene de elegir las características de los personajes, los ambientes, los niveles de dificultad, las herramientas, etc. Algunas respuestas fueron las siguientes:

Hombres: Hay diferentes niveles de pelea; se escogen los personajes; se seleccionan las ligas para que juegues a ganar la copa del mundo; estos juegos tienen niveles de dificultad según la experiencia, etc.

Mujeres: Se elige juego y el lugar a donde se quiere ir; se elige tipo de vehículo, velocidad, frenos y combustible; se pueden determinar jugadores y contrincantes; requieren niveles distintos de concentración, etc.

2.- Sistema de recompensas y castigos.- De acuerdo con los hombres y las mujeres la mayoría de los videojuegos operan con sistemas de recompensas y castigos. El sistema de recompensas implica que el jugador obtiene un premio (puntos, avance de nivel, poderes, etc.) cada vez que realiza correctamente una acción determinada. (Vence enemigos, resuelve acertijos, gana la carrera, etc.)

Las respuestas de hombres y mujeres fueron: Vas pasando de nivel conforme se gana un partido; se necesita mucho coco para resolver acertijos; cuando resuelves eso te dan llaves o algún instrumento para abrir puertas, etc. Se va pasando a etapas según se vaya ganando; se va pasando de nivel conforme encuentras las llaves y la salida; hay que eliminar enemigos para avanzar; se avanza de niveles matando adversarios; pueden existir misiones que dan puntaje para avanzar; se avanza mediante la destrucción de otros; el personaje golpea y gana puntos (comida). Tocar un tambor para ganar vidas o diamantes y pasar de nivel y al vencer un contrincante se cambia de país, etc.

Por otra parte, el sistema de castigos implica que el usuario perderá lo que ha ganado o en algunos casos puede retroceder de nivel o ser enviado al inicio del juego si no

realiza correctamente lo que se espera de él o ella. En muchos de los videojuegos donde existen personajes, este sistema tiene relación con la capacidad que tienen para sobrevivir y defenderse de sus enemigos.

Algunas de las respuestas al respecto fueron: Se trata de pasar misiones sin que te maten; debes andar con cuidado por que si te ven las cámaras o ellos estas en peligro porque si te matan vuelves a empezar; es llegar a la meta sin caer; es difícil porque hay tráfico y te hacen *chocar*; hay que cuidarse de los competidores pues también te hacen chocar. El reto es mantenerte dentro de la pista sin volcarte; hay que cuidarse de no chocar con la patrulla; sólo cuidar que no te toquen los enemigos; es hacer puntos evadiendo alguna mina, si no, pierdes; cuidarse de trampas y otros personajes; existen obstáculos como curvas, tipo de auto, etc.

Por otro lado, cuando los hombres respondieron a esta pregunta manifestaron que les desagradan algunos juegos que tienen personajes demasiado vulnerables, por ejemplo, cuando *mueren* con sólo tres golpes o tienen pocas municiones o demasiados enemigos, etc.

3.- Estrategias de juego.- Esta categoría tiene relación con la actividad concreta que deben realizar para poder usar el videojuego. Existe una gran variedad de estrategias que dependen de muchos factores como el tipo de juego, la temática, la finalidad o meta (su objetivo puede ser didáctico, de esparcimiento, para desarrollar habilidades manuales o mentales, etc.) las características del usuario como la edad, las habilidades manuales, algunas veces el sexo, los conocimientos previos, etc.

A continuación se presentan las categorías y algunas de las respuestas de los encuestados que permiten describir con mayor precisión cada una de las estrategias:

a) Luchas y peleas.- Básicamente la estrategia consiste en la adquisición de habilidades en la lucha cuerpo a cuerpo. El usuario emplea el teclado, joystick o botones de consola para que él o los personajes puedan luchar contra sus contrincantes. Estos son algunos de los ejemplos de este tipo de juegos de acuerdo con las respuestas: Se pelea con varios equipos; se hacen combinaciones de patadas, son peleas callejeras, se tienen que hacer movimientos rápidos; se escogen dúos de personajes para pelear con otro dúo; la estrategia consiste en hacer combinaciones de golpes; se tienen que aprender trucos de pelea; debes ser muy rápido para derrotar a tu adversario; se salva a una mujer, para ello se pasan varios niveles matando soldados, etc.

b) Misiones y aventuras.- En este caso la estrategia trata de combates o luchas con la particularidad de que simultáneamente cuenta con una historia o trama que algunas veces se traduce en una misión. La interfase gráfica es muy sofisticada.

En este tipo de estrategias se combina el desarrollo de habilidades manuales (uso de teclados, consolas, etc.) con mentales (razonamiento lógico para resolver acertijos o situaciones). Las siguientes son algunas de las descripciones hechas por los encuestados: Terminar con todos los zombis de la ciudad; la estrategia es no acercarte demasiado a ellos para que no te maten; se necesita mucho coco para resolver los acertijos y claves.

Este juego no es 100% de aventura, está combinado con acción.

Un joven tiene que rescatar su reino para que llegue a la normalidad; se trata de avanzar en las diferentes edades de la humanidad; se trata de rescatar un juguete robado; se cambia de armas para poder ganar y explorar cada ambiente de juego; se juega conociendo personajes y recogiendo pistas de la historia; tiene que encontrar llaves y armas para aclarar asesinatos o matar enemigos; las estrategias consisten en usar el arma adecuada, etc.

c) Carreras.- Aquí se agruparon los videojuegos cuya estrategia está relacionada con la simulación de competencias entre autos, motos, aviones o naves espaciales. En este caso los encuestados utilizan también las interfaces de la computadora o la consola para movilizar los vehículos y las habilidades que desarrollan que generalmente son de tipo manual. Los encuestados dijeron lo siguiente: Es una competencia de autos deportivos en donde las pistas son de diferentes ciudades y países, el reto es ganar la competencia; es necesario ir ganando carreras con los carros que elijas o compres; es una carrera de autos donde puedes hacer explotar al contrincante; como la carrera es contra tiempo tengo que ser muy rápido; se compite con otros autos o contra tiempo, etc.

d) Deportes.- Como su nombre lo indica, esta categoría incluye estrategias donde los encuestados participan en competencias deportivas como fútbol, básquetbol, béisbol, etc. Algunas resapuestas fueron: Hay que tener un balón; seleccionar las ligas para que juegues a ganar la copa del mundo; se debe mover bien el balón, se hacen alineaciones, si ganas te vas enfrentando a equipos más fuertes; debes batear la pelota que te lanza el contrincante; es muy fácil, sólo hay que ganar los partidos, etc.

e) Estrategia simple.- En esta categoría se agruparon los videojuegos que están incluidos en la mayoría de los sistemas Windows *(Tetris, Buscaminas, Solitario)* y algunos otros que no contienen ambientes gráficos o estrategias de juego sofisticadas *(Pac Man, Tiro al Blanco, Los Simpsons, etc.)* Se denominaron de estrategia simple ya que los encuestados deben desarrollar casi exclusivamente habilidades más lógicas o mentales, es decir, una vez que se aprende la mecánica del juego, su actividad se limita a repetir las acciones aprendidas. Algunas de las descripciones que hacen de estos juegos son las siguientes: Esquivar fantasmas; tener habilidad para comerse las bolitas y así pasar de nivel; al inicio debes capturar un monstruo, luego éste te ayuda a capturar a otros; colecciona tarjetas y diferentes *ítems*. Los trucos los sacas apretando un triángulo y un movimiento de palanca; es hacer puntos evadiendo minas, si no, pierdes; debes buscar los números y ordenarlos; tienes que evitar que la pelota se caiga, si no, pierdes; yo recomiendo que se coman la mitad de las bolitas utilizando las de poder y así se aprovecha para comer al fantasmita que se encuentre más cerca para acumular más puntos, etc.

f) Mario Bros, Infantil y No clasificados.- Estas tres categorías comparten elementos muy similares a la anterior, es decir, no poseen elementos gráficos muy sofisticados, las estrategias de juegos son simples y requieren de habilidades manuales más que lógicas. Estos juegos fueron clasificados separadamente debido a la frecuencia de su uso entre los hombres y mujeres.

DESCRIPCIÓN DE PERSONAJES

Respecto a los personajes que conforman cada uno de los videojuegos, se detectó que estos se incluyen o no, dependiendo de las características de los juegos. En algunos de ellos el personaje principal es el propio usuario quien es colocado en un ambiente determinado e interactúa con otros personajes secundarios o elementos gráficos.

Si el juego versa sobre Luchas y peleas los personajes son hombres o mujeres jóvenes con cuerpo atlético o vestimenta de *karatecas* en su mayoría provenientes de historietas y dibujos animados. Si es de Misiones y aventuras suelen ser princesas, hadas, zombis, monstruos, soldados, gobernadores, etc., algunos de ellos con poderes especiales. En el caso de los juegos de Carreras son autos, motocicletas, aviones o naves espaciales y el usuario es quien los conduce.

Los videojuegos de Deportes están conformados por la caracterización de jugadores o equipos profesionales reales, generalmente del sexo masculino. Los juegos de Estrategia simple, Mario Bros, Infantiles y No clasificados cuentan con una gran variedad de personajes o elementos gráficos como hongos, fantasmitas, figuras geométricas, naipes, minas, pelotas, etc.

A continuación mostramos un resumen general de las características de los personales mencionadas por los encuestados.

CARACTERÍSTICAS GENERALES DE LOS PERSONAJES

1.- Sexo.- En la mayoría de los juegos el personaje que lleva el rol principal es del sexo masculino. En pocos el personaje es una mujer.

2.- Edad.- De 12 a 35 años.

3.- Características de los personajes masculinos.- Amigables, risueños, maduros, crueles, agresivos, fuertes y brutales.

*4.- Características de los personajes femeninos.-*Tiernas y en ocasiones agresivas.

5.- Dones especiales.- Poseen poderes especiales, mucha agilidad, hacen magia, tienen poderes psíquicos, etc.

6.- Características físicas de personales masculinos.- Piel blanca, pocos de piel morena. Fuertes, fornidos, atletas, altos, delgados, pelo obscuro, castaño, negro, cabello largo o corto, pelo chino, sanos, ojos azules, físico perfecto y con bigote.

7.- Características físicas de personajes femeninos.- Guapas, delgadas, con mucha agilidad, fuertes, apiñonadas, bonitas, visten ropa entallada, estatura media, cabello largo y negro.

8.- Elementos que participan en el juego.- Todo tipo de armas, cuadritos, bolitas, pelotas, cubos, fantasmitas, autos de todo tipo, motos, tablas para esquiar, patinetas, balones, aviones, naves, misiles, tanques, caballos, catapultas y barcos de guerra.

Para finalizar con la revisión estadística de los videojuegos se mencionarán brevemente algunas otras características que fueron detectadas:

1.- Están integrados mayormente por personajes de sexo masculino, aunque también aparecen algunas mujeres.

2.- Muchos de ellos poseen poderes y habilidades especiales y aparentemente su finalidad es recrear en ellos la figura del héroe.

3.- Muchas de las historias o tramas utilizadas se basan en la dicotomía del *Bien y el Mal.*

4.- Se hace uso de actividades que implican competencia entre equipos o personas en las que se enfatiza el despliegue de habilidades físicas o mentales, y

5.- En pocos casos se detectaron juegos con una finalidad estrictamente didáctica.

De acuerdo a la información hasta este momento recabada, se concluye que los videojuegos, desde el punto de vista social, están ligados al enfoque pedagógico, ya que una parte fundamental en la formación de la sociedad es la educación.

Por lo que respecta a la encuesta realizada, se observa una formación encaminada hacia una concepción de una cultura del videojuego, ya que de acuerdo a los datos arrojados, las personas tienen muy presente todos y cada uno de los elementos que conforman un videojuego.

Por parte de los desarrolladores es primordial el entender que deben de crear videojuegos con conceptos claros y basados en la ingeniería de software, de modo que cumplan con el objetivo de estos, es decir, que fomenten el correcto aprendizaje del usuario final que es la base del núcleo social.

NIVEL ECONÓMICO

Tratando de sustentar un entorno global dentro del contexto tecnológico del país, las tendencias actuales en el mundo, es que los gobiernos en lugar de atender los cambios e innovaciones, así como el tener, orientar y propiciar un ambiente adecuado para la inversión en el campo tecnológico, primordialmente a la industria de la computación, se enfoca en duplicar servicios que el sector privado ofrece.

En el caso de México, los regímenes que antecedieron a las dos ultimas administraciones (incluyendo la actual) sólo se enfocaron en contemplar los avances surgidos alrededor del mundo y no actúaron para que el auge tecnológico llegara al país.

Un reciente estudio califica a Canadá, Singapur y a Estados Unidos de Norteamérica como líderes innovadores en materia de *e-goverment*, le siguen Noruega, Australia, Finlandia, Holanda y el Reino Unido que se catalogan como *visionarios seguidores.*

En una tercera línea se encuentran Nueva Zelanda, Hong Kong, Francia, España, Irlanda, Portugal, Alemania y Bélgica; después, apenas iniciando, se ubican Japón, Brasil, Malasia, Sudáfrica, Italia y México, esto significa que el país se encuentra en una etapa de

atención y entendimiento para hacer bien las cosas, sin embargo aunque esta encuesta nos pone en el camino, no significa nada si consideramos que los países líderes no llegan ni a la mitad de lo proyectado en ese rubro. Es conveniente aclarar un par de elementos: Primero, que en los países mencionados en la encuesta el desarrollo del proyecto *e-goverment* ha sido recibido con entusiasmo salvo cuando el gobierno se excede en sus funciones y compite con la iniciativa privada y, segundo, que en ninguna de estas naciones hay tan pocos usuarios de Internet como en México, donde aproximadamente un 3% de la población esta conectada.

Desgraciadamente la visión del gobierno considera que este bajo porcentaje se debe a un mal trabajo realizado por los centros de acceso a la Internet de la iniciativa privada (ISP) y considera que podrán hacerlo mejor, por lo tanto, comenzará a ofrecer el servicio donde existían estaciones de telégrafos y desde los municipios.

Cuestionamos, ¿Se debe creer que no hay más población conectada porque la iniciativa privada no sabe vender y el gobierno si? Cuando éste no hace su trabajo de modo que el problema no sea de ingreso per capita, cultural y de infraestructura.

Como es sabido, la administración gubernamental pasada tomó conciencia y tuvo que enfrentarse al problema de conversión informática del año 2000, sin embargo, aunque se libró con éxito, el gobierno descubrió y se enfrentó a un desarrollo informático sin directriz la cual propicia una ineficiente automatización de los servicios públicos, así como una insuficiencia de infraestructura y equipamiento.

En el gobierno actual, el Ejecutivo adiciona como una de sus funciones sustantivas la dirección de las acciones generales de carácter informático que conlleven a la modernización de los procesos y servicios públicos, así como la coordinación de los proyectos interinstitucionales en la materia y la evaluación permanente de la calidad de los niveles de automatización. A su vez, el gobierno busca fomentar la inversión nacional y extranjera para la apertura y formación de nuevas empresas en el ramo tecnológico informático mediante un marco normativo de incentivación, aunado con un entorno de seguridad en todos los ámbitos.

Para fomentar esta función primordial, el gobierno lanzó el proyecto *e-México,* el cual se sustenta en la cultura y la civilidad de la sociedad, logrando impulsar el arte y la creatividad. Dentro de los objetivos generales del programa es el posicionar en un lugar privilegiado al país en el campo de la tecnología informática, así como el reorientar los contenidos en el contexto de entretenimiento informático (videojuegos por ejemplo) y en el educativo; el implantar servicios gubernamentales automatizados y de información estricta y cualitativamente elaborados; el promocionar el idioma castellano en la red de comunicaciones Internet estableciendo un acuerdo con todos los países hispanohablantes; la regulación adecuada de los identificadores telemáticos nacionales: Correo electrónico, nombres de dominio y numeración en telecomunicaciones sobre protocolos IP, etc.

El programa *e-México* se caracteriza por contar con dos directrices de trabajo bien definidas en el campo público y en el privado.

Las aplicaciones y servicios medulares en el público son el comercio electrónico, la educación a distancia, la telemedicina y el teletrabajo, claro está, estos servicios deben tener el contenido como ingrediente básico y los multimedios como toque de calidad.

También se busca la implementación de servicios electrónicos gubernamentales

que trabajen bajo la confianza, seguridad e integridad en el uso de la información que interactúa entre la sociedad y la autoridad.

Las áreas de oportunidad son:

1.- Información:

a) De interés público.
b) Como mecanismo de rendición de cuentas a la sociedad.
c) Para mejorar la atención a la ciudadanía por concepto de servicios.

2.- Trámites:

a) Ciudadanos.
b) Empresariales.
c) Con otras instituciones gubernamentales.
d) Al interior de las instituciones.
e) Con instancias de otros órdenes de gobierno.

3.- Servicios:

a) Educación y capacitación.
b) Salud.
c) Seguridad pública y social.
d) Protección civil.
e) Urbanos.
f) Turísticos.

Los beneficios que aporta el proyecto *e-México* son, de inicio, que no se requiere presencia física en el lugar del tramite; proporcionar servicios a distancia; evitar la corrupción mediante requisitos ya predefinidos; disminuir la interacción con servidores públicos, evitando discrecionalidad; ahorros en tiempo y costos; cualquier terminal-usuario se convierte en ventanilla y lo más importante la transparencia de la acción gubernamental.

Como líneas de trabajo se contempla elaborar una planeación tecnológica ligada a proyectos gubernamentales prioritarios para la atención ciudadana, canalizar esfuerzos y recursos a proyectos que tengan efecto multiplicador, un análisis con objeto de desregular y ajustar procedimientos para la simplificación de trámites y prestación de servicios, la fomentación de una cultura informática a través de la capacitación y, establecer un marco jurídico seguro.

Como conclusión para el análisis del proyecto *e-México* en el campo público es, que la situación actual de las infraestructuras de tecnología de la información muestra evidentes debilidades, así como un atraso en sus niveles de automatización, esto como resultado de la carencia de estrategias y lineamientos claros y concretos para el impulso de la informática como medio para la optimización de los servicios públicos, así como la inexistencia de un organismo rector como sustento del desarrollo de las soluciones informáticas.

El presente gobierno se propone impulsar la transformación de la administración pública federal en organizaciones modernas y competitivas, lo cual, ante la dinámica tecnológica actual y la creciente demanda por parte de los usuarios y prestadores de servi-

cios públicos por hacer más eficiente y transparente la gestión pública, exige vincular los procesos de transformación administrativa con el uso de la tecnología.

Lo anterior implica, por una parte, diseñar, promover, implantar y evaluar estrategias orientadas a elevar los niveles de automatización de los procesos internos y aquellos en los que interactúan particulares y autoridades y, por la otra, establecer un marco normativo acorde a la realidad, un marco que, por citar algunos ejemplos, permita el uso de medios de comunicación electrónica a los particulares para presentar solicitudes, que las notificaciones de las dependencias a particulares se envíen por medios electrónicos, que permita utilizar medios de identificación electrónica en sustitución de firma autógrafa con el mismo valor probatorio, en fin, emitir reglas de carácter general que logren mejorar la normatividad actual en materia pública dando su real valor al uso de la tecnología, de modo que el gobierno pueda proporcionar a la sociedad un beneficio.

Por lo que respecta al campo privado, el gobierno federal, a través de las instancias correspondientes y como línea de acción del proyecto *e-México,* busca propiciar un entorno que favorezca la inversión para la apertura o expansión de empresas enfocadas a la tecnología. Este proyecto tiene como finalidad el generar políticas y mecanismos a fin de explotar la Internet (hacer realidad el acceso a toda la sociedad); la apertura y ubicación en el territorio nacional de más empresas dedicadas a la producción de sistemas multimedios y de contenidos (videojuegos por ejemplo); fomentar en las industrias la cultura tecnológica; el concebir mecanismos de financiamiento e incentivación de inversiones; la formación de recursos humanos altamente capacitados con objetivos muy precisos a través de instituciones universitarias públicas y privadas y, finalmente, la promoción permanente y más enérgica de una cultura nacional de información.

Cabe mencionar que México ofrece grandes oportunidades desde el punto de vista comercial, por ejemplo: Una posición geográfica privilegiada en un mundo globalizado, un gran mercado interno, mano de obra relativamente más barata que otros países, etc.

El gobierno federal busca captar las inversiones particulares mediante un programa de incentivación tributaria que se refleje en una disminución escalonada en materia de impuestos; requerimientos relativamente bajos de inversión; fomentar una vocación exportadora de productos de calidad; establecer, en conjunto con la banca, mecanismos de financiamiento; un combate frente a la inseguridad y el delito, principalmente a la piratería; etc. También busca establecer un marco normativo que de facilidades para la inversión, dentro de las reformas pueden destacar: Establecer nuevas modalidades de trabajo a través de la utilización de medios electrónicos; una legislación en materia de propiedad intelectual, industrial y de comercio electrónico; una libertad de expresión e imprenta en los medios electrónicos y, finalmente, una adecuada construcción de infraestructura nacional en telecomunicaciones.

De acuerdo a la información hasta este momento recopilada, el gobierno busca definir lineamientos claros y propiciar un ambiente de inversión en el campo de la tecnología.

A continuación se hace un estudio del pasado reciente en cifras y datos directamente relacionados con la industria del videojuego con el objetivo de dar a conocer la viabilidad y el riesgo que representa el iniciar una empresa del ramo desde el punto de vista económico.

Como antecedente, debemos reconocer que el gran problema de la rama de tecnolo-

gía, principalmente en la industria del software, es la *piratería*, por citar, México se encuentra entre los primeros países del mundo con mayor índice de programas con copias ilegales.

Hay que aclarar que las cifras citadas a continuación, por concepto de videojuegos, salvo algunas excepciones y que servirán de referencia, son datos dados a conocer por empresas en el extranjero ya que desgraciadamente en México sólo existen tres compañías dedicadas a la industria del videojuego, por lo tanto no nos permite tener datos globales que sirvan de parámetro sobre el comportamiento esta industria en el mercado nacional. Los principales países productores de videojuegos son: Estados Unidos, Canadá, Japón, Francia, Alemania, Inglaterra y España.

Ahora bien, México no es el único país con problemas de *piratería*, según la *Interactive Digital Software Asociation* (IDSA), las pérdidas que alcanzó la industria a causa de este delito durante el año 2000 a nivel mundial, fue de 3.2 billones de dólares. Suma que representa prácticamente la mitad de los ingresos por ventas durante 1999. De acuerdo a la IDSA es preocupante que países como Tailandia donde el 93% del software que se vende es totalmente *pirata* y en el caso de México este fenómeno bloquea el desarrollo de la industria ya que mueve el 90% del mercado (en Estados Unidos un videojuego vende un mínimo de 50 mil copias, mientras que en México se venden 10 mil).

Ahora bien, este problema no es privativo de los países en vías de desarrollo, por ejemplo, el 65% de los títulos que se compraron en España, el año pasado, tuvo un origen ilegal, según cálculos del sector, frente a los 9 millones de juegos que se vendieron por causes legales, unos 20 millones fueron *piratas*; a su vez, se asegura que este ilícito trae pérdidas por 167 millones de dólares anuales, además actúa como freno para el desarrollo de la industria en ese país.

Viendo la *piratería* como un problema social, la parte más importante y que fomenta este fenómeno es la misma sociedad. El primer estudio sociológico al respecto realizado en México revela que los consumidores ven con buenos ojos el uso de copias ilegales, es así que el 75% de los videojuegos que llegaron al mercado el año pasado fueron ilegales. Además el 45% de la población usuaria mayor de 30 años admite que cuenta con copias de este tipo en su casa. Las razones para justificar esta práctica, tipificada como delito en el Código Penal, van desde el ahorro que supone para la economía doméstica hasta la percepción de que la industria del videojuego ya gana suficiente dinero.

En México se iniciaron los videojuegos con la aparición en el mercado del sistema Atari a fines de 1979 junto con cartuchos como: *Pacman, Phoenix, Popeye,* etc.

Los videojuegos proliferaron a través de máquinas de videojuegos o *chispas*, en 1985 se aumentó en forma desmesurada su producción (en agosto de ese año había cerca de 700 aparatos en el Distrito Federal y para el cierre de actividades del mes de febrero del año siguiente eran ya 4,000 lo que representó un aumento del 500% en sólo seis meses). Hasta hace un par de años, México era importador de programas de videojuegos, actualmente, a pesar de no tener una línea para revertir el mercado importador, existen tres empresas 100% nacionales que surgen como primer intento en la producción de videojuegos.

Aztec-Tech desarrolló el primer juego con sello Hecho en México llamado *Worldmasters*. El producto es un juego que reproduce distintas estrategias de guerra en

la historia de la humanidad, empezando con los cavernícolas pasando por los romanos, el viejo oeste, las guerras mundiales, Vietnam y el golfo Pérsico, para concluir con una batalla futurista en la Luna. El *Worldmasters* comenzó a trabajarse hace tres años y hasta la fecha ha requerido casi un millón de pesos de inversión; el producto se lanzará en diciembre, sin embargo, como medida de protección, *Aztec-Tech* busca ofrecerlo, aún en etapa de preparación, al mercado estadounidense a través de una distribuidora o *publisher* con una inversión de un millón de dólares.

Radical Studios es otra empresa mexicana fundada hace seis meses, que ha logrado una versión demofuncional de un juego de combate llamado *Starfare* y está desarrollando un juego de fantasía. La empresa considera como opción para la entrada al mercado internacional la maquila de juegos sencillos.

La Asociación Mexicana de Videojuegos (AMV) indica que la *piratería* en México ha desanimado a los desarrolladores transnacionales y a los inversionistas locales para establecer plantas de producción, a ello hay que sumar que los profesores de programación computacional no abordan el tema de los programas interactivos de carácter lúdico, que podría ser una opción profesional a futuro.

El principal objetivo de la AMV, que está a punto de constituirse legalmente con la asesoría de la *Interactive Digital Software Asociation* de Estados Unidos, es fomentar una cultura de esta opción de entretenimiento, dado que su desarrollo implica el trabajo de un gran equipo de profesionales. *«...Queremos descubrir talentos mexicanos para ponerlos en contacto con los productores que se concentran en Estados Unidos y Japón y, más adelante, lograr que estas empresas establezcan en México sus filiales...»* **(3)**

En opinión del Presidente de la AMV las posibilidades interactivas de los videojuegos, así como el trabajo creativo que implican, podrían convertirlos en un medio de expresión artística, al grado que podría transformarse el videojuego en el octavo arte.

Desde el punto de vista del recurso humano, para desarrollar un videojuego se forman equipos de hasta 60 personas que se ocupan no sólo de la programación, sino de crear la historia y delinear los personajes, desarrollar el guión, componer la música y crear una atmósfera adecuada. Es así que en un equipo de desarrollo se pueden distinguir áreas especificas y por consiguiente personal especializado como: Diseñadores gráficos, modeladores, animadores, programadores, escritores, músicos, compositores, etc. Pero no sólo el equipo de desarrollo interviene en la creación de los videojuegos, existen otros igual de importantes que forman parte de la industria, como las productoras, que tienen un triple cometido (gestionar el dinero de los inversionistas, pagar a los grupos de desarrollo y se encargan de acordar con las distribuidoras de cada país para el lanzamiento del juego); otro grupo son los distribuidores que tienen la función de distribuir y publicitar los juegos dentro de un ámbito local e internacional; muchas veces estas empresas también son productoras y, finalmente, el grupo más importante: los usuarios, que adquieren el juego y con ello provocan el desarrollo de la industria.

Estos grupos son los que a primera vista forman parte en el proceso creativo del videojuego, pero no son los únicos, también influyen los fabricantes de hardware y la pren-

(3) Carlos González Aguilar. Presidente de la Asociación Mexicana de Videojuegos.

sa especializada que tienen mucha importancia e influencia, aunque sea colateral.

La industria del videojuego se caracteriza por ser lucrativa, ya que se enfoca a un mercado mundial sumamente extenso y en constante crecimiento. Hay cifras que nos arrojan un panorama, por ejemplo, en Estados Unidos hubo videojuegos que vendieron alrededor de 50 millones de copias; en España, durante el año 2000, se vendieron 9 millones de unidades de forma legal, una cifra considerable teniendo en cuenta que en este país hay casi 14 millones de hogares, lo que representa una media de casi un videojuego por hogar. En el mercado mundial de entretenimiento, la industria del videojuego superó en ganancias a la de la cinematografía durante el año 2000.

En este año, 2003, se calcula obtener ganancias superiores a los 2 mil millones de dólares y en una proyección a futuro, para el 2015, habrá rebasado al cine y a la televisión; a su vez, la industria cinematográfica y del entretenimiento se apoyan en la del videojuego por ser parte del auge que vive, tal es el caso de *Tomb Raider*, película *hollywoodense* recientemente estrenada y basada en el personaje surgido del videojuego del mismo nombre.

Por otro lado, en un mundo globalizado, donde las tendencias se enfocan al intercambio de información de forma rápida y segura, la industria del videojuego no se queda rezagada, según expertos, el hecho de que haya computadoras personales en prácticamente todo el mundo no es un logro de Microsoft sino de los videojuegos, no es exageración afirmar que la informática se ha desarrollado tan rapidamente gracias a éstos. El mercado del ocio electrónico se está enfocando a los juegos en red a través de la Internet, es así que empresas importantes como *Electronics Arts, Sony, Microsoft, Nintendo* y *Sega* vean como seria oportunidad esta forma de entretenimiento si se considera que el mercado de juegos *on line* ha recaudado, en lo que va del año 2003, alrededor de 6,700 millones de dólares en ganancias, sin embargo, la industria del videojuego no existe como un ente aislado, sino que se mantiene rodeada de un entorno con el que interactúa y que depende directamente de él.

La industria del videojuego se enfrenta a una expansión significativa a medida que los fabricantes de consolas preparan el lanzamiento de nuevos sistemas, según un informe del *The Yankee Group*, el número de hogares en los Estados Unidos con videojuegos crecerá de 35.9 millones en 1998 a 43.5 millones a finales del 2003.

En definitiva los videojuegos son una industria que mueve mucho dinero y cuenta con el potencial suficiente para convertirse en el medio de entretenimiento de este milenio. Enfocándonos en México específicamente, debemos considerar que es un mercado que está iniciando. *La Asociación Mexicana del Videojuego* comentó: *«...estamos consientes de que la sociedad mexicana compra* piratería *porque los precios de los juegos originales y las consolas son muy altos, así, mientras no haya una cultura de videojuego y la* piratería *continúe, a México no se le tomará en cuenta como un mercado serio.»* Sin embargo, a futuro se plantea desarrollar un videojuego en México ya que tiene un costo por lo menos cinco veces menor que en cualquier país desarrollado. En conclusión podemos decir que las bases de inversión, teóricamente, se están desarrollando en el gobierno, pero falta mucho por hacer en materia de seguridad pública y derechos de autor . Tomando como ejemplo el *Worldmasters* de *Aztec-Tech* sólo demuestra que tenemos el potencial humano para competir a nivel mundial y que lo único que se necesita es apoyo gubernamental.

ASPECTOS TEÓRICOS

Ya conocidos los antecedentes y el entorno socio-pedagógico y económico que les rodea; para comprender los videojuegos así como el diseño de los mismos, debemos primero definir lo que entendemos por la palabra juego, así como determinar sus características fundamentales.

Los juegos son una parte fundamental de la vida humana, sin embargo, comúnmente cometemos el error de llamar juego a aquellas actividades en las que cooperamos aunque las encontremos desagradables. Esta ambigüedad presenta dos barreras potenciales para la comprensión de los mismos, primero, el uso liberal de los términos relativos a juegos promueve una idea exagerada de nuestra propia comprensión, es decir, se tiene una idea errónea del tema; esto es, cuando se hace un análisis cuidadoso y crítico ignoramos las complejidades del diseño de juegos. Los diseñadores, cuya habilidad relevante es la programación, se encargan de elaborarlos sin ninguna preparación más que de su propia experiencia como jugadores, ellos valoran su propia comprensión y socavan su potencial de aprendizaje. El segundo obstáculo es la ambigüedad. Se han aplicado los principios y conceptos de juego tan extensamente que han diluido su significado original.

Los diseñadores de juegos no tienen conjuntos bien definidos de términos comunes con los cuales comunicarse. Discusiones sobre diseños de juegos frecuentemente se desintegran en argumentos sobre el significado de los términos, problema que sobresale dado que no se tienen conceptos claros con una adecuada estructuración.

Todos los juegos en general tienen cuatro factores comunes:

1.- Representación.- En primer lugar, un juego es un sistema formal *cerrado* que representa un subconjunto de la realidad. Entiéndase por *cerrado* que el juego es completo y autosuficiente en su estructura. El modelo del universo creado en él es internamente completo, es decir, no se necesita hacer referencias a agentes exteriores al juego.

Un mal diseño en el juego produce disputas ya que permite que se desarrollen situaciones que las reglas no contemplan, esto trae como consecuencia que los jugadores extiendan las reglas para cubrir las nuevas situaciones originando discusiones. Un juego diseñado adecuadamente elimina esta posibilidad dado que las reglas cubren todas las posibles contingencias encontradas en él. Como comentario diremos que un juego que representa un subconjunto de la realidad demasiado grande desafía la comprensión del jugador por lo que se hace casi indistinguible de la vida misma, privando al juego de una de sus características más atrayentes, que es el enfoque en sí mismo. El juego es una colección de partes que interaccionan cada una con las demás, a menudo de forma completa, por lo cual se considera como un sistema, de modo que la fantasía humana sea el agente que transforma una situación objetivamente irreal en una subjetivamente real. Es así que la fantasía juega un papel importante en cualquier juego.

La distinción entre representación objetiva y subjetiva se hace clara al considerar las diferencias entre simulaciones y juegos.

Una simulación es un intento serio de representar de forma precisa un fenómeno real en una forma más manejable.

Un juego es una representación simplificada artísticamente de un fenómeno. Los objetivos de los dos son diferentes; una simulación se crea con fines computacionales o

de evaluación; un juego se crea con objetivos educativos o de entretenimiento existiendo un punto medio en donde las simulaciones y los juegos se combinan.

2.- Interacción.- Algunos medios de representación de la realidad son estáticos, esto es que no cambian su forma de actuar con el jugador conforme pasa el tiempo. Algunos medios son dinámicos, es decir, muestran cambios con el tiempo. Esa es una parte de la diferencia entre una historia y un juego.

Una historia presenta una serie de sucesos en una secuencia temporal que sugiere una relación de causa-efecto; estos sucesos son a menudo deliberadamente ficticios. Verdaderamente el concepto entero de ficción es solamente válido porque los hechos presentados en ésta son secundarios en importancia.

La relación causa-efecto sugerida por la secuencia de hechos son la parte importante de la historia que representa la realidad, no a través de sus hechos, sino por medio de las relaciones causa-efecto sugeridas por la secuencia de éstos, no obstante, los juegos también intentan representar la realidad.

Una diferencia importante entre juegos e historias es que la segunda presenta sus hechos en una secuencia inmutable, mientras que el primero ofrece ramificaciones de posibles secuencias y permite al jugador hacer elecciones en cada punto de ellas para crear así su propia historia. Al jugador se le anima a explorar secuencias alternativas, positivas y adversas.

La interacción es importante por algunas razones; en primer lugar, inyecta un elemento social o interpersonal en el acontecimiento, transforma el reto del juego desde un punto de vista técnico a uno interpersonal.

En la actualidad los videojuegos proporcionan la posibilidad de tener un alto grado de interactividad de modo que lo provee del elemento social.

3.- Conflicto.- El conflicto surge de forma natural en la interacción del juego. El jugador está percibiendo activamente algún objetivo cuyos obstáculos le evitan lograrlo fácilmente y éstos pueden ser de dos tipos: Pasivos o estáticos y activos o dinámicos.

Destacan los activos los cuales requieren de un agente inteligente, si éste bloquea activamente los intentos del jugador para alcanzar sus objetivos el conflicto entre el jugador y el agente es inevitable, por ello el conflicto es fundamental en todos los juegos.

Como comentario señalamos que nuestros conflictos del mundo real son normalmente indirectos, difusos en el tiempo y también frecuentemente les falta resolución. Difícilmente una persona consigue una victoria completa en los conflictos de la vida diaria. Como los juego son representaciones subjetivas de la vida real, centran la atención en un aspecto particular del mundo.

4.- Seguridad.- Un juego, entonces, es un artificio para proporcionar experiencias de conflicto y de peligro mientras se excluyen sus realizaciones físicas.

En suma, un juego es una manera segura de experimentar la realidad, más precisamente, las consecuencias de un juego son siempre menos rígidas que las situaciones que el juego modela.

Por supuesto, también tienen consecuencias, las penalidades de perder puede algu-

nas veces ser una disuasión significativa para jugar.

¿POR QUÉ LAS PERSONAS JUEGAN?

Una manera de responder a la pregunta del por qué las personas juegan es cuestionarse por la historia de los mismos. Los juegos son ahora variados, intrincados y culturalmente complejos para indicar el cumplimiento de cualquier necesidad simple. Quizás los fundamentos de los juegos serían más evidentes en sus primeras versiones.

Si deseamos retroceder a los comienzos de los juegos, es válido sostener que la motivación fundamental para todos los que juegan es aprender. Jugar es una forma segura de aprender, sin embargo, no se necesita ser consciente de ello. Otras tienen poco que ver con el aprendizaje y pueden asumir una mayor importancia particular que la motivación ancestral de aprender, estas incluyen: Fantasía, ponerse a prueba uno mismo, por conveniencias sociales, ejercicio y la necesidad de reconocimiento.

ASPECTOS TEÓRICOS DE LOS VIDEOJUEGOS

En el mismo ámbito de los juegos, a partir de la creación de las computadoras, se encuentra un nuevo tipo conocido como videojuego, en él, la computadora actúa como oponente y/o árbitro en la mayoría de los casos, además de proporcionar una gran variedad de gráficos animados. La forma más común de videojuego es la de impulsar la habilidad y acción que enfatizan la coordinación de la mano y la vista.

Entonces, dado que éstos forman parte de los juegos, deben cumplir los aspectos arriba mencionados, sin embargo, para desarrollar un videojuego también es necesario tener en cuenta los siguientes aspectos:

Sector del mercado.- Este punto se refiere a la parte del estrato social al cual se enfocará el videojuego, se deben considerar cuatro grupos que son: Niños, adolescentes, adultos y toda la familia.

Cabe mencionar que el sector del mercado esta directamente implicado con el contenido.

Antecedentes.- En éste debemos considerar que el videojuego es un modelo internamente completo, por lo que sólo es necesario hacer referencia a agentes externos como historietas, caricaturas, películas, otros videojuegos, etc., para contribuir en la creación de la trama o historia que se desarrollará, logrando que el mundo virtual del videojuego sea interesante y retador. Por lo tanto es indispensable saber que la trama del videojuego es parte fundamental en el diseño del mismo.

Plataforma.- Dentro de ésta se tendrá que especificar en qué sistema operativo estará sustentado, por ejemplo: *Windows, Mac, Linux,* etc. Sin olvidar que al tener un buen diseño en la parte del *game engine* este punto pasaría a segundo término ya que sólo dependería del compilador que se utilice.

Tipo de juego.- Tomando en cuenta la información recabada en el punto: *Importancia de los Videojuegos a Nivel Sociocultural* y basándonos en los diferentes textos que hacen

una clasificación más objetiva de los videojuegos estos se dividen en los siguientes tipos:

1.- Estrategia.- Los juegos de estrategia enfatizan el pensamiento lógico y la planeación creando una tensión mental y manejo del tiempo, haciendo que usualmente el jugador tenga que tomar decisiones rápidas y se involucre con el personaje. Una organización y ejecución táctica son necesarias dado que los diseñadores dejan en manos del jugador las decisiones primordiales del desarrollo de éstos como el distribuir sus elementos así como cuándo incrementar sus habilidades.

Hay juegos de estrategia en los que se involucra el tiempo, estos agregan un elemento más porque debe considerarse que pueden ocurrir múltiples eventos al mismo tiempo.

2.- Acción.- Los juegos de acción son mejor conocidos como *juegos de coordinación de manos y ojos,* en ellos también se necesita una estrategia y casi siempre son en tiempo real demandándole al jugador la coordinación de la mano con la vista y un tiempo de reacción rápida.

3.- Deportes.- Este tipo de juegos modela deportes populares. Ahora bien debido a la falta de ideas originales en los primeros años de los videojuegos los diseñadores miraron los deportes para simularlos. En la actualidad este tipo es una rama especial de los juegos de acción ya que requieren de las mismas habilidades.

Se tienen varias modalidades, uno contra uno, uno contra la computadora o equipo contra la computadora y finalmente equipo contra equipo (dos contra dos), cabe mencionar que el realismo con el que se desarrollan son un parte fundamental de los mismos.

4.- Simuladores.- En cierto aspecto todos los videojuegos que imitan sucesos reales en la pantalla son simulaciones, sin embargo, estos videojuegos suponen una situación de la vida real (a veces compleja, otras sencilla) que el jugador esta invitado a controlar. La computadora le pide que tome decisiones y después actualiza el modelo utilizando sus elecciones como variables, es reconfortante saber que si éstas decisiones son malas las consecuencias no afectaran la vida real. Los simuladores más comunes recrean el manejo de máquinas como aviones, helicópteros, tanques y submarinos.

5.- Aventura.- Esta clasificación es una extensión natural de las simulaciones, la diferencia radica que en lugar de imitar al mundo real la mayoría de ellos crean un mundo fantástico. El objetivo del jugador no es intentar mejorar el mundo, sino aventurarse a través de él, por ejemplo resolviendo un problema o encontrando un tesoro.

Los juegos más largos y complejos complican la situación casi al infinito, existen unos tan elaborados que sólo algunos jugadores los terminan. El placer está en irse acercando un poco más hacia el objetivo final.

Existen juegos en tiempo real haciendo que la trama del juego se vuelva más interesante con la generación de eventos que no dependen de las decisiones del jugador.

6.- RPG (Role Playing Game).- Este tipo de videojuego es similar a los de aventuras, con la diferencia que éstos son de desarrollo y crecimiento, en los cuales se involucran

juegos de estadística, estrategia en combate para resolver un rompecabezas, etc.

Por otro lado, una de las diferencias principales es que mientras uno sigue la línea del juego de aventuras en los de RPG vamos armando o buscando las piezas fundamentales para formar nuestra propia historia.

7.- Tableros.- Este tipo son la versión en computadora de los diferentes juegos lúdicos o clásicos, como los ya conocidos juegos de mesa: *Ajedrez, backgamon, damas chinas, solitario, black jack, damas españolas, rompecabezas,* etc.

8.- Educativos.- Aunque todos los videojuegos son, de alguna forma, educativos, éstos se diseñan teniendo en cuenta objetivos pedagógicos explícitos. Este grupo no está muy poblado ya que existen muy pocas personas interesadas en el uso educativo de los videojuegos.

9.- Galería de Tiro.- Son una rama de los juegos de acción; consiste en disparar a ciertos elementos. En un principio éstos eran fijos, en la actualidad tanto el objetivo como el personaje principal son dinámicos logrando complicar el desarrollo del juego.

PERSPECTIVA VISUAL EN EL VIDEOJUEGO

Primera persona.- Se refiere a la persona en un videojuego cuando el tipo de gráficas que se observa fuera como si estuviese viendo a través de los ojos del personaje del juego.

Tercera persona.- Se refiere cuando el jugador puede ver en escena al personaje con el que está jugando en la pantalla; esta perspectiva tiene la ventaja de observar el ambiente en el que se está desarrollando el juego, así como los movimientos que puede hacer el personaje (lo que en primera persona es imposible).

Esta visión también es conocida como *vista sobre el hombro.*

Tablero (Top down).- Este tipo de perspectiva consiste en ver desde arriba la escena del juego. Es utilizada en los de tablero, así como en muchos de estrategia.

Isométrica.- Comúnmente confundida con la vista *top down,* este tipo de perspectiva da la opción del juego en tres dimensiones, muchos desarrolladores de videojuegos prefieren esta vista porque es posible ver el ambiente y el personaje al mismo tiempo.

Vista plana de lado.- Generalmente conocida como *flat* o *side view*, es la vista tradicional de dos dimensiones que ha ido perdiendo terreno con respecto a las demás perspectivas dado que el personaje solamente se podía mover en dos direcciones, vertical y horizontalmente.

Juegos basados en texto.- Existen muy pocos videojuegos de este tipo, es decir, que no utilizan gráficos, en ellos no se tiene ninguna perspectiva del personaje dado que todo se presenta con texto en la pantalla.

INTERFAZ

Se refiere a cada uno de los elementos que conforman el videojuego y que de cierta forma ayudan a la calidad y a la atracción que tienen como característica.

A su vez los elementos que conforman la interfaz de un videojuego son:

1.- Escenario.- Dentro de la interfaz de escenarios es importante mencionar que es una parte fundamental para el videojuego, ya que ésta nos dará las perspectivas y el contexto en dónde se desarrollarán las actividades, además, tiene que ir de acuerdo con la trama que compone la historia del juego.

El escenario puede consistir en construcciones de edificios, autos, laberintos, etc.

2.- Personajes.- El diseño de personajes es importe ya que estos harán que el jugador se identifique con alguno de ellos (de ser posible) por sus características físicas, habilidades, nombre, etc.

También hay que considerar la trama de la historia de modo que los personajes, además del protagonista, sean lo más apegados a la realidad del contexto.

3.- Objetos.- Los objetos se dividen en dos clases: Estáticos y dinámicos. Los estáticos únicamente son parte del escenario para dar mayor realismo al juego, mientras que los dinámicos, aparte de dar realismo, se vuelven parte fundamental para el desarrollo de la historia.

4.- Comportamiento.- Se refiere a la forma correcta en que interactúan los elementos que intervienen en la interfase como los objetos y los personajes. Se debe considerar que el comportamiento que tienen estos elementos debe ser lo más cercano a la realidad que marca la trama del videojuego, sin dejar pasar desapercibido el lado fantástico del juego, es importante tomar en cuenta las características que los conforman, ya que finalmente estos elementos darán vida al juego.

5.- Audio (música y efectos especiales).- Con respecto al audio se pueden considerar dos partes importantes: La música, que sirve para ambientar las diferentes escenas de la historia sin que aporten información al jugador; y los efectos especiales, que son fragmentos complementarios de ambientación que ayudan a dar más realismo a la acción del videojuego, así como también en algunos casos proveen al jugador de información durante el desarrollo del juego indicando algún suceso.

6.- Jugabilidad.- Los elementos que constituyen la jugabilidad y que forman parte fundamental del juego son las reglas, los objetivos y las metas; claro está que debe haber una relación directa entre cada uno de estos elementos.

Como ya se ha mencionado, las reglas deben estar previamente definidas y diseñadas ya que influirán en los objetivos y metas de la trama. Con esto se pretende que el jugador tenga la satisfacción de culminar sus logros ya que al encontrarse una adversidad constante e inminente (dado la falta de objetivos y metas claras), provocaría que el jugador desistiera.

GAME ENGINE

Game engine o máquina de videojuegos; con ello nos referimos a la parte del software que nos ayudará a integrar las diferentes partes del juego como: Escenarios, personajes, música y efectos sonoros, periféricos de entrada y/o salida (controles), el manejo de bases de datos que definen el comportamiento de los eventos y personajes, principalmente.

CAPITULO III

ANÁLISIS

ASPECTOS TEÓRICOS Y APLICACIÓN HACIA EL VIDEOJUEGO

A partir de esta etapa se iniciará el desarrollo del videojuego utilizando los conceptos de ingeniería. La información contenida en los siguientes capítulos se estructura en tres fases:

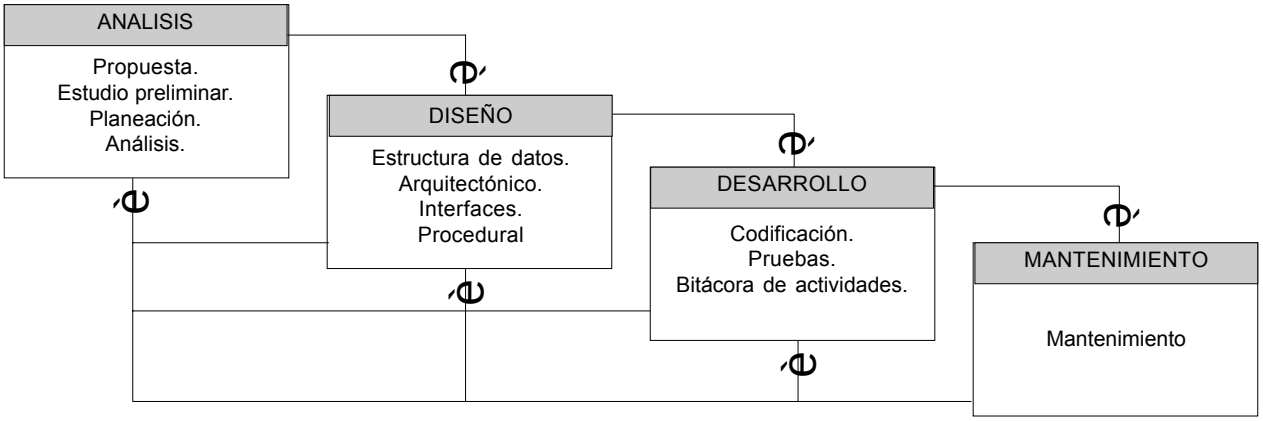
1.- Aspectos teóricos detallados de la ingeniería de software.- En ésta se recaba la información teórica de forma más precisa para sustentar las etapas que conforman el ciclo de vida del producto final.

2.- Referencias teóricas de los videojuegos.- En ésta se hace mención de todos los aspectos teóricos (definidos en el capítulo anterior), que forman parte de cada etapa que integra el ciclo de vida del producto final, y

3.- Información para la aplicación real.- En esta etapa, y de acuerdo a la información obtenida en los dos puntos anteriores, se sustenta el desarrollo del videojuego que a su vez fungirá como objetivo final de este proyecto.

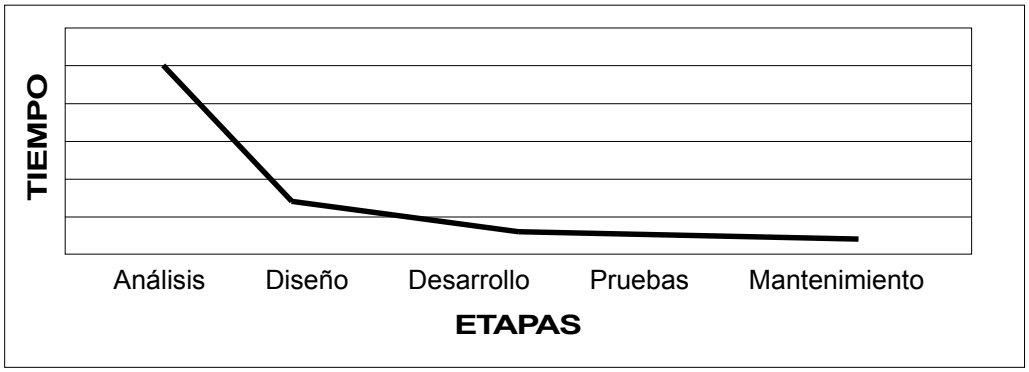
Por lo que respecta al ciclo de vida o paradigma que se utilizará para el desarrollo del trabajo se establece el denominado Top down-bottom up. Este tiene sus orígenes en el *clásico* o llamado también de *cascada*, con la diferencia de que el *Top down–bottom up* nos permite regresar hacia alguna etapa preliminar de modo que, en su caso, se logra depurar los errores que se presenten y que no fueron observados o contemplados en su momento. El detalle de este ciclo de vida establecido se observa en la gráfica 13.

Cabe resaltar que las dos primeras etapas, es decir la de análisis y diseño, son las de mayor importancia ya que de acuerdo a la documentación de las mismas se reflejará la calidad que tendrá el producto final. Con esto se pretende conseguir un comportamiento



Gráfica 13

tal como en el diagrama de demanda de un producto, en el que se logra ejemplificar las diferentes etapas del ciclo de vida. (Ver gráfica 14)



Gráfica 14

De acuerdo al ciclo de vida de cualquier sistema, la etapa de análisis es aquella que sienta las bases necesarias para un adecuado diseño y desarrollo de la aplicación. En este capítulo se verán a detalle los elementos que la conforman desde el punto de vista teórico para posteriormente aplicarlo en el videojuego.

PROPUESTA

El análisis de los sistemas empieza tomando en consideración una visión global. Se analiza el dominio de negocio o producto para establecer todos los requisitos básicos. El enfoque se estrecha a una visión de dominio donde cada uno de los elementos del sistema se analiza; cada uno de éstos es asignado a uno o más componentes de análisis que son, por consiguiente, estudiados por la disciplina más apegada a ese fin.

En la ingeniería de software existe un documento en el cual se engloba toda la información necesaria que generan los elementos que conformarán la aplicación, este documento es generalmente conocido como *Propuesta* que es propiamente un escrito que detalla aspectos fundamentales como: Nombre del proyecto, objetivos, recursos que intervienen (humanos y materiales), costos asociados, tiempos estimados, condiciones comerciales, etc.

Una parte primordial en la propuesta es la especificación del sistema, la cual es un contenido que sirve como fundamento para la ingeniería de software, de hardware, bases de datos y humana. La especificación del sistema describe la función y el rendimiento de un sistema basado en computadora y las restricciones que gobernarán su desarrollo, también delimita cada elemento del sistema asignado, por ejemplo, proporciona al desarrollador una visión del papel del software dentro del contexto del sistema en su totalidad y los distintos subsistemas descritos en los diagramas de flujo de la arquitectura, asimismo describe la información que se introduce y se extrae del sistema. Debe resaltarse que existen diversos esquemas que pueden emplearse para definirla, por ejemplo, el formato y el contenido real que lo dictan los estándares de software o de ingeniería de sistemas, o bien las costumbres, preferencias y experiencias de los desarrolladores.

La propuesta, independientemente del modo como la realicemos, puede verse como un proceso de representación; los requisitos se representan de manera que, como fin último, tenga éxito la implementación del software. Se debe tomar en cuenta ciertos principios que servirán para una adecuada formulación de especificaciones, como la separación de la funcionalidad y de la implementación, definir el entorno en que va a operar el sistema, reconocer que la especificación debe ser tolerante a un posible crecimiento si no es completa. Una propuesta es siempre un modelo o abstracción de alguna situación real que normalmente suele ser compleja.

Todos estos principios básicos proporcionan una base sólida para representar los requisitos del software, pero deben llevarse a la realidad.

A continuación se establecen los contenidos que formularán nuestra propuesta para el desarrollo de la aplicación final, en este caso de un videojuego:

Respecto al nombre del proyecto, hemos decidido llamarlo: ***Para titularse, es cuestión de un laberinto.***

Dentro de éste se pretende cumplir con los siguientes objetivos:

1.- Desarrollar un videojuego utilizando la metodología *Top down-button up.*

2.- Dar a conocer que el desarrollo de videojuegos puede ser un campo de acción viable en todos los aspectos.

3.- Demostrar que aplicando las bases de la ingeniería de software el desarrollo de videojuegos puede ser más sencillo y mejorado, a diferencia de un desarrollo empírico.

4.- Por la parte funcional buscar un desarrollo mental y de coordinación en el usuario a través de la dinámica del juego mismo.

5.- Una sana diversión en la elaboración del proyecto de modo que se refleje en el entretenimiento del usuario, y

6.- Un objetivo de carácter más personal que es la obtención del título profesional.

Dentro de los recursos humanos y materiales con los que contamos para el desarrollo de este proyecto, se mencionan los siguientes:

Recursos humanos: Las tres personas que integramos el equipo (Carlos Aroche, Moisés León y Arturo Vidal), en conjunto con la asesoría de nuestro director de tesis Ing. Santiago Igor Valiente. No debemos dejar de reconocer que en la parte de análisis y diseño se requerirá de la aportación de ideas de profesionales en áreas como diseño gráfico, diseño artístico, aspectos pedagógicos, etc., mencionados en los créditos del producto final.

Recursos materiales: Se cuenta con tres equipos de cómputo y aplicaciones, los cuales pertenecen a cada uno de los integrantes del equipo.

De acuerdo a las características de este proyecto, debemos indicar que son fundamentalmente académicas, mientras que los costos asociados y las condiciones comerciales son elementos que no se toman en cuenta en el desarrollo del mismo, sin embargo, y de acuerdo a la información del capítulo anterior, no podemos dejar de reconocer que el aspecto comercial es viable.

El tiempo estimado para el desarrollo de este proyecto es de, aproximadamente, entre 26 y 28 meses a partir de diciembre del 2000. Hay que aclarar que este lapso está asociado al punto de planeación que se contempla en el análisis, por lo que los diagramas de tiempo estimados se verán a detalle en el apartado correspondiente.

Una vez realizada la propuesta llegamos a una de las partes fundamentales en el desarrollo del videojuego, que es la historia y la temática.

Historia y temática: Nuestra breve reseña consiste en una historia fantasiosa. Comienza cuando un *bogus* o personaje del mundo de la fantasía cansado de ser un elemento de la imaginación decide convertirse en un ser que pueda vivir en la realidad. Ante su gran sueño, el hechicero supremo y regidor del mundo de la fantasía le da a conocer el secreto de una prueba que debe superar para poder pasar a la dimensión que añora.
El sabio maestro le revela que para poder ingresar a la tierra de la realidad y así lograr su sueño, debe cruzar un laberinto que le proveerá de los conocimientos necesarios para poder interactuar con los habitantes de la tierra. Una vez que el personaje tiene conocimiento de la ubicación exacta de la prueba, inicia su travesía con el objetivo de llegar al laberinto e iniciar la aventura con la que cumplirá su meta.

En esta etapa se inicia el juego.

El laberinto se compone de pasillos y salas dentro de las cuales el usuario deberá resolver las diferentes *trivias* o serie de preguntas para llegar al final. Por lo que respecta a las *trivias*, éstas podrán ser de diversos temas: Ciencia, entretenimiento, conocimientos generales, etc.

De acuerdo a la dinámica del videojuego las preguntas no estarán directamente asociadas a una sala específica, es decir, se realizarán de manera aleatoria independientemente de la sala en que se encuentre el protagonista. Por obviedad, si la *trivia* no es resuelta correctamente se tendrá que abandonar la sala y con esto perder la oportunidad de avanzar.

ESTUDIO PRELIMINAR

Dentro de este apartado cabe mencionar que es muy importante conocer al cliente, sin embargo, en nuestro caso no se cuenta con uno específico, ya que el enfoque se orienta hacia un estrato o grupo social. El grupo al que nos referimos, y que de acuerdo a la temática que nos refleja la historia, es el de niños o estrato infantil.

Cualquier enfoque que se le quiera dar al desarrollo de videojuegos, debe descansar sobre un empeño de organización de calidad. La gestión total de calidad y las filosofías similares fomentan una cultura continua de mejoras de procesos, y es ésta la que conduce al desarrollo de enfoques cada vez más robustos en la elaboración de aplicaciones.

Como un elemento del estudio preliminar en la ingeniería de software, se debe esta-

blecer un marco común de proceso, en nuestro caso, se deben definir un pequeño número de actividades del marco de trabajo aplicables al desarrollo del videojuego. Otra parte fundamental del estudio preliminar es la conocida como *Ingeniería y Modelado del Sistema.* Como el videojuego por sí mismo interactúa con el usuario, se deben establecer requisitos a cada uno de sus elementos. Esta visión es esencial cuando la aplicación se debe interconectar con otros elementos tales como hardware, bases de datos y jugadores.

De acuerdo a las características del ciclo de vida que se propone, nuestra metodología trabaja sobre un enfoque sistemático, es decir, lo hace de manera secuencial en el desarrollo del videojuego, pero además permite el retomar etapas anteriores con el objetivo de ir depurando los detalles.

El estudio preliminar comienza con el conocimiento y entendimiento del entorno, posteriormente se continúa con el proceso de reunión de requisitos tomando en consideración las siguientes tareas: Identificar las necesidades del estrato usuario, establecer el objetivo del sistema para evaluar la viabilidad y, finalmente, realizar análisis en el campo técnico (asignar los recursos materiales y humanos). Esto servirá de base para continuar con el análisis, el diseño, la codificación y el mantenimiento.

Dentro de la etapa del entendimiento del entorno, se definen todos los elementos que conforman al videojuego en específico y la forma en que interactúan con todos los elementos externos, esto con el simple objetivo de poder asignarles una jerarquía en el ámbito del proyecto. En la parte del manejo de información, se realiza un estudio a la arquitectura de datos y de aplicación, así como a los avances de la tecnología para tener como objetivo principal la satisfacción de las necesidades generales del sistema.

Por lo que respecta al proceso de identificación de requerimientos y necesidades. Se lleva a cabo un estudio al estrato desde el punto de vista social, pedagógico y económico con la intención de entender los objetivos de la aplicación, así como para definir las metas para alcanzarlos.

Una vez que se ubican los objetivos globales se requiere de cierta información suplementaria; dentro de esta recopilación se deben considerar algunos agentes directamente relacionados con los usuarios, es decir, los supuestos, las simplificaciones, las limitaciones, las restricciones y las preferencias.

La viabilidad como proyecto está muy relacionada con el análisis de riesgo, por lo que es necesario evaluarla de manera complementaria con el análisis.

Existen dos áreas principales de interés: Viabilidad técnica y diversas alternativas; la primera es frecuentemente la más difícil de valorar y se debe considerar el riesgo de desarrollo, la disponibilidad de recursos y la tecnología.

El análisis técnico empieza con una valoración de la viabilidad técnica y se evalúan los principios tecnológicos del videojuego al mismo tiempo que se acopia información adicional sobre el rendimiento y fiabilidad; en algunos casos esta fase también incluye cierto diseño, posteriormente se realiza el análisis de los elementos que conforman el estudio preliminar de nuestro juego desde el punto de vista social, pedagógico y económico.

Cabe mencionar que de acuerdo a la metodología propuesta para este trabajo, los elementos pertinentes a los aspectos teóricos son: Antecedentes, plataforma, tipo de juego, perspectiva y sector del mercado.

Antecedentes: Desde la perspectiva pedagógica, los juegos de laberinto lejos de parecer metódicos presentan diversas formas de obtener las metas, en este caso el *Triunfo*, por lo que es necesario establecer varias estrategias, logrando con ello agilizar la mente del jugador con el objetivo de fomentarle la solución a problemas, la formación de habilidades, etc. Además, las salas *(trivias)* tienen el objetivo de retroalimentar la información acerca de los tópicos o temas correspondientes a cada una de ellas. En la interfase se busca crear un ambiente amigable hacia el usuario, logrando un acercamiento familiar y de confianza con el sistema completo. (En el caso del hardware, la computadora y en el del software, el videojuego)

Es muy común escuchar lo entretenido que son los juegos, es por eso que la temática de nuestra aplicación se orienta, desde el aspecto social, a crear un ambiente sano y lejano a la violencia, logrando a su vez que el usuario vea los videojuegos como entretenimiento y al mismo tiempo le forme, intelectualmente, un reto a vencer para que aprenda a desenvolverse e interactuar con los demás miembros de la sociedad.

De acuerdo con los datos arrojados en el estudio del ILCE visto en el capítulo anterior, se buscó un videojuego de estrategia simple (laberinto) con el objetivo de satisfacer las inquietudes de los encuestados.

Aunque no es un punto a considerar por la finalidad de este trabajo (netamente académico), el aspecto económico y las cifras señaladas en el enfoque relativo, podemos indicar que cualquier contexto comercial que se le quiera dar a nuestro videojuego formará parte de un mercado en constante crecimiento.

Plataforma: Debido a las características del proyecto de tesis y a la capacidad de recursos en materia de hardware, se optó por trabajar sobre plataforma de PC con sistema operativo *Windows*. Por lo que toca a la parte del desarrollo, se operará con las herramientas de programación: El diseño estructural del laberinto se lleva a cabo en *Autocad 2000,* la parte de diseño artístico, es decir, mapeo de texturas, iluminación y ambientación, se ejecutan en *3D Max Studio*, además la flexibilidad de esta herramienta se utiliza para la exportación de los archivos a los formatos que el Fly 3d *(Game engine)* necesita.

Tipo de juego: De acuerdo a la clasificación que se llevó a cabo en los aspectos teóricos del capítulo anterior y a la historia y temática explicada con anterioridad, podemos decir que nuestro videojuego puede clasificarse como una mezcla entre *Juego de aventura* y de *tablero*.

Perspectiva: Dentro de este apartado se ha considerado usar la perspectiva en primera persona, la cual se utilizará a lo largo del desarrollo de éste, es decir, en la navegación a través del laberinto así como en la interactividad dentro de las salas.

Sector de mercado: Este punto tiene su antecedente en el inicio del estudio preliminar en donde se refiere a un estrato social infantil.

PLANEACIÓN

Se conoce como planeación toda estimación de recursos (tiempos, actividades y costos)

necesarios para llevar a cabo un proyecto de desarrollo.

Para establecer una correcta planeación se debe tener en consideración algunos aspectos que pueden ser evitados a través de la experiencia de conocimientos técnicos y de manejo de recursos, estos aspectos son de gran variedad pero los que destacan son: Fecha de entrega o de finalización poco realista, cambio de requisitos, errores predecibles y no predecibles, dificultades técnicas y humanas, etc. La planificación de un proyecto de software es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería de software. Es importante resaltar que la planificación evoluciona con el tiempo; durante las primeras etapas se identifican las actividades principales así como las funciones del producto a las que se aplica, a medida que el proyecto va mejorando, la planificación se refina o detalla, es decir, se identifican en el programa tareas específicas.

La planificación de proyectos se debe guiar por una serie de principios básicos: Compartimentación, interdependencia, asignación de tiempo, resultados definidos, etc. Para poder hacer una planeación que tenga cierto grado de certeza y eficacia, debe definirse un conjunto de tareas, cada una diseñada para satisfacer las necesidades del proyecto; estas son una colección de etapas que deben cumplirse con disciplina para alcanzar la calidad optima del videojuego. Para desarrollar una planificación de proyecto, se deben distribuir todos nuestros conjuntos de tareas a lo largo de la duración del mismo; claro está, para poder tener un control de desarrollo, teóricamente, se debe usar un modelo de proceso lineal secuencial, iterativo o evolutivo de tareas. Como hemos mencionado, las tareas principales pueden emplearse para definir una planificación temporal del proyecto, sin embargo, esta planeación debe refinarse para crear una detallada del proyecto.

El refinamiento se empieza tomando cada tarea principal y descomponiéndola en un conjunto de subtareas. Éstas, apuntadas en el refinamiento, forman la base para una planificación de las actividades en el desarrollo del proyecto, incluso, hay que ejecutar un cierto número de pequeñas tareas antes de alcanzar el objetivo; algunas de estas quedan fuera del camino final y pueden realizarse sin preocuparse del impacto en la fecha de terminación, otras se encuentran en un camino crítico, si estas se retrasan el proyecto entero se pone en riesgo. Actualmente, existen herramientas útiles para la planeación y el control de proyectos: *Ruta crítica, Diagrama de Gantt* y *Cronograma*.

Para la herramienta de ruta crítica, se plantean las diferentes actividades del proyecto asociándolas como un conjunto de tareas (representación gráfica del flujo de actividades de un proyecto). Esta se emplea como mecanismo a través del cual se introduce la secuencia de tareas y las dependencias entre las mismas con el objetivo de determinar el camino crítico, es decir, aquellas actividades necesarias que no se pueden retrasar en su desarrollo, ahora bien, teóricamente, se emplean dos formas de análisis para la ruta crítica: *Pert* y *CPM*. En el caso de Pert, debe tomarse en cuenta tres diferentes tiempos (optimista, más probable y pesimista) basados en la distribución de probabilidad beta, mientras que para el CPM, entran en juego los costos asociados a cada actividad o tarea; de acuerdo a ésto y a las características de este trabajo, para nuestro videojuego no se aplica ninguno de estos análisis de forma rigurosa debido a que no intervienen diversos tiempos (sólo la meta flexible fijada por los miembros del grupo de trabajo) y tampoco los costos asociados por ser de carácter netamente académico.

Nuestro análisis para ruta crítica, se basa en la descomposición del trabajo en una red de tareas o actividades, sólo tomando en cuenta el plazo de tiempo comprendido desde la fecha de inicio hasta el momento final de cada una de ellas. Como consecuencia de esta información se genera un gráfico de tiempos, también llamado diagrama de Gantt.

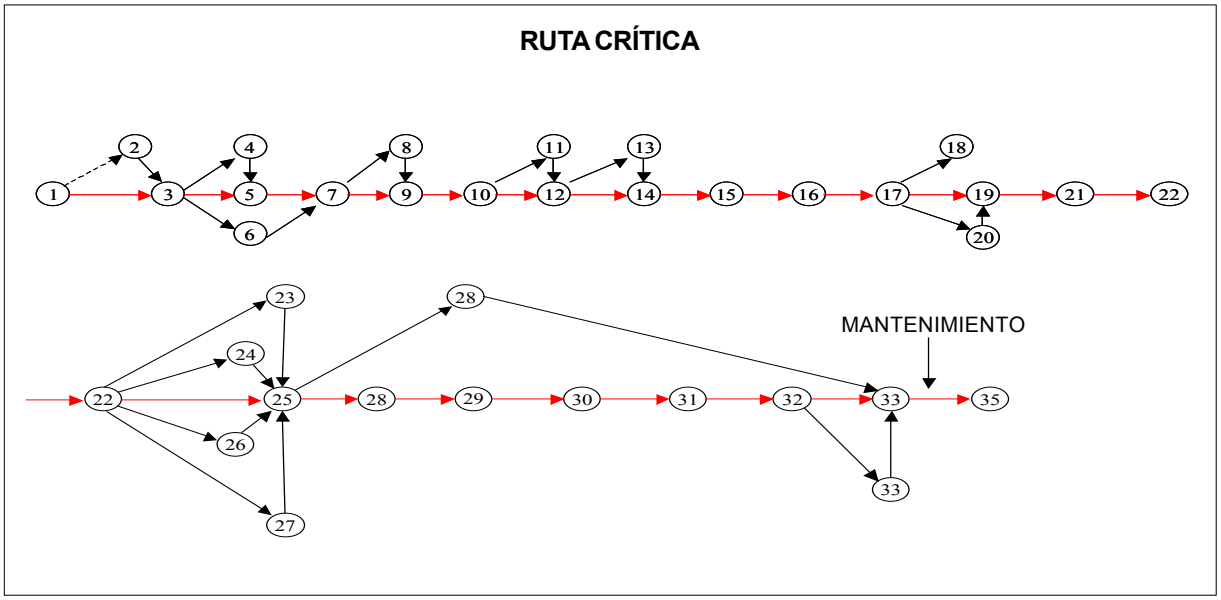
Se puede desarrollar un gráfico de tiempo para todo el proyecto, alternativamente se pueden desarrollar diferentes esquemas para cada actividad o para cada recurso que intervenga en el proyecto.

En un diagrama de Gantt todas las tareas del proyecto se enlistan en una columna, posteriormente y mediante barras horizontales se indica la duración en tiempo de cada una de las actividades; es posible que se puedan traslapar actividades, sin embargo, al ser un elemento gráfico muestra avances o retrasos en una forma muy sencilla. Una vez que se ha introducido la información necesaria para crear este tipo de diagrama, también como parte de la planeación, se generan tablas de proyecto o cronogramas, que son un listado tabular de todas las actividades del desarrollo, sus fechas previstas y reales de inicio y finalización, e información variada y relativa de la actividad.

Empleando las tablas junto con los gráficos de tiempo se puede tener conocimiento del seguimiento y progreso de cada actividad. Por otro lado, y dado que los aspectos de este trabajo son netamente académicos y no de tipo comercial, estos se mencionan pero no se toman en cuenta como parte fundamental de la planificación.

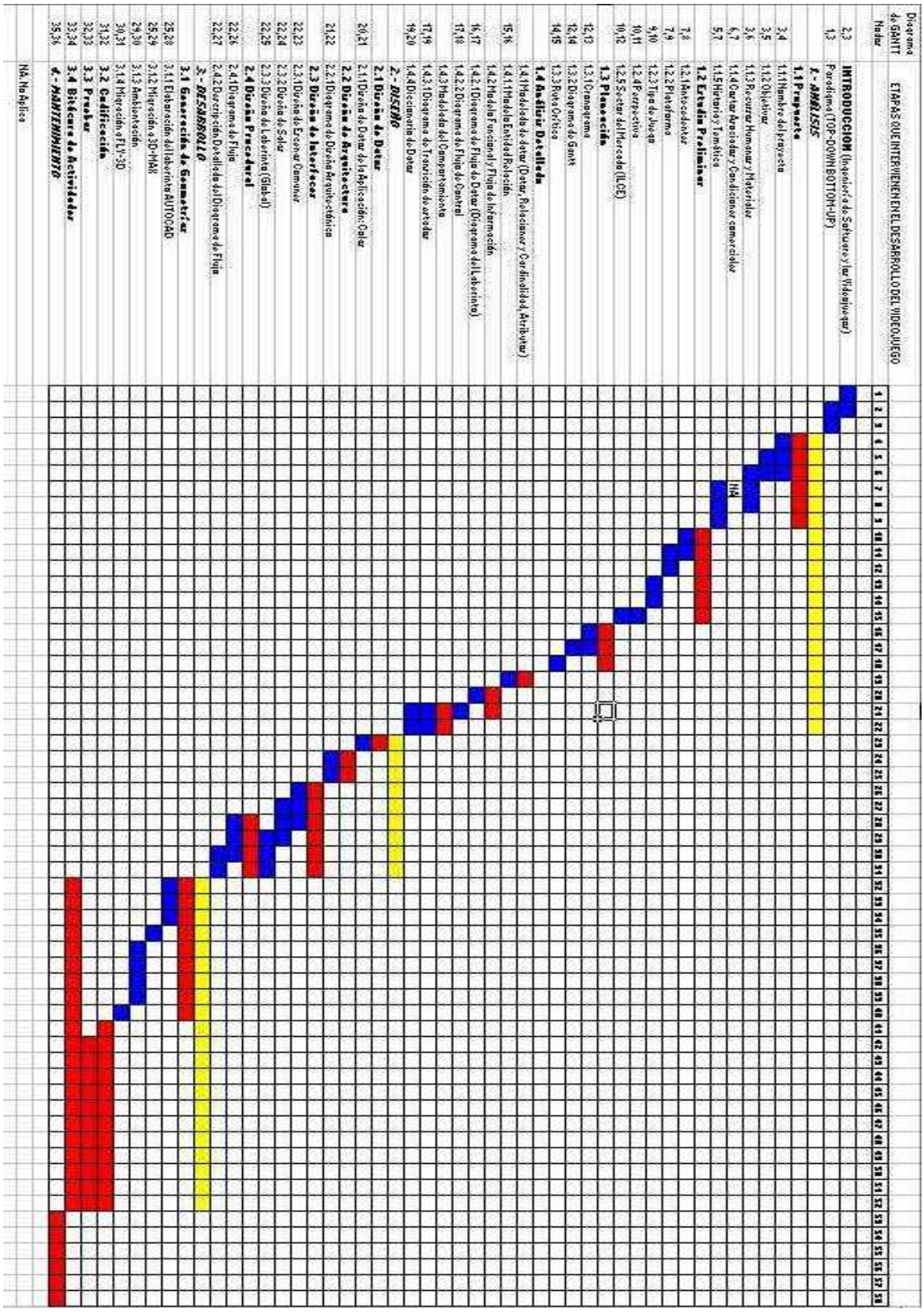
Los aspectos comerciales es un documento que se produce en la culminación de las tareas de planificación y proporciona la información necesaria de recursos y costos que intervienen en el desarrollo del videojuego. El propósito final de la planificación en conjunto con la propuesta, es que nos proporciona una idea de las condiciones del proyecto.

Hay que tomar en cuenta que para nuestra aplicación es importante en el contexto de su entorno y revisar el ámbito en el que se emplea para generar las estimaciones necesarias. En resumen, podemos decir que la parte de la planeación se observa claramente en la ruta crítica, diagrama de Gantt y cronograma. (Ver gráficas 15, 16 y 17)



Gráfica 15

DIAGRAMA DE GANTT



CRONOGRAMA

INTRODUCCION

(Ingeniería de Software y los Videojuegos)
Paradigma (TOP-DOWN BOTTOM-UP)

1.- ANÁLISIS

1.1 Propuesta.

1.1.1 Nombre del Proyecto.
1.1.2 Objetivos.
1.1.3 Recursos Humanos y Materiales.
1.1.4 Costos Asociados y Condiciones Comerciales.
1.1.5 Historia y Temática.

1.2 Estudio Preliminar.

1.2.1 Antecedentes.
1.2.2 Plataforma.
1.2.3 Tipo de Juego.
1.2.4 Perspectiva.
1.2.5 Sector del Mercado (ILCE).

1.3 Planeación.

1.3.1 Cronograma.
1.3.2 Diagrama de Gantt.
1.3.3 Ruta Crítica.

1.4 Análisis Detallado.

1.4.1 Modelado de datos (Datos, Relaciones y Cardinalidad, Atributos).
1.4.1.1 Modelo Entidad Relación.
1.4.2 Modelo Funcional y Flujo de Información.
1.4.2.1 Diagrama de Flujo de Datos.
1.4.2.2 Diagrama de Flujo de Control.
1.4.3 Modelado del Comportamiento.
1.4.3.1 Diagrama de Flujo de Control.
1.4.4 Diccionario de Datos.

2.- DISEÑO

2.1 Diseño de Datos.

2.1.1 Diseño de Datos de la Aplicación: Colas.

2.2 Diseño de Arquitectura.

2.2.1 Diagrama de Diseño Arquitectónico.

2.3 Diseño de Interfaces.

2.3.1 Diseño de Escenas Comunes.
2.3.2 Diseño de Salas.
2.3.3 Diseño de Laberinto (Global).

2.4 Diseño Procedural.

2.4.1 Diagrama de Flujo.
2.4.2 Descripción Detallado del Diagrama de Flujo.

3.- DESARROLLO

3.1 Generación de Geometrías.

3.1.1 Elaboración del Laberinto AUTOCAD.
3.1.2 Migración a 3D-MAX.
3.1.3 Ambientación.
3.1.4 Migración a FLY-3D.

3.2 Codificación.

3.3 Pruebas.

3.4 Bitácora de Actividades.

4.- MANTENIMIENTO

ANÁLISIS DETALLADO

Para el éxito de un desarrollo es esencial una comprensión total de los requisitos del software, si un videojuego no se analiza correctamente; el diseño o codificación pasa a segundo término.

La tarea del análisis detallado consiste en un proceso de descubrimiento, refinamiento, modelado y especificación. Se refina en detalle el ámbito del software, inicialmente establecido por el programador y mejorado durante la planificación del proyecto.

El análisis y la especificación de los requisitos puede parecer una tarea relativamente sencilla, pero para los videojuegos si no se establece realmente el estrato social al que va enfocado o no se realiza un análisis, es muy probable que exista ambigüedad. El análisis de los requisitos es una tarea de la ingeniería de software que cubre el hueco entre la definición a nivel sistema y el diseño del software. Éste permite especificar la función y el rendimiento del segundo, indica la interfaz con otros elementos del sistema y establece las restricciones que debe cumplir; además refina la definición de la aplicación para posteriormente construir los modelos de dominio de datos, funcionales y de comportamiento.

El análisis detallado establece modelos útiles para la etapa de diseño (diseño de datos, arquitectónico, de interfaz y procedimental) y finalmente proporciona al desarrollador medios para valorar la calidad una vez que se ha construido el software; en nuestro caso, un videojuego.

Ahora bien, el análisis del software puede dividirse en cinco áreas de esfuerzo: Reconocimiento del problema, evaluación y síntesis, modelado, especificación y revisión. En nuestro caso las etapas de reconocimiento, de evaluación y síntesis y de especificación se han tratado en los puntos anteriores a este capítulo.

En el análisis también se deben definir todos los objetos de datos observables externamente, evaluar el flujo y contenido de la información, definir y elaborar todas las funciones de la aplicación, entender el comportamiento del software en el contexto de acontecimientos que afectan al sistema, establecer las características de la interfaz y descubrir restricciones adicionales. Durante la actividad de análisis, se deben crear modelos del sistema en un esfuerzo por entender mejor el flujo de datos y de control, el tratamiento funcional y el comportamiento operativo. Los modelos se crean para entender mejor la entidad que se va a construir; éstos deben ser capaces de modelar la información que transforma el software, las funciones que permitan las transformaciones y el comportamiento del sistema cuando ocurren éstas. Durante el análisis del software, los modelos se concentran en lo que deben hacer los sistemas y no en cómo lo hacen; en muchos casos los modelos que creamos emplean una notación gráfica que muestra información, procesamiento, comportamiento y otras características con símbolos reconocibles.

Los modelos creados en el análisis desempeñan un papel muy importante, entre lo que se destaca: Ayuda a entender la información, la función y el comportamiento del videojuego haciendo más fácil y sistemático el análisis; se convierte en el punto de mira para la revisión y por tanto la clave para determinar si se han completado la consistencia y precisión de especificaciones y, finalmente (la más importante) que se convierte en el sustento de la etapa de diseño, proporcionando una representación esencial que puede trasladarse al contexto de la implementación.

En resumen, el análisis estructurado se lleva a cabo a través de tres tipos de modelado: De datos, de flujo de información o funcional y modelado del comportamiento; además complementado con el diccionario de datos.

El modelo de datos permite identificar objetos y sus relaciones mediante una acotación gráfica. En el contexto del análisis estructurado se utiliza como herramienta el diagrama entidad-relación, esta herramienta gráfica define todos los datos que se introducen, se almacenan, se transforman y se producen dentro de una aplicación. Sobre el diagrama entidad-relación éste se centra sólo en los datos, representando una red existente para un sistema dado.

El modelado de datos se compone de tres piezas de información interrelacionadas: El objeto de datos o entidad, los atributos que describen al objeto y la relación que conecta las entidades entre sí. Como breve definición, podemos decir que el objeto de datos es una representación de cualquier composición que deba comprender el videojuego, los atributos se definen como todas aquellas propiedades que describen al objeto y la relación como una forma de conexión entre las entidades. Estos elementos básicos proporcionan la base del entendimiento del dominio de información del videojuego, además de estos elementos también se debe comprender cierta información adicional, en muchos casos la relación no proporciona información suficiente para el propósito que se quiere, esto nos conduce al concepto del modelado de datos llamado cardinalidad, que es la especificación del número de ocurrencias de una entidad que se relaciona con aquellas de otra entidad, esta cardinalidad puede ser de tres tipos: Uno a uno (1:1), uno a muchos (1:N) y muchos a muchos (M:N).

Otro término fundamental en el modelado de datos es la *modalidad*, la cual se define como el parámetro que establece si existe o no una necesidad explícita en una relación. El modelado de datos y el diagrama entidad-relación proporcionan al análisis una notación concisa y útil dentro del contexto de la aplicación, en la mayoría de los casos el enfoque se utiliza para crear una parte del modelado de análisis, pero también para el diseño de bases de datos y para soportar cualquier otro método de análisis de requisitos. La información se transforma a medida que fluye a través de un sistema; como es sabido el sistema acepta entradas en una gran variedad de formas y utiliza sus recursos de igual manera para transformar la entrada en salida. El análisis estructurado se basa en el modelado del flujo y del contenido de la información y a su vez éste emplea como herramienta gráfica el diagrama de flujo de datos, complementado con el de control.

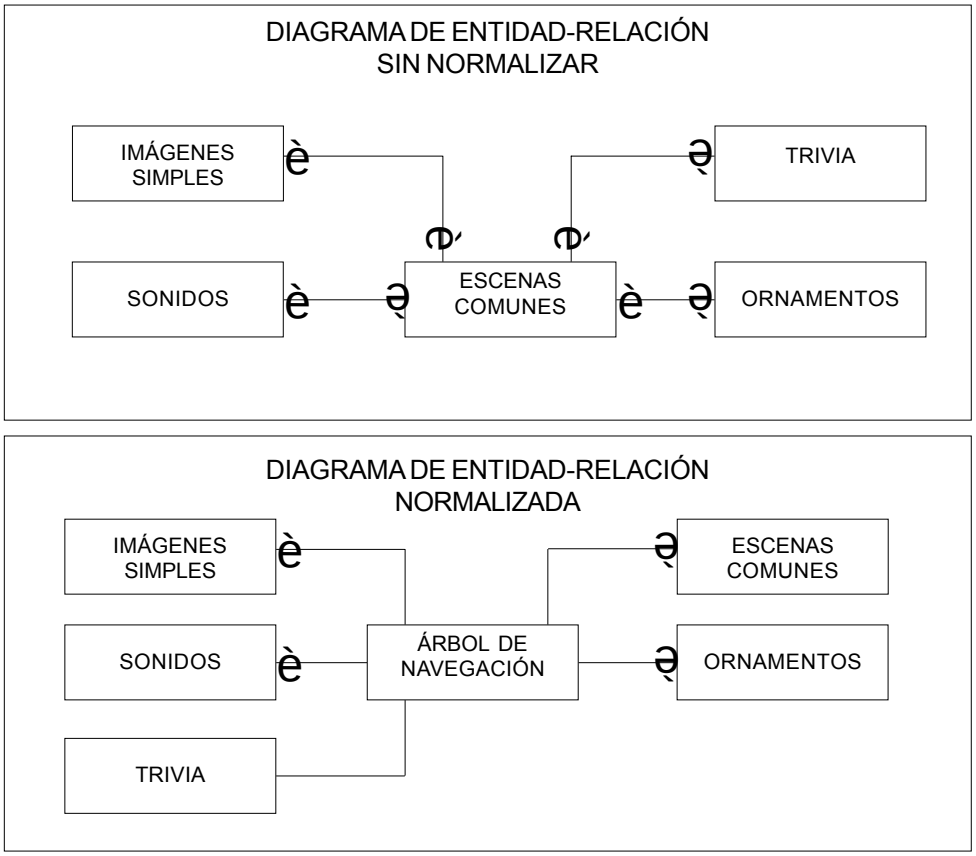
El diagrama de flujo de datos es una técnica que representa el fluir de la información y las transformaciones que se aplican a los datos al moverse de la entrada a la salida. Se puede utilizar esta herramienta para representar un software a cualquier nivel de abstracción, de hecho, los diagramas de flujo de datos pueden ser divididos en niveles que representen un mayor flujo de información y detalle funcional. Es importante señalar que el diagrama no proporciona ninguna indicación explícita de la secuencia de procesamiento, ésta puede estar implícita en el diagrama, pero la representación procedimental explícita generalmente queda pendiente hasta el diseño del software. El diagrama de flujo de control contiene los mismos procesos que el de datos, con la diferencia de que el primero muestra el flujo de control en lugar de datos.

El modelado del comportamiento es uno de los principios fundamentales de todos

los métodos de análisis de requisitos. El diagrama de transición de estados representa el comportamiento de un sistema que los muestra y los sucesos que hacen que éste cambie, además, indica qué acciones se llevan a cabo como consecuencia de un suceso determinado. Como definición podemos decir que un estado es un modo observable del comportamiento y cada uno representa un modo de comportamiento del sistema. Un diagrama de transición de estados indica cómo se mueve el sistema de uno a otro.

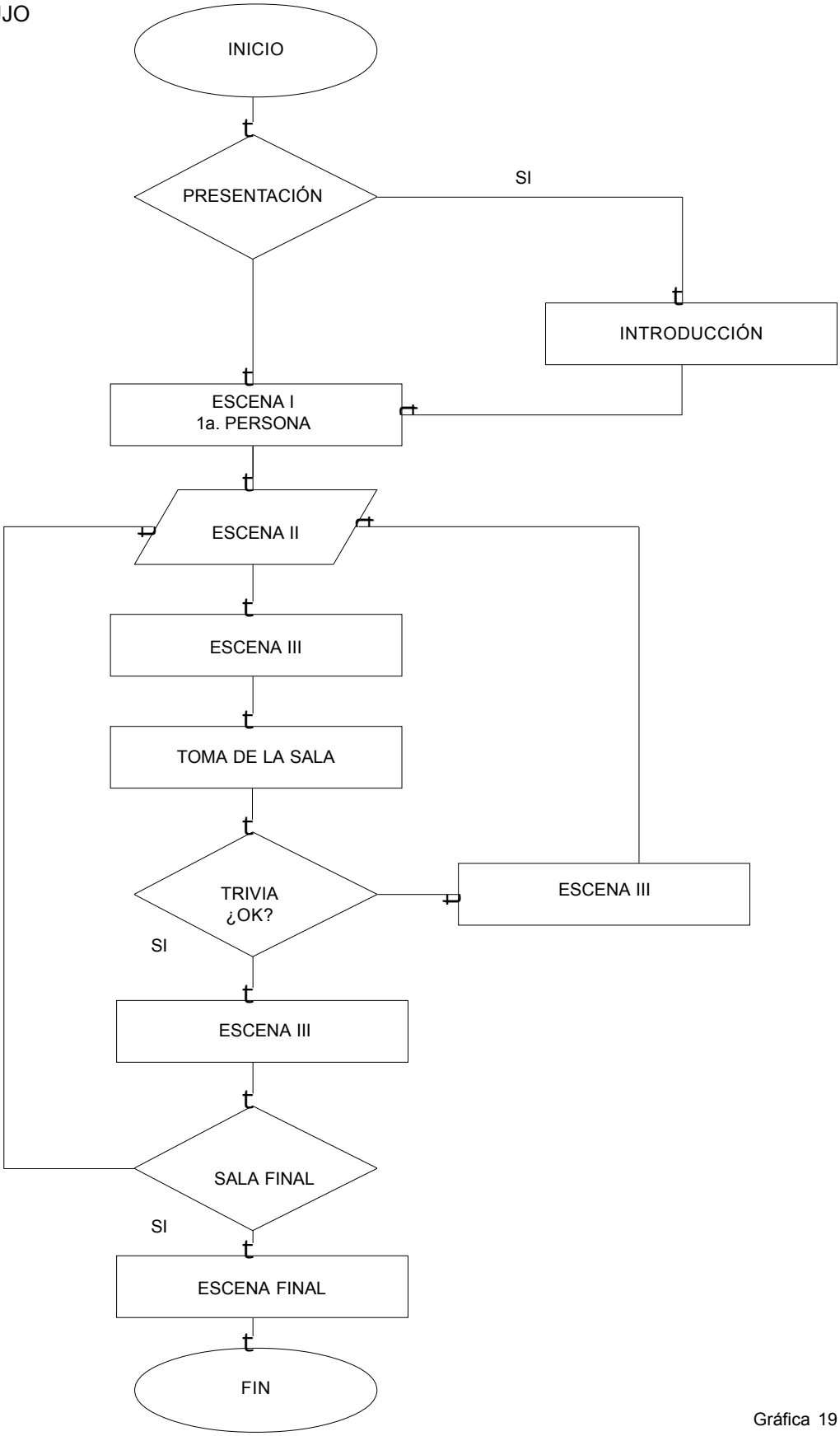
El modelo de análisis acompaña representaciones de objetos de datos, funciones y control. En cada representación las entidades y/o elementos de control desempeñan un papel importante, por consiguiente, es necesario proporcionar un enfoque organizado para representar las características de cada objeto de datos y elementos de control, esto se realiza con el diccionario de datos y éste se define como un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que se obtenga una misma comprensión de las entradas, salidas, componentes de almacenamiento y cálculos intermedios. Aunque el formato del diccionario varía, la información debe contener, al menos, el nombre, un alias, dónde y cómo se usa, una descripción e información adicional.

Una vez definidos todos los elementos que intervienen en el modelado estructurado del análisis, así como de las herramientas gráficas en que se basan, se definen todos los diagramas necesarios para culminar nuestra etapa de análisis del videojuego, incluyendo un sencillo diagrama de la máquina de videojuego o game engine de acuerdo a la relación (ver gráficas de la 18 a la 23), así como el diccionario de datos de la aplicación.



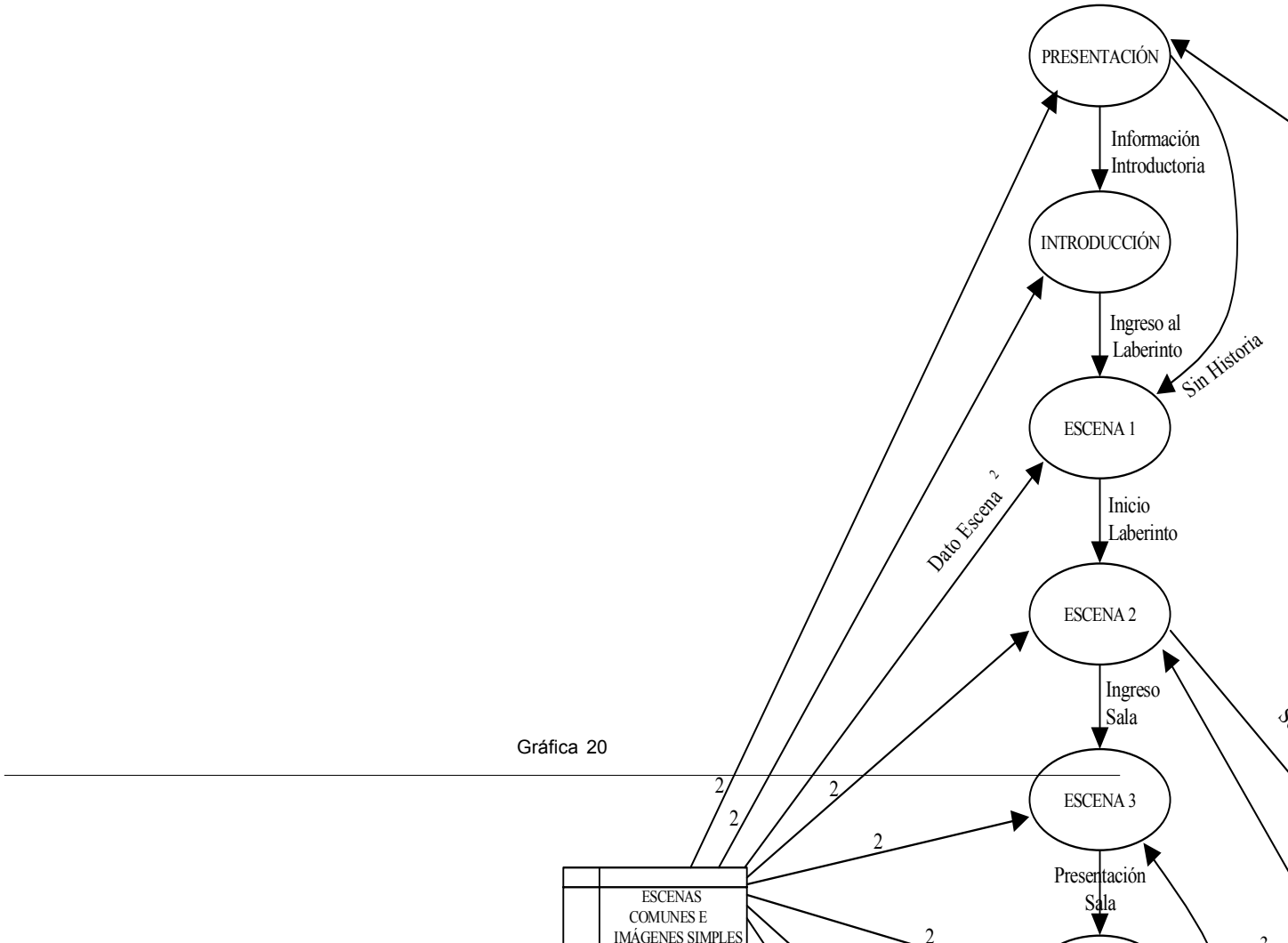
Gráfica 18

DIAGRAMA DE FLUJO



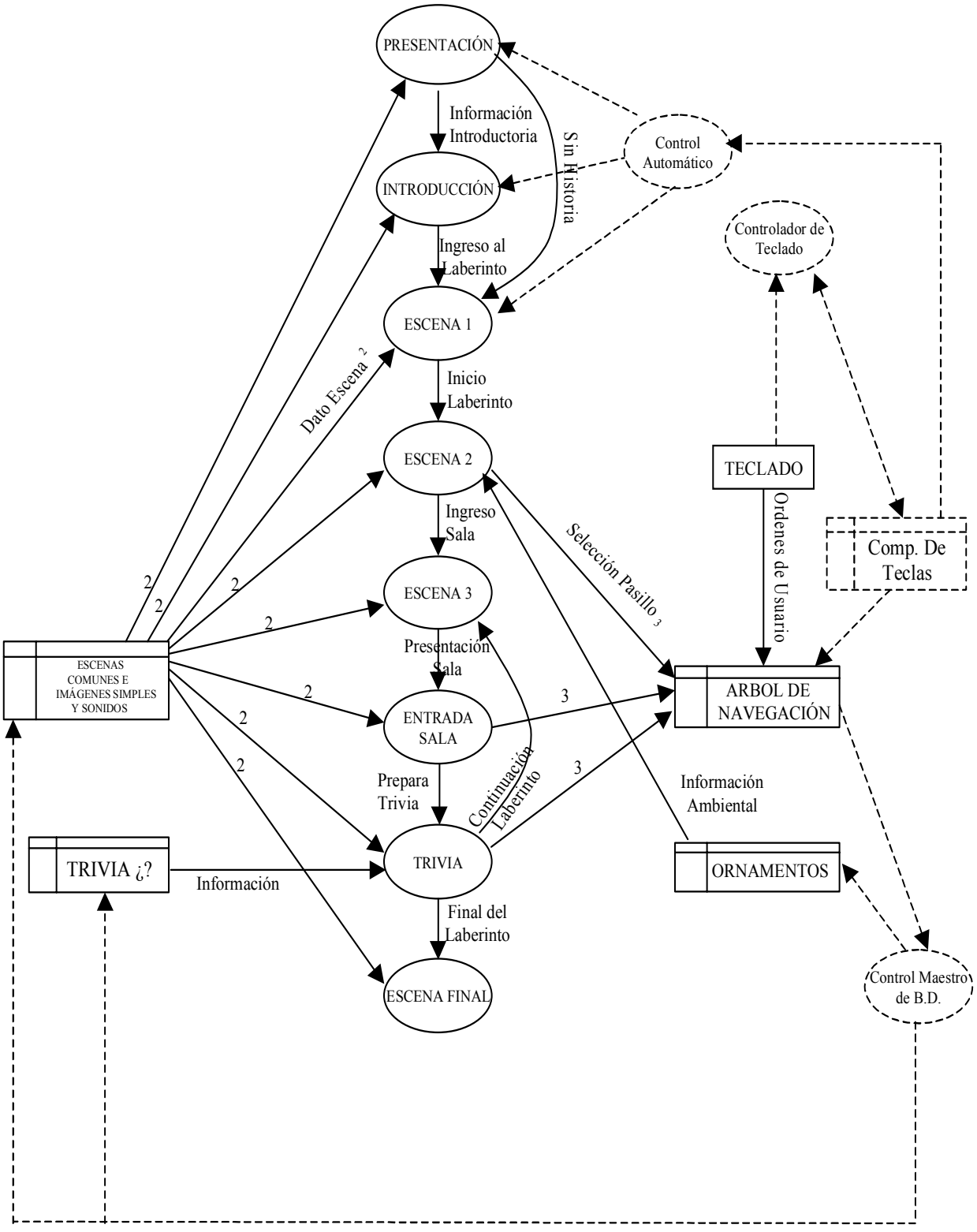
Gráfica 19

DIAGRAMA DE FLUJO DE DATOS



Gráfica 20

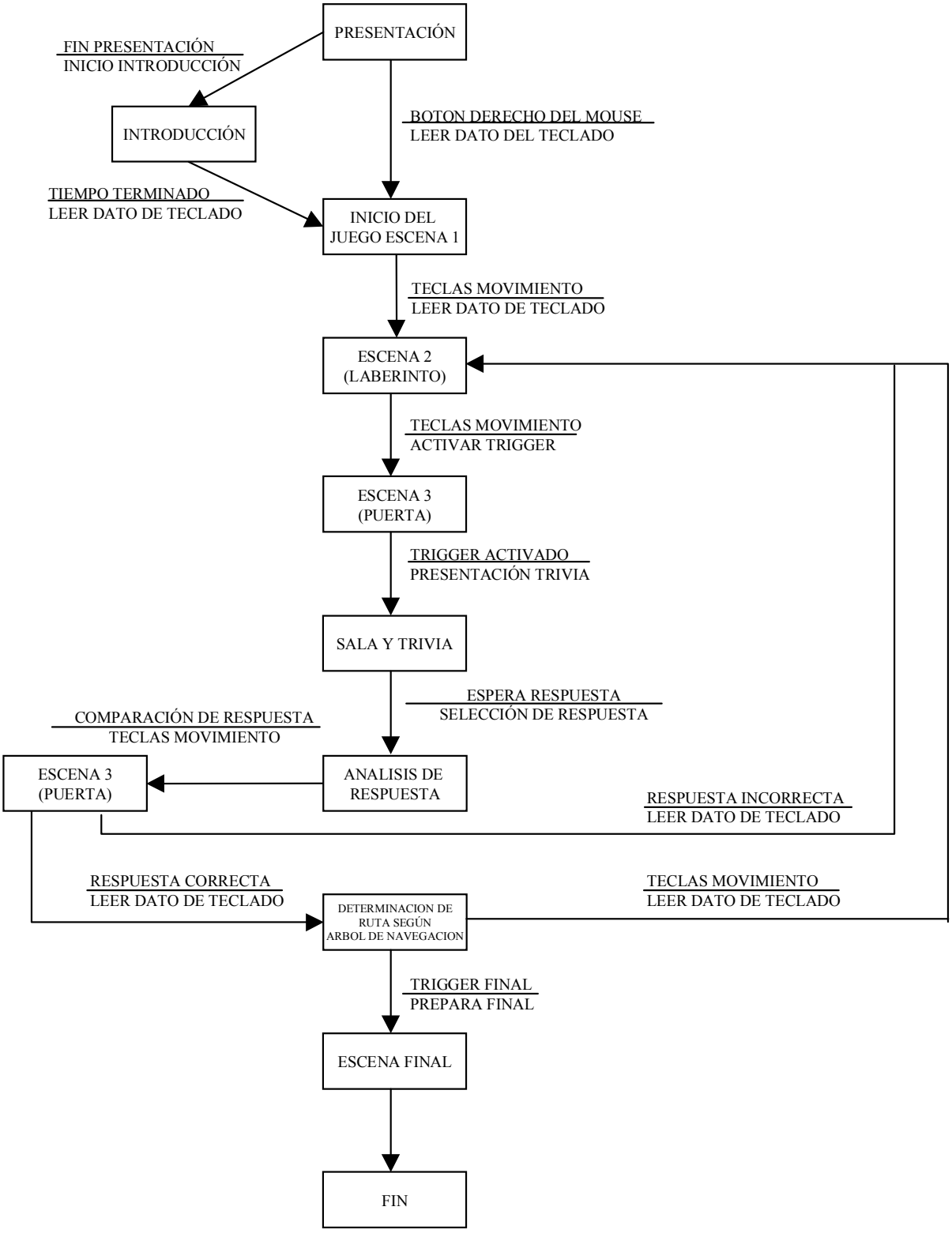
DIAGRAMA DE FLUJO DE CONTROL



CONTROL MAESTRO DE BD = GAME ENGINE

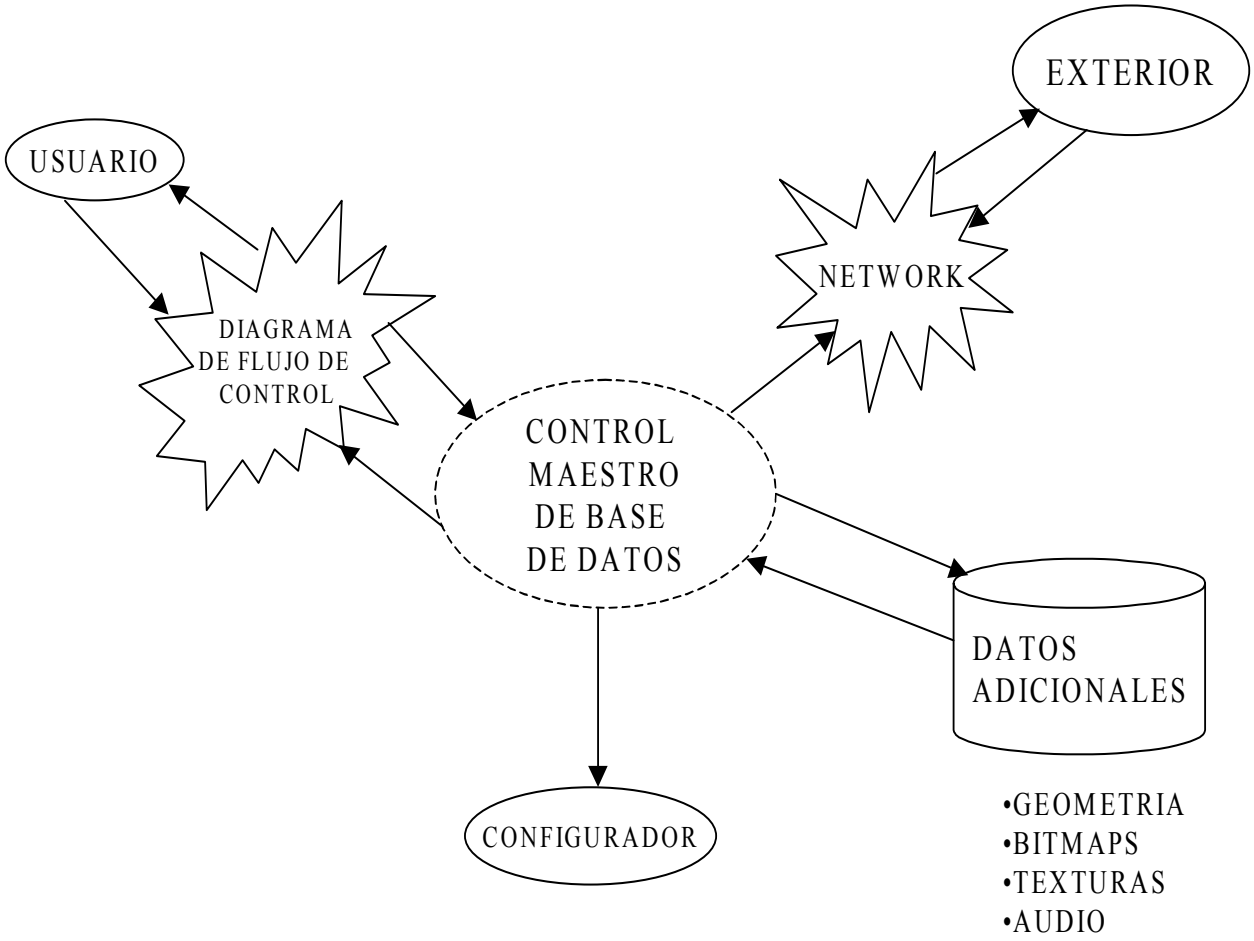
Gráfica 21

DIAGRAMA DE TRANSICIÓN DE ESTADOS



Gráfica 22

DIAGRAMA DEL CONTROL MAESTRO
GAME ENGINE (Teórico)



CONTROL MAESTRO DE BD = GAME ENGINE

CABE ACLARAR QUE ESTE DIAGRAMA TEORICO ES MUY GENERAL DE CÓMO FUNCIONA EL *GAME ENGINE* O MAQUINA DEL VIDEOJUEGO. SIN EMBARGO NO PODEMOS DEJAR DE MENCIONAR QUE EL *GAME ENGINE* TIENE A SU VEZ SUS CONTROLADORES NECESARIOS PARA LOS SONIDOS, LA MUSICA, LOS GRAFICOS Y LA INTERFACE DE RED.

PARA ESTE TRABAJO SE UTILIZA UNA MAQUINA DE VIDEOJUEGO DE DOMINIO PUBLICO Y, POR EL CARÁCTER DEL MISMO, LA INTERFACE DE RED NO ES EMPLEADA EN NUESTRO CASO.

DICCIONARIO DE DATOS

Las entidades que intervienen en el videojuego existen como archivos, como geometría de objetos, sonidos, etc. Y es el *game engine*, quien se encarga de relacionar estos archivos entre sí.

Por otro lado, las bases de datos no existen como tales, pero nos sirven para realizar un mejor análisis e identificar en que momento se debe presentar o ejecutar un sonido, objeto, imagen, etc.

1.- Control maestro (Game engine).- Manejador o máquina de videojuego encargado de sincronizar la interacción de información (datos) y funcionalidad de éste. Se encarga de generar y almacenar datos como la geometría, bitmaps, texturas, objetos y referencias a sonidos. Por las características de la aplicación se decidió usar el *Fly3d* ya que es de carácter público, es decir, abierto a cualquier usuario que lo requiera.

2.- Escenas comunes, imágenes simples y sonidos.- Zona encargada de almacenar la información general de las escenas, imágenes y sonidos que conforman la arquitectura del videojuego. El *game engine* es el encargado de relacionar los archivos de sonidos, video, imágenes simples, etc., en el momento en que se le indique. Cada objeto de los que interactúan en el juego son capaces de enviar un mensaje a otros de manera que al ser activado puede indicar la ejecución de un sonido, video, etc.

3.- Control de navegación.- Almacena la información de manera estructurada de todos los elementos que intervienen en el videojuego. Junto con el control maestro sincroniza al resto de las entidades para el correcto funcionamiento. El *game engine* usa el archivo donde se encuentra la geometría del ambiente.

4.- Ornamentos.- Área encargada de almacenar la información pertinente a la ambientación del laberinto; son archivos que el *game engine* toma para insertar los objetos en la posición indicada al momento de diseñar la geometría en el modelador. (3D Max Studio).

5.- Base de datos trivia.- Entidad encargada de almacenar la información de las *trivias* que darán o denegaran los accesos a cada una de las salas para participar en las actividades lúdicas o juegos.

a) El campo ID funciona como llave primaria a través de la cual tenemos acceso a otros campos.

b) El campo pregunta, es el texto de la pregunta que se hace antes de tener acceso a la sala.

b.1) Los campos opciones, presentan las posibles respuestas a la pregunta.

b.2) El campo respuesta contiene el valor de la opción correcta para contestar la pregunta.

6.- Presentación.- Video de introducción al juego donde se integra información ge-

neral como el título, los créditos, etc., y antecede a la historia.

7.- Historia.- Video que narra todo el ambiente histórico que rodea al juego; introduce al usuario con el personaje y explica la temática del desarrollo. Antecede a la Escena 1.

8.- Escena 1. Entrada.- Escena referente a la entrada al laberinto en primera persona sin interacción, es decir, no interviene dato alguno del teclado. Con esta escena se termina la falta de interacción con el usuario e inicia la aventura. Antecede a Escena 2.

9.- Escena 2. Laberinto.- Escena referente al desplazamiento a través del laberinto con el objetivo de llegar a alguna de las salas. Es una escena en primera persona con interacción de datos de movimiento del teclado para indicar el sentido del desplazamiento. Antecede a la Escena 3.

10.- Escena 3. Puerta.- Escena referente a la puerta de acceso a la sala respectiva, la escena es en primera persona con interacción de datos del teclado. Antecede a la escena de Sala y trivia.

11.- Sala y trivia.- Escenas diferentes de las salas asociadas con su trivia respectiva. Inmediatamente a la aparición de la sala se presenta la trivia con su respuesta en opción múltiple. Escena en primera persona con la intervención de datos del teclado para la solución a la pregunta.

12.- Escena final.- Video que marca el final del juego mostrando los créditos y datos generales.

CAPÍTULO IV

DISEÑO

ASPECTOS TEÓRICOS Y APLICACIÓN HACIA EL VIDEOJUEGO

Una vez realizada la instancia de análisis, la siguiente es la de diseño. Este proceso traduce los requisitos en una representación del software mediante la cual se puede evaluar la calidad integral antes de que comience la codificación o programación.

El diseño es el primer paso en la fase de desarrollo de cualquier sistema de ingeniería. Teóricamente podría describirse como el proceso de aplicar distintas técnicas y principios con el propósito de definir una aplicación con suficientes detalles que permitan su realización.

Dentro del paradigma propuesto en este trabajo, la etapa de diseño está muy ligada a la de análisis, es decir, cada uno de los elementos del modelado del segundo proporciona información necesaria que sustenta la creación del primero. De acuerdo con esto, se puede decir que el proceso de diseño es aquel en el que se desarrolla un modelo interviniendo directamente la intuición y los criterios basados en la experiencia, además de un conjunto de principios que delinearán la evolución del modelo manteniéndolos siempre en la más alta calidad.

El diseño del software se sitúa en el núcleo técnico del proceso de ingeniería de software, y es la primera de tres actividades técnicas (diseño, codificación y pruebas) necesarias para construir y verificar el software.

Cada una de estas actividades transforma la información de análisis de manera que se refleje en un producto final válido, en nuestro caso, un videojuego.

De acuerdo a los fundamentos teóricos de la ingeniería de software la fase de diseño se debe traducir en uno de datos, uno arquitectónico, de interfaz y un procedimental.

El diseño de datos transforma el modelo del dominio de la información, creado durante el análisis en las estructuras de datos necesarias para implementar el software, de acuerdo a esto tanto las entidades de datos como las relaciones con afectación directa sobre ellas definidas en el diagrama entidad-relación; y el contenido del diccionario de datos proporcionan la base suficiente para la estructuración de éstos.

El diseño arquitectónico define la relación entre los principales elementos estructurales del programa, es decir, para llevar acabo esta etapa se requiere de toda la recopilación de información necesaria de los diagramas de la etapa de análisis (flujo de datos, flujo de control, transición de estados, entidad-relación) además de un entendimiento de la interacción con sistemas externos. En nuestro caso, además de los diagramas indicados se debe tener en cuenta al jugador de acuerdo a la información tratada en el capítulo II.

El diseño de interfaz describe cómo se comunica el software consigo mismo, con los sistemas y con los operadores. Una interfaz implica un flujo de información como los datos y las variables de control, por lo tanto el diagrama de flujo de datos y de control proporcionan la información necesaria para el diseño de la interfaz.

El diseño procedimental transforma elementos estructurales de la arquitectura del programa en una descripción modular de los componentes del software. La información

que se obtiene en el diagrama de transición de estados sirve de base para este tipo de diseño.

En resumen, se considera que en el diseño se deben integrar todos los requisitos explícitos contenidos en el modelo de análisis y a su vez acondicionar todos los requisitos implícitos que generan el perfil del jugador.

DISEÑO DE DATOS

La importancia del diseño del software se puede decir con una sola palabra: *Calidad*.

Para el trabajo de diseño de cualquier aplicación, se deben tener en consideración factores de calidad externos e internos; los primeros son aquellas propiedades que pueden observar los usuarios como la velocidad, la fiabilidad, la variabilidad de funciones, etc., mientras que los internos son los que permiten al desarrollador la elaboración de alta calidad técnica.

Si bien teóricamente el proceso de diseño parece simple, en la práctica no es así; es un conjunto de pasos repetitivos que permiten al desarrollador describir todos los aspectos del software a construir. La capacidad creativa, la experiencia acumulada, el sentido del *buen* software y un empeño global en la calidad son factores críticos en el éxito del diseño.

Existen algunos principios básicos que permiten al desarrollador navegar a través del proceso en esta etapa, dentro de éstos podemos mencionar que se deben considerar enfoques alternativos durante su realización; no se debe reinventar nada, es decir, se fomenta la reutilización; el diseño tiene que presentar uniformidad e integración; debe valorar la calidad del diseño mientras se está creando (no después de terminarlo) y lo más importante, diseñar no es escribir códigos. Para lograr los factores antes mencionados y cumplir con los principios básicos del diseño se deben contemplar algunos aspectos, por ejemplo: Abstracción, refinamiento, modularidad, arquitectura del software, jerarquía de control o estructura del programa y estructura de datos.

La abstracción y el refinamiento son conceptos complementarios, la primera permite especificar datos y procedimientos, mientras que el segundo ayuda a revelar detalles a medida que progresa el diseño.

De igual forma, la arquitectura del software conlleva modularidad, es decir, se divide el software en componentes identificables y tratables por separado, éstos se conocen como *módulos*, los cuales están integrados para satisfacer los requisitos del programa (videojuego), además se agrega a la estructura del programa que es la encargada de éstos asociándoles una jerarquía.

La estructura de datos es una representación de la relación lógica entre los elementos individuales de datos.

Como la estructura de la información afecta invariablemente al diseño procedural final, la de datos es tan importante como la del programa en la representación de la arquitectura del software. La estructura de datos dicta las alternativas de organización, métodos de acceso, capacidad de asociación y procesamiento de la información.

Los conceptos fundamentales del diseño descritos arriba, sirven para incentivar diseños modulares. Un diseño modular reduce la complejidad, facilita los cambios (un aspecto crítico de la capacidad de mantenimiento del software: Capítulo VI), y hace más fácil

la implementación al fomentar el desarrollo en paralelo de diferentes partes del videojuego.

Existen tres conceptos dentro del diseño modular, sin embargo, la independencia funcional es fundamental ya que es un producto directo de la modularidad y de los conceptos del diseño.

La independencia funcional se consigue desarrollando módulos con una función única y con una falta de interacción hasta donde es posible con otros, es decir, se pretende diseñar un software o un videojuego de manera que cada módulo cumpla con una subfunción específica de los requisitos y tenga una sencilla interfaz cuando se *vea* con otras partes de la estructura del desarrollo. Los módulos independientes son más fáciles de probar (también de mantener), dado que los efectos secundarios causados por la modificación del diseño-código están limitados, la propagación de errores es reducida y se pueden reutilizar módulos.

En resumen, la independencia funcional es la clave para un buen diseño y éste es la clave de la calidad del software, además, la independencia se mide usando dos criterios cualitativos: Cohesión y acoplamiento. La primera es una medida de la fuerza relativa funcional de un módulo y el acoplamiento es una de interdependencia relativa entre los módulos.

Una vez desarrollada la estructura del programa, se puede conseguir una modularidad efectiva aplicando los conceptos de diseño definidos anteriormente.

Los principios y conceptos de diseño definidos establecen el fundamento para la creación del modelado que comprende representaciones de datos, arquitecturas, interfaces y procedimientos.

Al igual que en el modelo de análisis, cada una de las representaciones del modelado de diseño está atada a otras por lo que se les puede hacer un seguimiento hasta los requisitos del software.

El diseño es dirigido por la información. Los métodos de diseño del videojuego se obtienen del estudio de la información generada durante la etapa del modelo de análisis: El dominio de los datos, el funcional y el de comportamiento sirven de directriz para la creación del diseño.

El diseño de datos es la primera de las cuatro actividades de diseño que se llevan a cabo durante la ingeniería de software. El impacto de la estructura de datos, en la del programa, y la complejidad procedimental hace que el diseño de los datos tenga una profunda influencia en la calidad del software. El diseño de una estructura de datos eficaz debe tener en cuenta las operaciones que se lleven a cabo sobre ella.

La actividad principal del diseño de los datos es seleccionar representaciones lógicas de objetos de datos, comúnmente conocidas como *estructuras de datos.* El proceso de selección de éstas puede incluir un análisis de estructuras alternativas para determinar el diseño de forma más eficaz.

Como comentario podemos decir que los datos bien diseñados pueden conducir a una mejor estructura y modularidad de la aplicación, en nuestro caso el videojuego y, además, a una menor complejidad procedimental.

Si bien pueden existir principios que se deben tomar en cuenta para la especificación de los datos, en la práctica no necesariamente sucede ésto, la experiencia funge como factor primordial para la selección más adecuada de las estructuras de datos, a

pesar de ésto, no podemos dejar de mencionarlos: Las estructuras de datos y las operaciones sobre ellas deben estar claramente definidas, los principios de operación y el diccionario de datos establecidos durante la fase de análisis se deben aplicar y definir respectivamente; para la modularidad, las estructuras de datos sólo pueden ser *conocidas* por aquellos módulos que deben hacer uso directo sobre los informes contenidos dentro de las estructuras; se debe desarrollar una biblioteca de estructuras que complemente al diccionario de datos en donde se especifique todas y cada una de la operaciones que se les pueden aplicar con el simple objetivo de la reutilización y un diseño del software, y el lenguaje de programación que debe soportar la especificación y realización de los tipos abstractos de datos.

Estos principios forman una base para llevar a cabo un enfoque de diseño de datos que pueden integrarse en las fases de definición y desarrollo del proceso de elaboración del producto final y objetivo del presente trabajo, el videojuego apegado a los lineamientos dictados por la ingeniería de software.

DISEÑO DE DATOS DE LA APLICACIÓN

Los datos utilizados durante la ejecución de la aplicación tienen una estructura de cola, dado que la interacción entre el jugador y la computadora es continua llevándonos a que cada conjunto de datos que el jugador envíe a la aplicación, deberá proporcionar un nuevo estado, todo esto de acuerdo al recorrido que el jugador escoja en el árbol del videojuego que describe las posibles rutas del laberinto.

La cola es una colección ordenada de elementos de la que se pueden borrar dispositivos en un extremo llamado *frente* o insertarlos en el otro llamado *final de la cola.* El primer elemento insertado en una cola es el primero en ser eliminado, por esta razón algunas veces las colas son denominadas como *Listas FIFO* (First Input First Output).

La cola de datos esta conformada por estructuras que pasan a través de los diferentes niveles de programación del videojuego, los cuales nos dan la información necesaria para obtener el recorrido, las geometrías, los sonidos, posición y/o movimientos, etc., del nuevo estado de la aplicación, lo cual estimula la reacción del jugador. Además se debe recordar que la correcta estructura de datos es eficaz al tener en cuenta las operaciones que se llevan a cabo sobre ella.

Cabe mencionar que el *game engine* es el encargado de mantener la relación entre los datos enviados por el jugador y los estados de la aplicación, todo esto para el correcto desarrollo del videojuego. Además debemos aclarar que no se hace un estudio más profundo acerca de las estructuras de datos que utiliza un game engine, ya que por su naturaleza sería un tópico muy extenso, propio de un trabajo de tesis completo, por lo que su definición es meramente teórica acerca de su funcionamiento.

DISEÑO DE ARQUITECTURA

Después del diseño de datos continúa el diseño arquitectónico del software. Su objetivo primordial es el de desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos, asimismo combina la estructura del programa y las de datos, definiendo interfaces que permiten el flujo de éstos a través del videojuego.

El diseño arquitectónico tiene sus orígenes en antiguos conceptos de diseño que

sustentan la modularidad, el diseño descendente o en cascada y la programación estructurada.

Es importante resaltar que la simplicidad estructural a menudo refleja eficacia. La optimización del diseño arquitectónico debe conseguir el menor número de módulos, tantos como sean necesarios y que la estructura de datos sea menos compleja para satisfacer los requisitos de información.

El diseño orientado al flujo de datos es compatible con un amplio rango de áreas de aplicación, de hecho, como todo el software (incluyendo los videojuegos) puede representarse como un diagrama de flujo de datos, un método de diseño que haga uso del diagrama podría, teóricamente, aplicarse en todos los desarrollos de aplicaciones. Un enfoque orientado al flujo de datos para diseñar es particularmente útil cuando se procesa secuencialmente la información y no existe ninguna estructura jerárquica formal.

El diseño orientado al flujo de datos es un método de diseño arquitectónico que permite una cómoda transición desde el modelo de análisis a una descripción del diseño de la estructura del programa. La transición desde el flujo de información (representado como un diagrama de flujo de datos) se lleva a cabo de una forma muy sencilla.

En resumen podemos decir que el diseño arquitectónico proporciona una imagen de la estructura del programa, sin embargo, desde el punto de vista sistemático debe tener una interacción con su entorno, por ello aparece una tercera etapa del diseño: La de interfaz.

A continuación como parte de la etapa de aplicación se presenta el diagrama que resume el diseño arquitectónico del videojuego.

Como antecedente, el diagrama que representa el diseño arquitectónico (ver gráfica 24 en la siguiente página) es una versión general sustentada por los diagramas vistos en el capítulo anterior.

DISEÑO DE INTERFACES

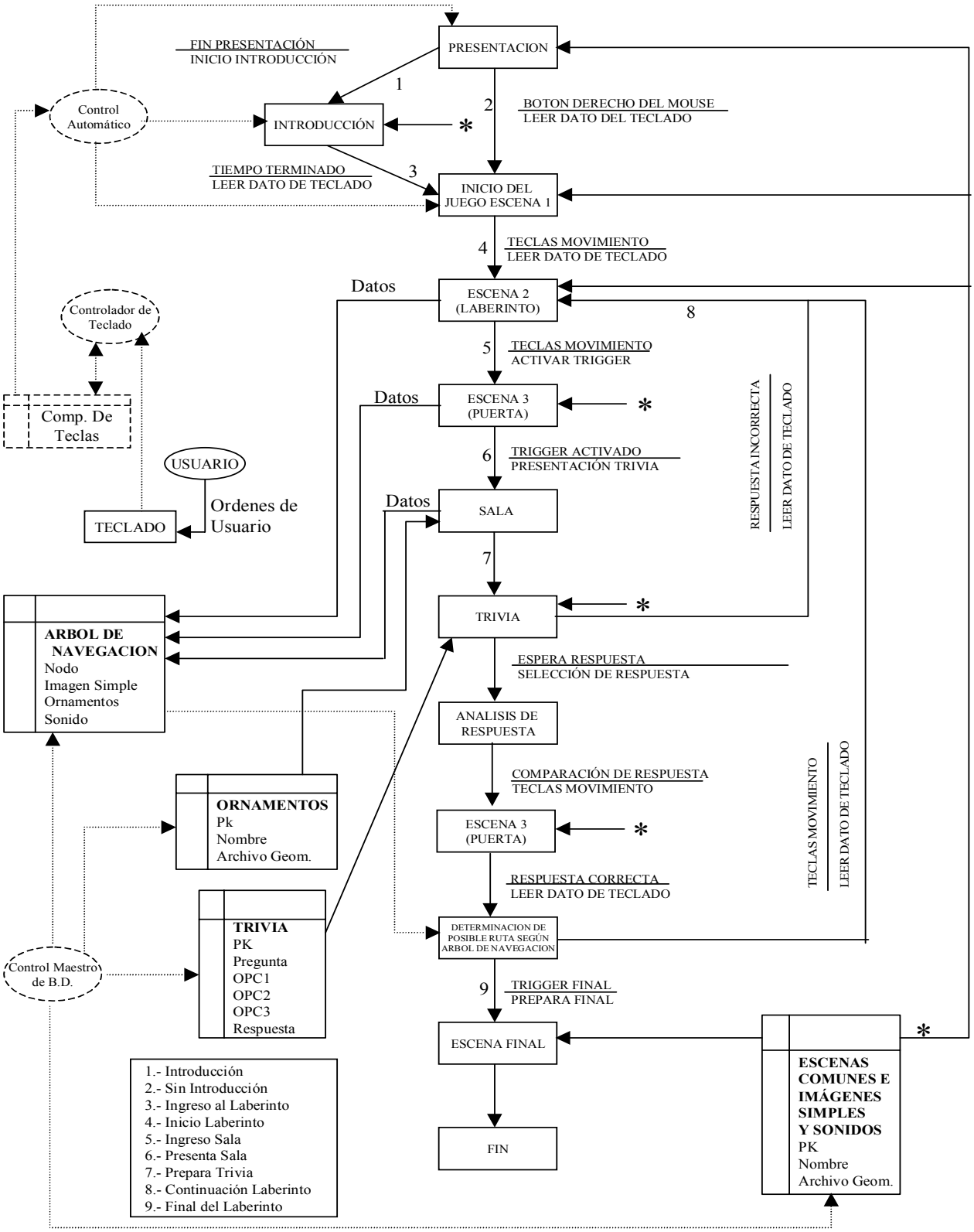
El diseño de la interfaz se concentra en tres áreas importantes: Diseño de interfaces entre los módulos que componen al videojuego; diseño de interfaces entre la aplicación de manera global, otros productores y consumidores no humanos de información y, finalmente, el diseño de interfaz entre el usuario y la computadora.

El diseño de las interfaces internas de la aplicación, comúnmente conocidas como interfaces intermodulares, depende directamente de los datos que fluyen dentro del sistema mismo y además de las características propias del lenguaje de programación (modelador y máquina del videojuego o *game engine*).

El modelo de análisis contiene mucha información requerida para el diseño de interfaces internas. El diagrama de flujo de datos describe cómo se transforman los objetos de datos o entidades al moverse a través del software. Las transformaciones del diagrama de flujo de datos se convierten en módulos dentro de la estructura del programa, por lo tanto los objetos de datos que fluyen en cada uno de éstos deben convertirse en un diseño para la interfaz intermodular que corresponda a esa transformación.

El diseño de la interfaz externa empieza con una evaluación de cada entidad externa que interactúa con el sistema mismo (videojuego). Se determinan los requisitos de datos y de control de las entidades externas y se diseñan las interfaces más apropiadas. En el caso de los videojuegos el diseño de interfaces externas es de manera muy compleja ya

DIAGRAMA DE DISEÑO ARQUITECTÓNICO



Gráfica 24

que además de entender las entidades externas desde el punto de vista lógico para la interacción con el software, se debe realizar un estudio desde el punto de vista físico de su funcionamiento con el simple objetivo de darle mayor realidad al momento de interactuar con el videojuego.

Ambos diseños de interfaces, externas e internas deben acoplarse con algoritmos de validación de datos y corrección de errores dentro de cada módulo. Como efectos secundarios que se propagan a través de las interfaces del programa, es esencial comprobar todo el flujo de datos de módulo a módulo y con el contexto exterior para asegurarse de que los datos se ajustan a los límites establecidos durante el análisis de requisitos.

El diseño de la interfaz de usuario tiene tanto que ver con el estudio de las personas o usuarios (encuesta del ILCE, capítulo II) como con los aspectos de la tecnología.

El proceso general para diseñar la interfaz de usuario empieza con la creación de diferentes modelos de la aplicación tal como se espera que se perciba desde afuera. Se definen las tareas orientadas al jugador y a la máquina requeridas para conseguir la función del sistema, se consideran los aspectos del diseño aplicables a todos los demás de la interfaz y, finalmente, se evalúa la calidad del resultado.

Para obtener un mejor resultado, el diseño se debe realizar en conjunto con la concepción que tiene el usuario, a modo de que la interrelación refleje fielmente la información sintáctica y semántica.

Para el diseño de interfaces de un videojuego, un papel de gran importancia es el mapeado de texturas que se le da a los escenarios en su conjunto, además del que se aplica a cada uno de los ornamentos y personajes que intervienen en el desarrollo final. El diseño de interfaces debe hacerse con un toque artístico con el objetivo de dar mayor realismo y credibilidad al juego.

Teóricamente existen cuatro diferentes modelos o concepciones que entran en juego cuando hay que diseñar una interfaz con el usuario. Se crea un modelo de diseño, el desarrollador establece un modelo de usuario, éste desarrolla un perfil mental y finalmente los creadores llevan a cabo una imagen del sistema. Para los videojuegos, en la mayoría de los casos, esto no sucede, los desarrolladores y creadores artísticos tienen la gran tarea de diseñar las interfaces desde el punto de vista creador-usuario; esto debido a la gran cantidad de estratos de usuarios o jugadores que pueden tener acceso a un videojuego, por lo que es difícil orientar el diseño hacia una parte social en particular. De acuerdo a ésto es que el diseño de interfaces de un videojuego se encamina hacia el realismo que puede tener desde el punto de vista ambiental y de comportamiento.

A medida que evoluciona el diseño de la interfaz del usuario, emergen casi siempre cuatro aspectos comunes del diseño: El tiempo de respuesta de la aplicación, las facilidades de ayuda al usuario, la manipulación de información de errores y el etiquetado de órdenes. En el caso de los videojuegos, un aspecto fundamental es el tiempo de respuesta que debe tener cada evento que se presenta durante la navegación. Se debe tener en cuenta dos características importantes: Duración y variabilidad.

Si la duración del tiempo de respuesta es demasiado largo, el resultado inevitable es la frustración y estrés del usuario, pero un tiempo de respuesta demasiado corto puede ser también perjudicial si la interfaz apresura al jugador. La variabilidad se refiere a la desviación del tiempo medio de respuesta, y en muchos aspectos, es la característica más impor-

tante; una variabilidad pequeña permite al usuario establecer un ritmo, incluso, si el tiempo de respuesta es demasiado largo.

El diseño de las interfaces se basa profundamente en la experiencia del diseñador y en las anécdotas recogidas en cientos de documentos técnicos y docenas de libros. Muchas fuentes de literatura, sobre todo de diseño, presentan un conjunto de directrices del diseño de interfaces que consiguen llevarlas a cabo de manera eficaz y amigable.

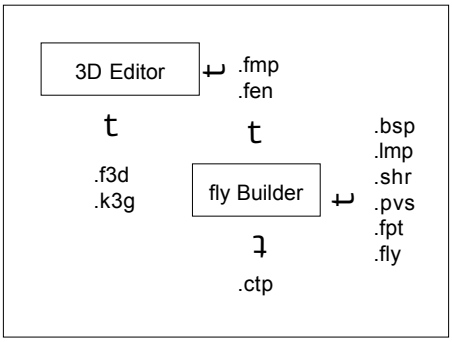
Algunas de las directrices más importantes son, en la parte de interacción, ser consistentes; buscar la eficiencia en el diálogo, el movimiento y el pensamiento; categorizar las actividades por función y organizar la pantalla de acuerdo a esto. En la parte de visualización de la información; que sea relevante en el contexto actual; no abrumar al usuario con demasiados datos; utilizar formatos que permitan una rápida asimilación de la información, permitir al jugador mantener el contexto visual; estudiar la geografía de la pantalla y usarla eficientemente; minimizar el número de acciones de entrada de datos que necesita realizar el usuario; mantener la consistencia entre la visualización y la introducción de datos; dejar al usuario controlar el flujo interactivo y, finalmente, eliminar entradas innecesarias.

DISEÑO DE INTERFACES DE LA APLICACIÓN

Como se ha mencionado se tiene una clasificación de las interfaces de acuerdo a la naturaleza con la que actúan: Internas y externas.

1.- Interfaces Internas.

A continuación se presenta un esquema en el que se indican los archivos que contienen la información que la máquina de videojuego procesa para la simulación o creación final (ver gráfica 25). El programa 3D Max Studio funciona como editor para generar los archivos (fmp, fen, f3d, k3g). Como parte de la aplicación de las interfaces de hace una breve explicación de cada uno de estos archivos:



Gráfica 25

FMP.- Contiene la información básica del nivel del videojuego. Además las coordenadas en tres dimensiones de todos los vértices y caras que componen la geometría del nivel, es el archivo más importante usado por el *fly Builder* en el proceso de construcción.

FEN.- Contiene la información de todos los objetos que existen en el nivel, así como de su posición o atributos especiales. Estos objetos pueden ser luces, puertas, objetos ambientales, etc.

BSP.- Contiene la información de los nodos de la geometría. Los datos dentro de él están divididos en nodos, donde cada uno puede ser interior (definiendo un portal que divide el espacio entre el positivo y negativo) o un hijo (que define un volumen convexo). Este archivo es generado por el *fly Builder* y el *game engine* lo carga antes de comenzar la simulación.

LMP.- Contiene el mapa de luces. Lo genera el *fly Builder* y lo carga el game engine al iniciar la escena para presentar la iluminación estática o de ambientación del nivel.

SHR.- La información de los *shaders* es almacenada en este archivo. En él se guardan los pasos y variables de cada uno. Los archivos de este tipo son generados al exportar cualquier objeto del modelador en tres dimensiones y el *game engine* lo carga cuando el objeto o una escena tiene asociado un *shader*. Se define shader como acabado, textura o efecto que se le asigna a una geometría.

PVS.- Contiene la información de la visibilidad que existe entre los diferentes nodos que conforman la geometría del nivel, la cual está almacenada en el archivo BSP. Este se carga una vez que se inicializó la escena correspondiente.

FPT.- Contiene la información de los portales. En este contexto los portales son polígonos que se relacionan de par en par con volúmenes convexos (almacenados en el archivo BSP). Esta información es utilizada por el algoritmo, indicando el camino por el cual se puede o no desplazar el jugador. Este archivo es cargado al mismo tiempo que el BSP para la simulación.

FLY.- Se encarga de la información relacionada con la escena. Se incluyen los valores de las variables del *game engine* desde información global hasta información específica del plugin. Cada entidad o elemento que se presenta en la escena, debe estar incluido en este archivo con todos sus parámetros. El archivo .fly es generado por el *fly Builder* tomando la información de los archivos .fen y el .ctp.

F3D.- Este es uno de los dos tipos de archivos que representan objetos en el *fly3D.* Los objetos definidos en este tipo de archivo son estáticos o su animación se basa única y exclusivamente en la modificación de sus vértices, caras o posición en cada uno de los cuadros de animación.

Este tipo de archivo se activa cuando el game engine carga el objeto correspondiente.

K3D.- Este es el segundo tipo de archivo de objetos en el *fly3D,* sin embargo representa objetos animados a través de una estructura sobre la cual se monta una textura o geometría para generar el movimient;o al igual que el F3D, éste se genera con un modelador de 3D (Autocad, por ejemplo).

CTP.- Archivo que enumera las entidades que son insertadas en la escena cuando se modela. Es usado para relacionar los objetos adicionales a la escena, con el tipo de objeto definido en los plugins. Este se debe generar manualmente y sólo se utiliza al momento de construir la escena.

2.- Interfaces Externas.

INTRODUCCIÓN

PRESENTACIÓN DE LA HISTORIA

En la presentación se muestran los créditos correspondientes al proyecto, así como la historia y temática del videojuego. Los créditos proporcionan los nombres del asesor y de los integrantes del equipo; así como el escudo de la UNAM, además datos necesarios de créditos, también se presenta una pequeña reseña de la historia la cual consta de un mapa ficticio del mundo de la imaginación en donde se desarrollará la historia. El texto se desplaza a través del mapa de modo vertical.

INICIO DEL JUEGO

ESCENA 1

(Puerta inicial)

Esta escena es el principio del video, al igual que todo el juego tiene una vista en primera persona, en donde al abrir la puerta se presenta un pasillo en donde el jugador podrá escoger un camino para continuar a través del laberinto y de esta manera encontrar la entrada a una sala.

ESCENA 2

(Laberinto o pasillo)

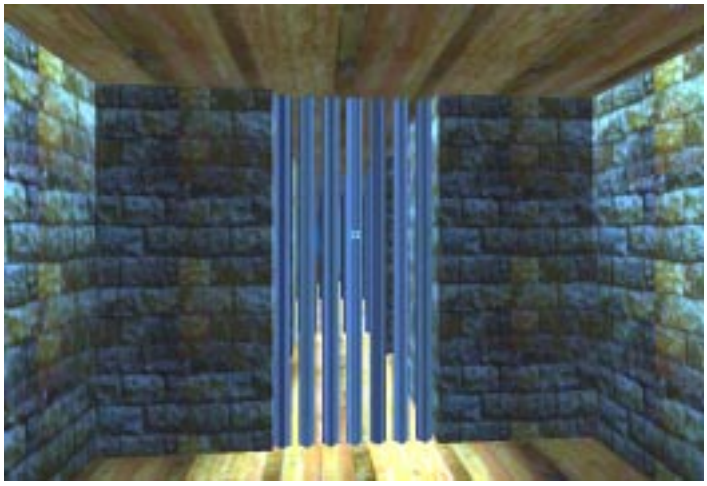
Al momento de pasar la primera puerta se observan las opciones que el usuario tiene para continuar en el laberinto; los pasillos que conducen a todas las salas tienen la misma textura, esto es para dificultar que el jugador tenga una referencia de ubicación dentro del mismo.

La ruta de navegación en el videojuego depende en gran medida del camino que se determine, éstas están fijas pero su recorrido lo establece el jugador ya que él, en cualquier momento, podrá cambiar de dirección, logrando con esto que los caminos se multipliquen.

ESCENA 3

(Puerta de entrada a la sala)

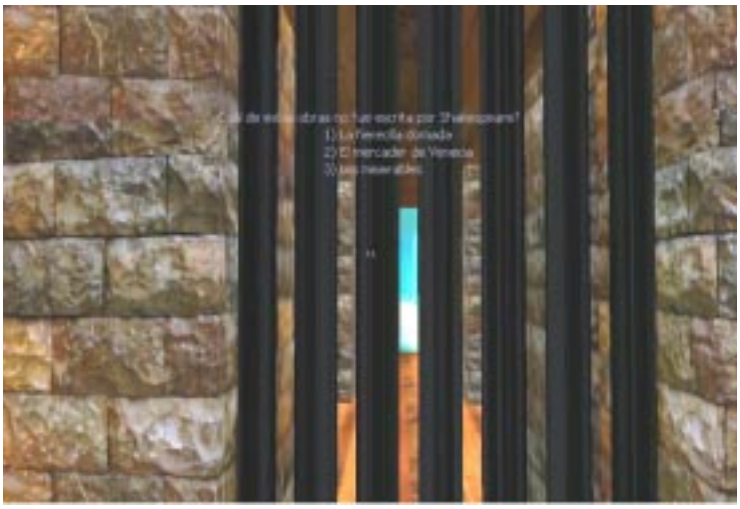
Una vez que el jugador ha determinado el rumbo a seguir dentro del laberinto, se presenta una nueva puerta la que dará acceso a una sala. La puerta siempre se abrirá ya sea para salir o entrar a la sala.



SALA Y TRIVIA

(Pregunta)

Una vez dentro el jugador se desplazará hacia la salida de la misma, en ese momento se presentará la trivia que debe responder para seguir su camino.



ANÁLISIS DE RESPUESTA

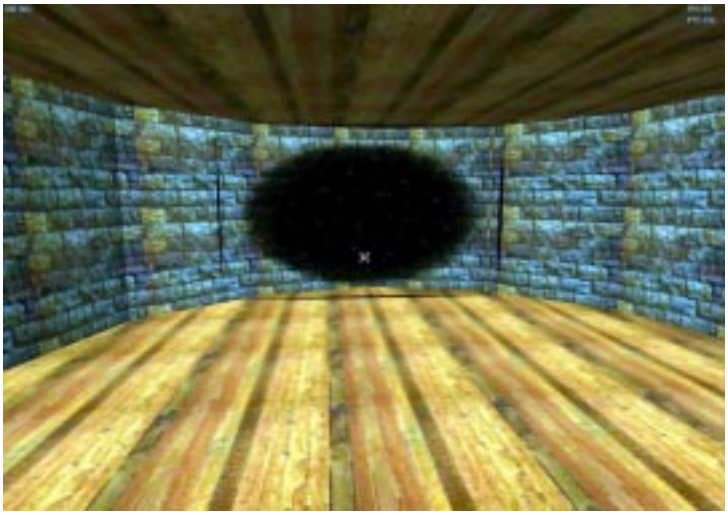
(Puerta de salida)

Una vez que se toma una opción de las posibles, el game engine definirá y mandará un mensaje, si la respuesta es correcta abrirá la puerta de salida, en caso contrario sólo quedará la opción de salir por la primera puerta.

ESCENA FINAL

(Fin del juego)

En este punto se presentará la animación correspondiente al final del videojuego.



DISEÑO PROCEDURAL

Este diseño se realiza después del de datos, arquitectónico y de interfaz. La especificación procedural necesaria para definir los detalles de los algoritmos se expresan en lenguaje sencillo y entendible. El diseño debe especificar los detalles sin ambigüedades. Con un lenguaje simple se puede escribir un conjunto de procesos procedimentales de diversas formas.

Los fundamentos de este tipo de diseño se forjaron con el uso de un conjunto de construcciones lógicas con las que podrían formarse cualquier programa. Estas construcciones hacían hincapié en el mantenimiento del dominio funcional, es decir, todas las construcciones tenían una estructura lógica predecible, por ejemplo: La secuencia, la condición y la repetición. La secuencia implementa los pasos de procesamiento esenciales en la especificación de un algoritmo; la condición proporciona los medios para seleccionar el procesamiento basándose en alguna ocurrencia lógica, y la repetición proporciona los *bucles* o *ciclos*. Estas tres construcciones son fundamentales para la programación estructurada, una importante técnica de diseño procedimental.

Las construcciones lógicas se propusieron para limitar el diseño procedimental del software a un pequeño número predecible de operaciones.

Cabe indicar que la métrica de la complejidad indica que el uso de construcciones estructuradas reduce la complejidad del sistema y por lo tanto mejora su comprensión, pruebas y mantenimiento.

Cualquier programa, independientemente del área de aplicación o complejidad técnica, se puede diseñar o implementar usando solamente las tres construcciones estructuradas.

Es incuestionable que las herramientas gráficas proporcionan excelentes formas gráficas que describen muy bien el detalle procedimental. El diagrama de flujo es la represen-

tación gráfica más conocida y utilizada en este tipo de diseño, sin embargo, si se emplea o diseña incorrectamente las herramientas gráficas puede traer como consecuencia un software erróneo. Otra herramienta que se aplica en el diseño procedimental es el lenguaje de diseño de programa, también conocido como lenguaje estructurado o pseudocódigo. Esta herramienta es un lenguaje rudimentario que utiliza vocabulario de un idioma y la sintaxis de uno estructurado de programación.

Independientemente del origen, un lenguaje de diseño debe tener las siguientes características: Una sintaxis fija de palabras claves para todas las construcciones estructuradas, declaraciones de datos y características de modularidad; sintaxis libre que describa las características del procesamiento; facilidad para la declaración de datos que deben incluir las estructuras de datos más simples y las más complejas y finalmente, la definición de subprogramas o técnicas que soporten varios modos de descripción de interfaz.

Una sintaxis básica debe incluir construcciones para la definición de subprogramas, descripción de interfaz, declaración de datos, técnicas para la estructuración de bloques, construcciones de condición, repetitivas y de entrada-salida. Hay que resaltar que la sintaxis se puede ampliar de manera que incluya palabras claves para multitareas y/o procesamiento concurrente, manipulación de interrupciones, sincronización de procesos, etc. Además de estas dos formas de representación, existen otras herramientas para el diseño procedimental tales como las tablas de posición o los diagramas de *N-S* (Nassi-Shneiderman), sin embargo, dependen de cada diseñador de acuerdo a la experiencia que tiene.

Dentro del análisis se presenta el diagrama de flujo (ver gráfica 19), el cual nos es útil para explicar con mayor precisión el diseño procedural de esta tesis.

A continuación, como parte de la etapa de aplicación, se hace una detallada explicación de dicho diagrama.

Como antecedente, la siguiente descripción del diagrama de flujo presenta la navegación a través del videojuego, sin embargo, la de las escenas se hace de manera breve, por lo que se debe consultar el diccionario de datos.

De acuerdo al diagrama de flujo de datos y al diccionario de datos (DD), el diagrama de flujo que representa el diseño procedural del presente trabajo se inicia con la escena de presentación (punto 6 del DD); de forma automática continúa la introducción, historia y temática (punto 7 del DD) con la ventaja de poderse interrumpir a decisión del jugador.

Una vez transcurridas estas dos instancias prosigue la escena I (punto 8 del DD), que prácticamente inicia la navegación; posteriormente la escena II (punto 9 del DD), referente al desplazamiento a través del laberinto.

De acuerdo a la libre selección por parte del usuario acerca de la sala a la que quiere ingresar, sigue la escena III (punto 10 del DD) que presenta el ingreso a la misma y que a su vez antecede a la escena sala y trivia (punto 11 del DD). Una vez intentada superar la trivia, nuevamente aparece la escena III para representar el abandono de la sala.

La navegación de desplazamiento (escena II), ingreso a la sala seleccionada (escena III), realización de la trivia con éxito o no (escena sala y trivia) y abandono de la sala (escena III) continúa de manera sucesiva hasta encontrar la sala marcada como final para que, una vez superada la prueba, continúa la escena final (punto 12 del DD) que marca el fin del videojuego.

CAPÍTULO V

DESARROLLO

(Programación)

Dentro de este capítulo se desarrollará el producto final fundamentado en todas las etapas vistas durante la fase de análisis; además, se complementa con los elementos del diseño.

En esta etapa se llevarán acciones que facilitarán la programación de una manera ágil y dinámica.

Hay que mencionar que de acuerdo a la forma que se viene trabajando, en cada uno de los capítulos anteriores (III y IV) se incluye como parte de éstos la información y características concernientes a la aplicación y que de alguna manera se reflejarán en el objetivo final de este trabajo de titulación, el videojuego mismo.

Se debe aclarar que para este capítulo una parte, como la comentada en el párrafo anterior, no existe, ya que tanto la codificación como las pruebas influyen de manera directa en la fabricación y correcto funcionamiento del producto final. En resumen podemos decir que la parte de la aplicación es el videojuego como tal, sin embargo, al final del capítulo, se incluye una bitácora de actividades que explican las etapas primordiales en la programación del videojuego.

CODIFICACIÓN

De acuerdo al capítulo anterior, el diseño se debe traducir en una forma legible por la máquina. La etapa de generación o codificación lleva a cabo esa tarea. Hay que recordar que si la etapa de diseño se realiza de una manera detallada y apegada a los lineamientos de calidad, la realización del código se ejecuta mecánicamente.

Se ha mencionado que el paradigma o metodología *Top down – bottom up* se lleva a cabo de forma lineal con la peculiar característica que permite retomar en cualquier momento etapas anteriores a fin de corregir procesos con el simple objetivo de que se refleje en la codificación.

Dentro de las nuevas tendencias en la codificación de software, una parte importante es la programación orientada a objetos cuya principal característica es la reutilización de los elementos que lo estructuran. Los videojuegos como parte de esta tecnología no son la excepción.

La reutilización es una característica importante para un componente de calidad orientado a objetos. Cualquier estudio sobre recursos de software estaría incompleto sin estudiar la reutilización, esto es, la creación y reuso de bloques de construcción de software. Tales bloques deben establecerse en catálogos para una consulta más adecuada, estandarizarse para una fácil aplicación y validarse para una también fácil integración. En resumen podemos decir que la reutilización se basa en estructurar componentes de programas ya existentes cuando es posible. Existen cuatro categorías de recursos de software: Componentes ya desarrollados (el software nuevo o ampliaciones, se pueden crear tomando como base componentes de proyectos anteriores e incluso propios); ya experimentados (las especificaciones, diseños, código o datos de prueba ya existentes y desarrollados para proyectos anteriores); con experiencia parcial (las especificaciones, diseños, código o datos de prueba ya existentes y desarrollados para proyectos anteriores,

pero que requieren una modificación sustancial) y finalmente, componentes nuevos (los componentes de desarrollo que se deben construir específicamente para las necesidades actuales). Esta categoría de recursos está asociada directamente al mantenimiento del software y cuya implicación se verá a detalle en el siguiente capítulo.

La programación orientada a objetos es una nueva manera de enfocar la programación. Desde sus comienzos ésta ha estado gobernada por varias metodologías.

En cada punto crítico de la evolución de la programación se crea un nuevo enfoque para ayudar al desarrollador a programar un software cada vez más complejo.

Las metodologías han pasado desde el cambio de conmutadores del panel frontal de las computadoras, lenguajes ensambladores, lenguajes de alto nivel, estructurados y programación orientada a objetos.

Aunque la programación estructurada nos ha llevado a excelentes resultados cuando se ha aplicado a programas moderadamente complejos, llega a fallar en algún punto cuando el programa alcanza un cierto tamaño. Para poder escribir programas de mayor complejidad se necesitaba un nuevo enfoque en la tarea de programación; a partir de ese punto se inventa la *programación orientada a objetos* (POO) y toma las mejores ideas incorporadas en la programación estructurada y las combina con nuevos y potentes conceptos que permiten organizar los programas de forma más efectiva.

La POO permite descomponer un problema en subgrupos relacionados; cada uno pasa a ser un objeto autocontenido que integra sus propias instrucciones y datos que le relacionan con ese objeto; de esa manera la complejidad se reduce y el desarrollador puede tratar programas más largos.

Todos los lenguajes de POO, incluyendo C++ (lenguaje bajo el que trabaja el *game engine fly3D*), comparten tres características: Encapsulación, polimorfismo y herencia. Analicemos estos conceptos.

1.- Encapsulación.- Es el mecanismo que agrupa el código y los datos que maneja y los mantiene protegidos frente a cualquier interferencia o mal uso. En un lenguaje orientado a objetos; el código y los datos pueden empaquetarse de la misma forma en que se crea una *caja negra* autocontenida. Dentro de ésta son necesarios, tanto el código como los datos; cuando éstos están enlazados se crea un objeto, en otras palabras, un objeto es el dispositivo que soporta la encapsulación.

En un objeto, los datos y el código, o ambos, pueden ser privados o públicos. Los datos o el código privado sólo los conoce y son accesibles a otra parte del objeto, es decir, una parte del programa que está fuera del objeto no puede acceder al código o a los datos privados. Cuando el código o los datos son públicos, otras partes del programa pueden acceder a ellos, incluso, aunque esté definido dentro de un objeto. Normalmente, las partes públicas se utilizan para proporcionar una interfaz controlada a las partes privadas del objeto.

Para todos los propósitos, un objeto es una variable de un tipo definido por el usuario. Puede parecer extraño que un objeto que enlaza código y datos se pueda contemplar como una variable, sin embargo, en POO es precisamente el caso. Cada vez que se define un nuevo objeto, se está creando un nuevo tipo de dato y cada instancia específica de esté tipo es una variable compuesta.

2.- Polimorfismo.- (Muchas formas) Es la cualidad que permite que un nombre se utilice para dos o más propósitos relacionados, pero técnicamente diferentes. El propósito del polimorfismo aplicado a la POO es poder usar un nombre para especificar una clase general de acciones, y dentro de ésta la acción específica a aplicar está determinada por el tipo de dato, por ejemplo, en C, que no se basa significativamente en el polimorfismo, la acción de valor absoluto requiere tres funciones distintas: abs( ), labs( ) y fabs( ). Estas funciones calculan y devuelven el valor absoluto de un entero, un entero largo y un valor real, respectivamente, sin embargo, en C++, que incorpora polimorfismo, a cada función se puede llamar abs( ). El tipo de datos utilizado para llamar a la función determina qué versión específica de la función se está usando. En C++ es posible usar un nombre de función para propósitos muy diferentes y eso se llama sobrecarga de funciones.

De forma general, el concepto de polimorfismo es la idea de una interfaz con múltiples métodos; esto significa que es posible diseñar una interfaz genérica para un grupo de actividades relacionadas, sin embargo, la acción específica ejecutada depende de los datos. La ventaja del polimorfismo es que ayuda a reducir la complejidad permitiendo que la misma interfaz se utilice para especificar una clase general de acción.

Es trabajo del compilador seleccionar la acción específica que se aplica a cada situación; el programador no necesita hacer esta selección manualmente, sólo necesita recordar y utilizar la interfaz general. Como ilustra el ejemplo del párrafo anterior, tener tres nombres para la función de valor absoluto en vez de uno, hace que la actividad general de obtener el valor absoluto de un número sea más compleja de lo que realmente es.

El punto clave a recordar sobre el polimorfismo es que permite manejar complejidades más grandes a través de la creación de interfaces estándar para actividades relacionadas.

3.- Herencia.- Es el proceso mediante el cual un objeto puede adquirir las propiedades de otro, concretamente, un objeto puede heredar un conjunto general de propiedades a las que puede añadir aquellas características que son específicamente suyas.

La herencia es importante porque permite que un objeto soporte el concepto de clasificación jerárquica. Mucha información se hace manejable gracias a esa clasificación, por ejemplo, pensemos en la descripción de una casa; ésta es parte de una clase general llamada *edificio*, a su vez, éste es una fracción de la clase general llamada *estructura*, que es parte de una clase aún más general de objetos que se puede llamar obra-hombre. En cualquier caso, la clase hija hereda todas las cualidades asociadas con la clase padre y le añade sus propias características definitorias.

Sin el uso de clasificaciones ordenadas, cada objeto tendría que definir todas las características que se relacionan con él explícitamente, no obstante, mediante el uso de la herencia es posible describir un objeto estableciendo la clase general (o clases) a las que pertenece, junto con aquellas características específicas que lo hacen único.

PRUEBAS

Una vez que se ha generado el código, se inician las pruebas del videojuego para comprobar que sus características funcionen de manera adecuada. La prueba del software es un elemento crítico para la calidad y representa una revisión final de las especificaciones,

del diseño y de la codificación.

La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los costos humanos y materiales asociados a un fallo del mismo están motivando la creación de pruebas minuciosas y bien planificadas.

El proceso de pruebas se centra en los contextos lógicos internos del software, asegurando que todas las estructuras que forman el videojuego se han comprobado y en los procesos externos funcionales, es decir, la realización de pruebas para la detección de errores para asegurar que la entrada definitiva produzca resultados reales de acuerdo con las espectativas requeridas.

Durante las fases anteriores de análisis y diseño, se intenta construir la aplicación partiendo de un concepto abstracto para llegar a una implementación tangible, posteriormente se realiza la prueba a fin de obtener una demolición del software construido.

Las pruebas son procesos de ejecución con la intención de descubrir un error; además, un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. Estos dos elementos se pueden considerar como íntegros objetivos en esta etapa.

En resumen podemos decir que el objetivo primordial de una prueba es detectar diferentes clases de errores con el menor tiempo y esfuerzo posibles. Si la prueba se lleva cabo con éxito descubrirá errores en el videojuego y como ventaja secundaria, demostraría hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y espectativas esperadas, además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan indicaciones de la fiabilidad del software y, de alguna manera, señalan la calidad del producto final. Otra cosa que se debe tener en consideración es que la prueba no puede asegurar la ausencia de defectos.

En circunstancias ideales en el momento de diseñar cualquier programa de computadora, no se puede olvidar el contexto de pruebas y que éstas sean prontas, expeditas y eficientes, por otro lado hay que indicar que existen métricas que pueden usarse para medir la consistencia y facilidad de éstas, pero es un tópico que no se plantea en este trabajo.

En las pruebas se debe considerar un conjunto de características propias de los videojuegos: Operatividad, observabilidad, controlabilidad, capacidad de descomposición, estabilidad y facilidad de comprensión.

La operatividad se refiere a que, en cuanto mejor funcione el videojuego más eficientemente se puede probar; mientras que la observabilidad es todo aquello que se puede ver y que se debe probar (salidas, cambio de estados, etc.)

Por su parte la controlabilidad hace referencia a que en cuanto sea mayor el control de los procesos que conforman al software es más simple automatizar y optimizar.

La capacidad de descomposición señala que controlando el ámbito de las pruebas se pueden aislar rápidamente los problemas y llevar a cabo los exámenes de regresión, por lo tanto, en cuanto menos haya que probar más sencillo y rápido se puede examinar y, finalmente, la facilidad de comprensión va relacionada a la documentación que se tenga del videojuego, es decir, entre más información se tenga disponible las pruebas se llevarán a cabo de forma más inteligente.

En síntesis, se deben diseñar pruebas que tengan la mayor probabilidad de encon-

trar un mayor número de errores con la mínima cantidad de esfuerzo y tiempo disponible. Cualquier producto de la ingeniería de software puede examinarse con una de estas formas: Conociendo la función especifica para la que fue diseñado. Se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa y al mismo tiempo buscando errores en cada función; y la otra, conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que todos los elementos que lo conforman encajan, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

La primera forma de examinación se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, es decir, las pruebas que pretenden demostrar que las funciones del videojuego son operativas, que la entrada se acepta de forma adecuada y que produce un resultado correcto, también, que la integridad de la información externa se mantiene.

La segunda forma se basa en un minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y/o ciclos.

Una vez definidos los conceptos de codificación y pruebas, a continuación, y como parte de la aplicación, se incluye una bitácora de actividades en donde se detallan todos y cada uno de los pasos que se llevaron a cabo para el desarrollo del producto final.

BITÁCORA

1.- Generación de la estructura del laberinto en Autocad.- Dentro de la primera actividad para la generación del laberinto se encuentra el desarrollo de manera estructural del espacio de éste. Para su elaboración se utilizó el software de Autocad como herramienta de diseño, aquí se incluyen todas las dimensiones adecuadas (volumen a escala) del escenario que conforma el videojuego.

2.- Exportación de la estructura del laberinto a 3D Max Studio.- Una vez lista la estructura del laberinto fue necesario exportar el archivo de Autocad a 3D Max Studio para poder dar los últimos detalles de la estructura del mismo, así como llevar a cabo la ambientación mediante el mapeo de texturas, luces y eventos o triggers. Podemos decir que 3D Max Studio es la interfaz entre Autocad y el fly3D o game engine; como dato, el 3D Max Studio se encarga de la generación de archivos correspondientes para que el game engine pueda obtener y manejar la geometría del laberinto.

a) Ambientación del escenario.- Una vez que se obtuvo la estructura del laberinto en Autocad y se logró exportar a 3D Max Studio, se procedió a la investigación del proceso de mapeo de texturas de cada uno de los elementos que conforman la estructura del laberinto (paredes y puertas).

Obteniendo el archivo en formato compatible se mapearon las texturas en el laberinto, con las propiedades necesarias para que se pudieran visualizar al momento de exportarlo al *fly3D.* Para dar una mejor ambientación al laberinto, el proceso de mapeo o texturización que se llevo a cabo fue por caras.

Dentro de estas pruebas se incorporó el manejo de la cámara y de las luces para su mejor visualización.

b) Investigación de parámetros especiales para exportación hacia fly3D.- En esta sección se investigó todo lo relacionado sobre la exportación del archivo de 3D Max Studio hacia *fly3D*. Se llevo a cabo la modificación mediante parámetros especiales para generar los archivos con extensiones FMF, F3D, FEM, FMP, que son necesarios para el game engine. Una vez teniendo el modelo final con texturas, triggers y puertas se exporta al formato que el game engine pueda leer para así poder generar los archivos del videojuego.

c) Investigación de visual C++ orientado a objetos.- Debido a la estructura del game engine, se procedió a la investigación de la programación de visual C++ orientado a objetos, logrando la modificación del código de cada uno de los objetos necesarios en el proyecto de tesis.

d) Modificaciones del archivo para insertar eventos (triggers).- En esta parte se modifica el código para obtener la funcionalidad requerida de los objetos *triggers* que disparan los eventos. Éstos son de suma importancia debido a que representan los elementos que provocan la interacción con el usuario del videojuego durante su navegación.

e) Investigación sobre el despliegue de texto en pantalla y captura de respuesta desde el teclado.- Dado que es necesario desplegar las preguntas en cada sala y en cierto momento, fue necesario modificar el código para leer desde un archivo de texto las preguntas y las opciones de respuesta, a su vez, se llevó a cabo las modificaciones para la captura de las respuestas correctas o erróneas por medio del teclado

f) Generación del archivo AVI para la introducción y el final del videojuego.- Dado que el juego lleva un video de introducción y final como parte de la historia, se generó el archivo AVI necesario para que el game engine los reprodujera en el momento deseado.

g) Generación de archivos CTP Y DLL para el proyecto de tesis.- Una vez modificado el código se generaron las DLL's que utiliza el *game engine* para la interacción de los objetos, así como el CTP, el cual se encarga de almacenar algunas características específicas de los objetos.

h) Construcción del archivo ejecutable del game engine.- Finalmente se generó el archivo ejecutable donde se desarrolla la aplicación de manera visual, obteniendo nuestro producto final o videojuego.

CAPÍTULO VI

MANTENIMIENTO

Uno de los objetivos primordiales de la ingeniería de software es mejorar la facilidad con la que se pueden implantar cambios y reducir la cantidad de esfuerzo para implementarlos.

Se podría pensar que el proceso de desarrollo de aplicaciones o software se termina cuando se tiene el producto final tangible, sin embargo, si queremos que el producto cumpla con ciertos elementos que garanticen su calidad, es claro que debemos fabricarlo apegados a un paradigma o ciclo de vida que contemple la ingeniería de software.

De acuerdo al ciclo de vida que se utiliza en el presente trabajo y a lo largo de los capítulos anteriores, se ha venido comentando que este paradigma tiene la característica especial de poder retomar alguna de las etapas anteriores con el objetivo de depurar o corregir. Esta característica o forma de trabajo en la etapa de mantenimiento no es la excepción. Tenemos como ejemplo la curva real de fallos (ver gráfica 5) durante su vida el software sufre cambios y modificaciones que sirven como medida de mantenimiento a fin de ir mejorando su calidad, o bien, solucionando problemas o inconsistencias.

Podemos decir entonces que la etapa de mantenimiento se centra en los cambios que van asociados a la corrección de errores o fallas, a las adaptaciones requeridas a medida que evoluciona el entorno de software y a los cambios debidos a las mejoras producidas por los avances tecnológicos que surgen día a día. De acuerdo a esto, definimos tres tipos de cambios que se pueden presentar en algún momento del ciclo de vida del software:

1.- Corrección.- Incluso llevando a cabo las mejores garantías de calidad, es muy probable que después del proceso minucioso de pruebas se descubran defectos. La acción correctiva modifica el software para solucionar los defectos.

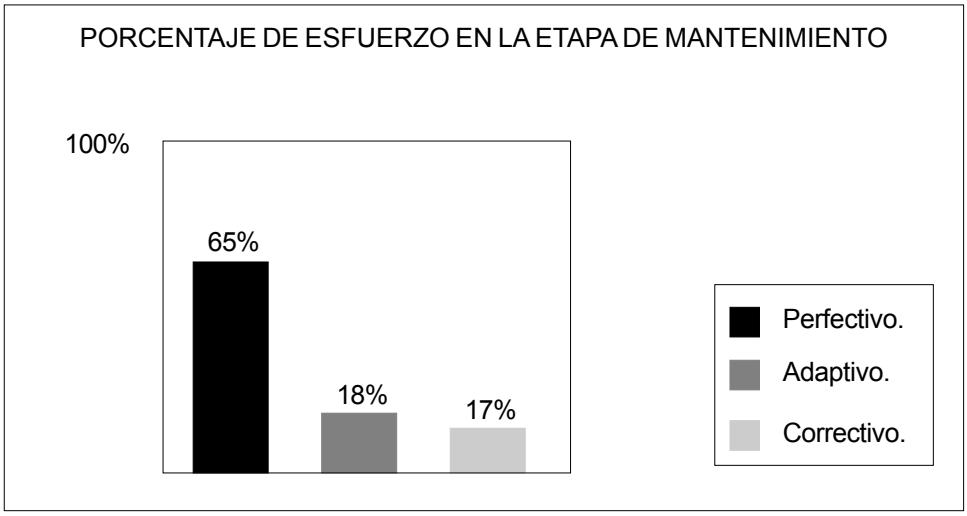
2.- Adaptación.- Con el paso del tiempo es probable que cambie su entorno original. La acción adaptativa produce modificaciones al software para acomodarlo a los cambios de su entorno externo.

3.- Mejora.- Conforme se utiliza la aplicación, el usuario puede requerir funciones adicionales que van a producir beneficios. La acción perfectiva lleva más allá al software de los requisitos funcionales originales. Esto influye en la comercialización del videojuego.

En resumen, la fase de mantenimiento vuelve a aplicar las etapas de análisis, diseño y codificación, pero en el contexto del software ya existente.

Contrario a lo que se piensa, el mantenimiento cuenta con más esfuerzo que cualquier otra actividad de la ingeniería de software; si lo apreciamos desde la perspectiva que los requerimientos de los videojuegos cambian debido a que el ambiente se transforma, o bien, que se encuentra bien acoplado a su ambiente pero éste repentinamente cambia, necesariamente el producto debe sufrir modificaciones. Por eso, todo software debe ser mantenido si se requiere que sea útil y cumpla con los mínimos requerimientos de calidad. De acuerdo a la experiencia, o bien a la intuición empírica en la elaboración de videojuegos,

podemos establecer un gráfico que representa la distribución de esfuerzo que se lleva a cabo en la etapa de mantenimiento. (ver gráfica 33)



Gráfica 33

Como se ve en la gráfica, representa un mayor esfuerzo hacer una mejora o ampliación, que una corrección o adaptación de elementos. Teóricamente esa gráfica puede ser cierta, sin embargo está directamente asociada a la documentación de análisis y diseño más completa y clara que se tenga.

La facilidad de mantenimiento es la factibilidad con la que se puede corregir un software si se encuentra un error, es la adaptación que se hace si su ambiente cambia o mejora si se desea un cambio de requisitos. No hay forma de medir directamente la facilidad de mantenimiento, por consiguiente, se deben utilizar medidas indirectas; por ejemplo, una simple métrica asociada a la facilidad es el tiempo medio de cambio, es decir, el tiempo que se tarda en analizar la viabilidad del cambio, en diseñar la modificación adecuada, en implementar o codificar el cambio, probarlo y finalmente distribuir (comercializar) el cambio al usuario. Como media, los programas que son más fáciles de mantener tendrán un tiempo medio de mantenimiento más bajo que los programas más difíciles de mantener.

En el capítulo anterior se hizo una definición de componentes o recursos de software, y que están directamente asociados al mantenimientos de éstos. Estos componentes están expuestos a sufrir alguno de los tres tipos de cambios (corrección, mejora o adaptación) que abarca el mantenimiento, es así que los componentes ya desarrollados a pesar de estar listos para utilizarse, debido a la modificación de su entorno se deben adaptar o mejorar; los componentes ya experimentados a pesar de ya estar probados se pueden modificar por cambio de requerimientos, sin embargo, el esfuerzo de mantenimiento es relativamente bajo, por el contrario los componentes con experiencia parcial requerirán de un esfuerzo considerable de mantenimiento en caso de sufrir modificaciones provocado también por cambios en sus requerimientos y, finalmente, los componentes nuevos sufrirán cualquiera de los cambios una vez probados o implementados en el software.

Cuando se construye un software de computadora, los cambios son inevitables, y estos aumentan el grado de confusión entre los desarrolladores o ingenieros de software que están trabajando en el proyecto. La confusión surge cuando no se han analizado los cambios antes de realizarlos, no se han registrado antes de implementarlos o no se han controlado de manera que mejoren la calidad y reduzcan los errores.

El contexto de coordinar el desarrollo de software, para nuestro caso, un videojuego, para minimizar la *confusión* se conoce como *gestión de configuración del software*. Este término se refiere a: Identificar, organizar y controlar los cambios o modificaciones que sufre el producto que construye un equipo de desarrollo, su meta primordial es maximizar la productividad y calidad minimizando los errores.

La gestión de configuración del software es una actividad que se lleva a cabo a lo largo del proceso de construcción del producto. Como el cambio se puede producir en cualquier momento, estas actividades sirven para identificarlo, controlarlo y garantizar que se implemente adecuadamente.

Es importante distinguir claramente entre el mantenimiento del software y la gestión de configuración del software. El mantenimiento es un conjunto de actividades de ingeniería de software que se producen después de que el software haya sido comercializado y esté funcionando, mientras que la gestión de configuración del software es un conjunto de actividades de seguimiento y control que comienzan cuando se inicia el proyecto de desarrollo y termina cuando el software queda fuera de circulación.

Podemos decir entonces que el mantenimiento de software forma parte de las actividades de la gestión de configuración del software.

Usualmente, es más caro añadir funcionalidad después de que el producto ha sido desarrollado que cuando se está diseñando. Esto se presenta por diversos factores como: Que el software puede estar deficientemente estructurado y por consiguiente es difícil de entender; que los cambios o procesos de mantenimiento pueden introducir nuevas fallas si es que estas modificaciones son complejas; la estructura del videojuego puede degradarse debido a cambios continuos, o bien, que no exista la documentación adecuada que describa al producto, no obstante de esta problemática, uno de los factores más importantes es que el equipo de desarrollo le da poco valor a la etapa de mantenimiento debido a que no es una actividad creativa. Todo esto trae por consecuencia una administración de configuraciones inadecuada lo que a menudo implica que no se tenga un buen control de las versiones del software.

Existe en la administración del mantenimiento un proceso llamado *control de versiones,* el cual combina procedimientos y herramientas para gestionar las versiones de los objetos creados durante el proceso de ingeniería de software. Este control puede ser tan sencillo como un número específico de versión asociado a cada objeto o tan complejo como una cadena de variables lógicas que especifiquen tipos de cambios funcionales aplicados al sistema.

Asociado al anterior existe otro llamado control de cambios. En un proyecto de desarrollo de software, el cambio incontrolado lleva rápidamente al caos, por ello este control combina procedimientos humanos y herramientas automáticas para proporcionar un mecanismo para la organización del mantenimiento.

El proceso de mantenimiento se lleva a cabo debido a cambios pedidos por los usua-

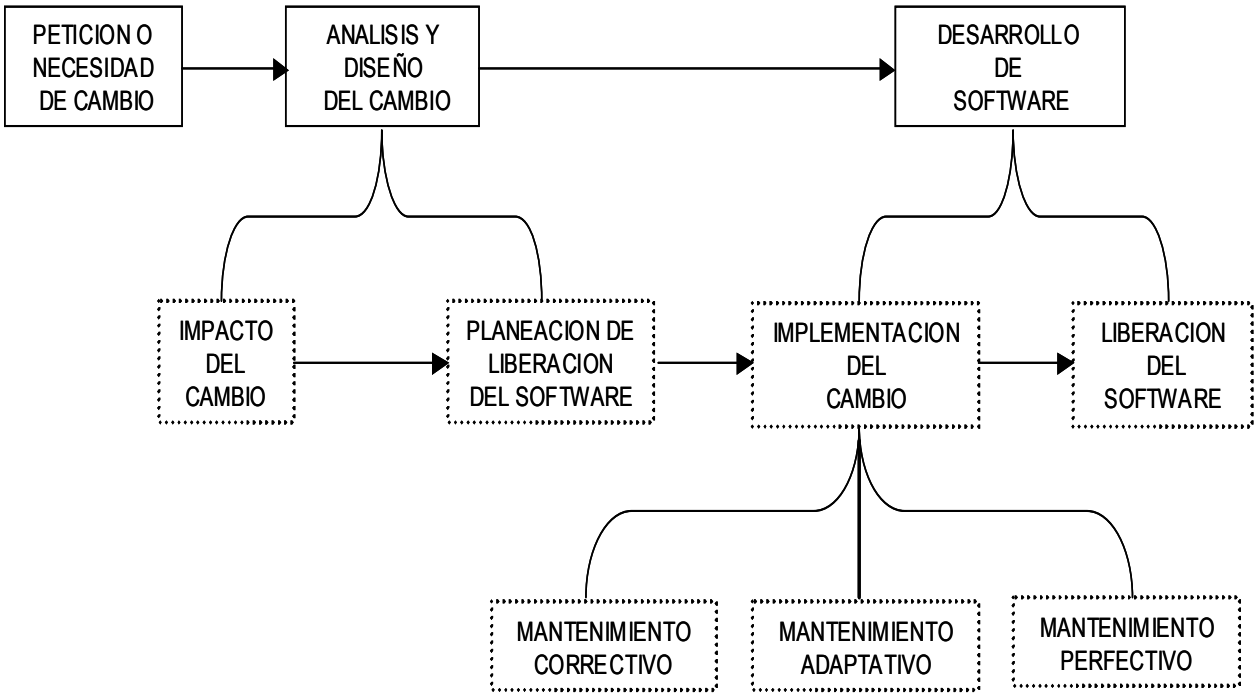
rios o por los requerimientos del mercado; la segunda opción es la que rige el control de cambios y versiones. Como se mencionó en el capítulo II, lo viable que sea el producto desde el punto de vista comercial es la pauta que indicará si el juego requiere de alguna actualización o bien la continuación de la historia y por consiguiente una versión nueva.

Los cambios normalmente se atienden en orden de pedido o necesidad y se implementan en una nueva versión del sistema

Como se mencionó en la etapa de mantenimiento, se requiere nuevamente una etapa de análisis y diseño para conocer la viabilidad del cambio, sin embargo hay casos donde los programas algunas veces necesitan ser reparados sin que se realice una iteración completa del proceso, lo cual provoca problemas ya que la documentación y los programas se desactualizan.

En la siguiente grafica (34) se observan todas las etapas que de alguna forma intervienen el proceso de mantenimiento y que garantizan que los cambios realizados conserven la calidad del software. De acuerdo a ésto podemos decir que una vez hecha la petición o la necesidad de un cambio en el software, el proceso de mantenimiento se compone de dos etapas fundamentales: La primera, encargada de hacer el trabajo analítico y de diseño teniendo en cuenta el impacto del cambio y la visualización del proceso de liberación del mismo (nueva versión); y la segunda, encargada de hacer tangible todo el proceso de la primera etapa, es aquí donde se implementa el cambio (mantenimiento correctivo, de adaptación y perfectivo) para posteriormente liberarlo.

A lo largo de este trabajo se ha comentado que parte del correcto desarrollo del videojuego debe fundamentarse en la documentación, además de la descripción de la

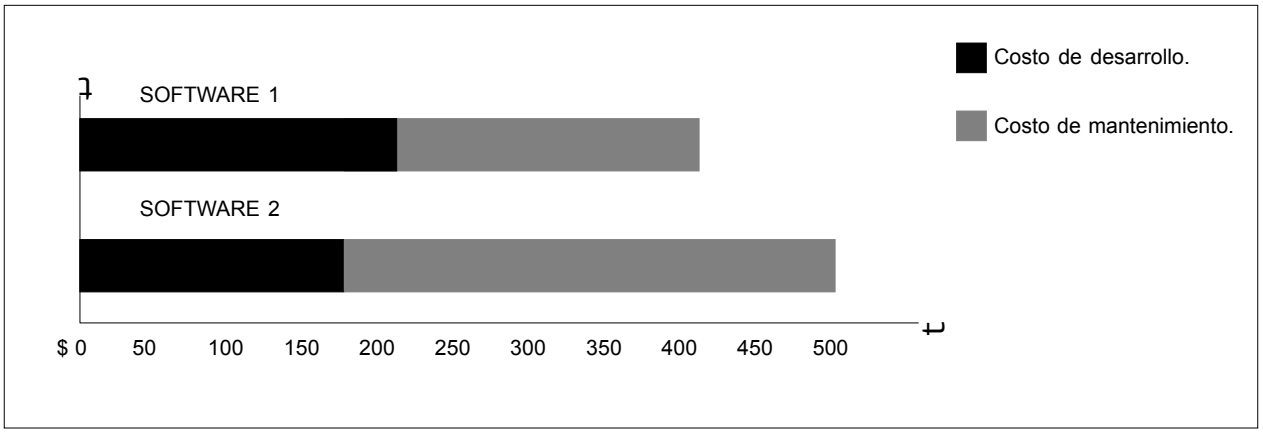


Gráfica 34

arquitectura del sistema (diagrama de arquitectura) y documentación del diseño del programa (diagrama entidad-relación, de transición de estados, diccionario de datos, etc.) se debe agregar un documento más y que forma parte de la etapa de mantenimiento: La guía del mantenimiento del sistema. En este punto es importante aclarar que como parte de este trabajo no se tiene contemplado el mantenimiento aplicado como tal al videojuego, dado que el contenido de este capítulo es solamente teórico.

Es lógico que cada en una de las etapas que conforman el ciclo de vida de un software implican un costo asociado. Específicamente para el proyecto de este trabajo algo importante en el desarrollo de un videojuego es la parte comercial.

Si recordamos el punto de importancia del capítulo II, en donde se hace un estudio económico y de acuerdo a cifras y proyecciones se concluyó qué tan viable puede ser el producir un videojuego. Si bien la inversión durante la fabricación del producto (incluyendo la comercialización) es importante, los costos asociados a la etapa de mantenimiento no deben dejar de considerarse desde el inicio del proyecto. De acuerdo a la experiencia, los costos asociados a la etapa de mantenimiento son entre dos y cien veces mayor a los del desarrollo dependiendo de la aplicación que se trate, estas cifras se deben a factores técnicos y otros que infieren directamente en el costo del producto y a la calidad del proceso en el desarrollo. (Ver gráfica 35)



Gráfica 35

Conforme el software va evolucionando en la etapa del mantenimiento el costo se va incrementando ya que entre más cambios sufre más se corrompe la idea original del sistema, por lo que en las etapas subsecuentes, en caso de requerir mantenimiento, es más difícil de llevar a cabo.

Existen diversos factores que afectan directamente el costo del mantenimiento:

1.- La interdependencia entre módulos.- Recordemos que si el software cumple con un diseño arquitectónico adecuado y los módulos que lo conforman cumplen con una dependencia necesaria, el mantenimiento es más barato y simple (es más fácil hacer cambios a un sólo módulo sin afectar a los demás que producir un cambio que implica modificaciones a más de uno).

2.- El lenguaje de programación.- Es más sencillo mantener un software programado en lenguajes de alto nivel que en lenguajes viejos.

3.- El estilo de programación.- Un programa estructurado y desarrollado de la mejor forma es más fácil de mantener.

4.- La validación y pruebas.- Si el sistema antes de su liberación o entrega se le aplican las pruebas necesarias y completas que nos puedan garantizar que su nivel de calidad es alto, lógicamente su proceso de mantenimiento es más simple.

5.- La documentación.- Una buena recopilación de escritos que nos integran la documentación del software (incluyendo diagramas) ayuda al entendimiento de éste, lo que influye directamente en la etapa del mantenimiento.

6.- La administración de la configuración.- Una buena administración implica que existan ligas entre el software y su documentación y que éstas son mantenibles.

7.- El dominio de la aplicación.- El mantenimiento del producto es más fácil en dominios maduros y bien entendidos.

8.- Estabilidad del personal.- En los recursos humanos es importante que el equipo esté altamente capacitado y tenga el conocimiento mínimo necesario del programa.

9.- La edad de los programas.- A medida que el programa es más viejo es más difícil de mantener. (Usualmente)

10.- El ambiente externo.- Si un programa es dependiente de su ambiente externo tendrá que soportar cambios que se reflejen en su entorno o medio ambiente, y

11.- La estabilidad del hardware.- El software diseñado sobre un hardware estable no requerirá cambios cuando éste cambia.

Asociado directamente al costo de esta etapa se encuentra la métrica del mantenimiento que son aquellas mediciones de las características del software que pueden, de alguna forma, predecir su mantenibilidad, sin embargo, hay que aclarar que no es un proceso simple, existen componentes cuyas mediciones están fuera del límite y por consiguiente son muy costosos de mantener. Dentro de las métricas que se pueden considerar podemos mencionar:

1.- El control de la complejidad que se puede medir en función de las condicionales del programa.

2.- La complejidad de datos que se refiere a las estructuras de datos e interfaces de los componentes.

3.- El acoplamiento, es decir, qué tanto se involucra el intercambio entre componentes o estructuras de datos.

4.- El grado de interacción con el estrato usuario, mientras más entradas y salidas del usuario existan el software requerirá más cambios.

5.- Los requerimientos de velocidad y espacio referidos a habilidades especiales difíciles de mantener.

Otros factores no técnicos que entran en juego son el tiempo promedio que se toma para el análisis de impacto y el tiempo promedio que se toma en implementar el cambio.

De acuerdo a la información recabada en este capítulo, podemos concluir que el software está en constante cambio, que trae como consecuencia que un programa desarrollado en el mundo real necesariamente deba sufrir cambios y progresivamente pierda su funcionalidad en el proceso de fabricación; que el software conforme evoluciona se incrementa su complejidad, es decir, si se llevan a cabo desmesuradamente cambios al producto o software el entendimiento en su funcionalidad se vuelve más difícil, y finalmente, que el proceso de puesta a punto del software es largo, cada cambio que sufre necesariamente extenderá el tiempo de desarrollo del mismo.

CONCLUSIONES

En un mundo globalizado, en donde la sociedad evoluciona y se mantiene en constante crecimiento, el estar comunicado es una de las necesidades más elementales.

El uso de los medios electrónicos es la herramienta con la que cuenta el hombre para satisfacer ese sentido. Los sistemas informáticos como parte importante del entorno mundial deben contar con la calidad necesaria para cumplir su cometido, sin embargo, no debemos perder la objetividad de que a pesar del crecimiento en forma exponencial del mundo de la informática, el hombre, hablando en términos de especie, todavía no adopta del todo la computación como parte de su forma de vida a fin de satisfacer sus necesidades.

En la mayoría de los casos, la principal causa por la que el ser humano no tenga el acceso a un medio electrónico es la falta de recursos económicos suficientes para su adquisición, incluso el estrato que cuenta con ello pasa por un proceso de adaptación, por no decir de temor, para descubrir todas las bondades que representa el uso de estos elementos tecnológicos, a pesar de que existen herramientas que auxilian en la transición, al respecto podemos decir que los videojuegos (como entretenimiento y diversión) son parte de esas herramientas y, por consecuencia, acercan al ser humano a la tecnología.

La inquietud, de parte de los integrantes que presentamos esta tesis, fue el de llevar acabo un estudio en materia de videojuegos con el objetivo de contar con los elementos necesarios para poder entender este nicho de oportunidades.

A lo largo de más de dos años de investigaciones que sustentan este trabajo se obtuvieron todos los elementos necesarios para concluir lo siguiente: *En México el desarrollo o producción de videojuegos es muy escasa o nula, a diferencia de países como Estados Unidos, Canadá, Japón, Francia, Alemania, Inglaterra y España que son los principales productores de este tipo de software. En México sólo existen tres empresas dentro de la industria.*

Cabe recordar que la industria del videojuego a nivel mundial es redituable, ya que se estima que se pueden obtener ganancias de 2 mil millones de dólares anuales en los próximos 15 años, además que se generarían cientos de empleos en diversas ramas, dado que se requieren programadores, diseñadores gráficos, pedagogos, psicólogos, músicos, modeladores y escritores entre otras ramas profesionales y de mano de obra. Por otro lado y si se quisiera establecer una verdadera industria en materia de desarrollo de software, sobre todo como atracción de capital extranjero, es necesario crear un marco jurídico, fiscal y de seguridad.

El tema de esta tesis surge de la inquietud por conocer más acerca del contexto y posibilidades en la graficación por computadora, aunado a la aplicación de los conocimientos obtenidos a lo largo de la carrera.

Al principio del proyecto las expectativas eran muy grandes, ya que pensábamos que el desarrollo de la tesis tendría cierto grado de complejidad, pero nunca nos imaginamos que tanto.

Dada la gran cantidad de videojuegos que circulan por el mercado hubiera parecido fácil crearlos, pero en realidad no se contaba con los conocimientos de diseño y modelado. Este problema se sumó a las complicaciones consideradas *normales* que se presentan en un desarrollo serio de cualquier elemento software.

La primera propuesta que tuvimos para la historia y temática del videojuego era de *Piratas* con un importante número de opciones y escenas, sin embargo, las diferentes dificultades fueron limitando las expectativas; cambiando de propuesta a un *Laberinto lúdico* y que en cada sala se presentara un juego de tablero o una actividad para pasar a la siguiente, para finalmente terminar con la propuesta actual que es en un laberinto con trivias.

En un principio, la dificultad para obtener información acerca de los videojuegos nos demoró, dado que no había documentación actualizada y en idioma castellano sobre cómo desarrollarlos.

Otro problema fue la necesidad de investigar qué herramientas existían para la elaboración de ellos, y una vez en conocimiento de ellas, se seleccionó la que mejor se adaptaba a nuestras necesidades para posteriormente investigar acerca de su funcionamiento. La herramienta principal para el desarrollo es el llamado *game engine* o máquina del videojuego, nuestra elección fue el *fly3D* (de carácter público, lo cual fue una ventaja ya que no hubo necesidad de investigar acerca de permisos por concepto de utilización), sin embargo se requería del *3D Max,* una herramienta de modelado para la geometría del videojuego.

Dada la falta de experiencia para la fabricación del modelo o escenario, nos dimos a la tarea de utilizar la herramienta de *Autocad* para la generación, para después ambientar (mapeo de texturas y luces) el escenario en *3D Max.*

La inexperiencia en el uso de las herramientas *Cad*, así en la programación orientada a objetos nos dificultó la modificación de las DLL´s del *game engine*, lo que aumentó el grado de complejidad del desarrollo de nuestra aplicación.

Las dificultades que se presentaron en el desarrollo incrementaron el tiempo estimado del proyecto, sin embargo, el objetivo de aprender más sobre graficación por computadora y la ingeniería de software, se cumplió. Aunque el videojuego haya disminuido en complejidad, los fundamentos para el desarrollo de uno se obtuvieron.

Por su parte los videojuegos como elemento perteneciente al software de computadora, si se desarrollan apegados a los conceptos que contempla la ingeniería de software se pueden construir de manera sencilla, logrando cumplir con estándares de calidad que satisfagan al usuario.

A través de este trabajo, se ha podido comprobar, una vez más, la importancia de la ingeniería de software y su relación con otras áreas.

Finalmente es importante mencionar que los videojuegos son un elemento que comienza a consolidarse en la gente, por lo que el contenido de éstos no sólo sirve como medio de entretenimiento, sino también como vehículo de aprendizaje y orientación para encausar sanamente el comportamiento de una sociedad.

BIBLIOGRAFÍA

Pressmann, R.S. "*Ingeniería del Software: un enfoque práctico*". Mc Graw Hill. Madrid, 1993. 3ª Edición.

Watt, Alan – Policarpo, Fabio. "*3D Game Real Time Rendering and Software Technology Volumen One*". Addison-Wesley, España, 2001.

Deloura, Marc. "*Game Programing Gems*". Editorial Charles River Media, Estados Unidos, 2000.

Curran, S. – Curnow, R. "*Juegos, Imágenes y Sonidos*". Editorial Gustavo Gili S.A., España, 1984.

Crawford, Cris. "*El arte del diseño de Juegos con Microcomputadora*". Mac Graw Hill. México, 1986.

Winston, Wayne L. "*Investigación de Operaciones: Aplicaciones y algoritmos*". Grupo Editorial Iberoamérica, México, 1994.

Galvis, A. *«Micromundos lúdicos interactivos: aspectos críticos en su diseño y desarrollo»*. 1988.

Gándara, M. *"Multimedios y Nuevas Tecnologías. Diplomado de Educación para los Medios".* UPN/ILCE. México, 1998

McDougall, A. – Sussex, R. – Cumming, G. – Cropp, S. "*Learner modelling by expert teachers: learner information space and the minimal learner model*". Aston University, England

Aaron M. Tenenbaum et. Al. "*Estructuras de Datos en C".* Prentice Hall, México, 1993.

Stroustrup, Bjarne. "*El lenguaje de programación C ++ Edición Especial*". Addison-Wesley, España, 2000.

Schildt, Herbert, "*C++ Guía de autoenseñanza*". Addison-Wesley, Inglaterra, 2000.

SITIOS EN INTERNET

http://www.fly3d.com.br/
http://www.sct.gob.mx
http://www.e-mexico.com.mx
http://www.famiweb.com/peques/videojuegos.htm
http://www.edunexo.com/Mexico/_primarios/videojuegos.htm
http://www.ucm.es/info/multidoc/multidoc/revista/num8/jpablos.html

ARTÍCULOS EN PERIÓDICOS Y REVISTAS

Universal, sección de Finanzas, Martes 5 de Junio de 2001

Reforma, sección Interface, Lunes 30 de Abril de 2001.