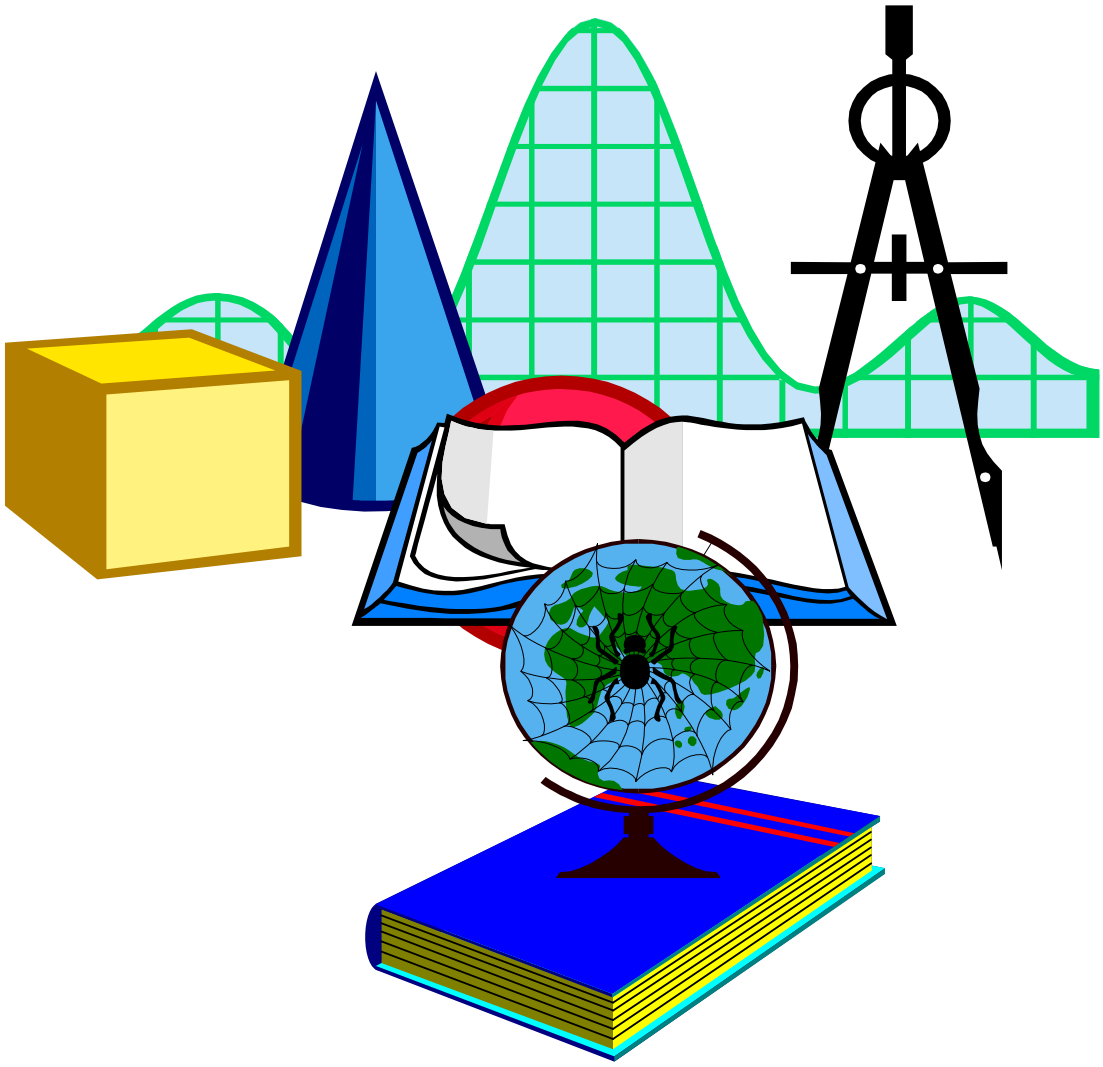
CAPITULO IV

METODOLOGIA PARA EL DESARROLLO DE ESCENARIOS VIRTUALES CON VRML



4.1. INTRODUCCION

Con el crecimiento vertiginoso de la tecnología y existencia de potentes servidores que forman parte de la gran red de la información o Internet, la presentación de la información en este medio masivo de entretenimiento y consulta ha ido cambiando de manera impresionante, de tal forma que la presentación de la información antes se la hacia solamente con texto, hasta llegar a presentarlo con imágenes, sonidos, texto e hiperenlaces; es decir, se ha llevado la multimedia al web, de esta misma forma se está y se ira vinculando la realidad virtual para lo cual existen herramientas para crear escenarios virtuales que cada día son más reales y no necesitan de características potentes del servidor, ni en los clientes para visualizarlos.

Todos están conscientes que Internet es la herramienta de hoy y el futuro para los negocios, es por eso, que todas las empresas desean tener un sitio en la red, el cual debe ser sencillo, práctico, real y lo más importante fácil de navegar y entender por el usuario. Esto no es difícil, con la Realidad Virtual en Internet, se puede presentar al cliente de una empresa, el producto a promocionarse, su forma, color y más a través de escenarios virtuales.

La metodología nace como una necesidad de crear un procedimiento práctico para cambiar rápidamente el ciclo de desarrollo de sitios web con realidad virtual no inmersiva, mejorando su aceptación y calidad, minimizando los costos de implementación utilizando VRML 2.0.

Por lo antes mencionados se ha decido crear una Metodología para el Desarrollo de Escenarios Virtuales con VRML, que guié y facilite la creación de escenarios virtuales con VRML.

4.2. CONCEPTOS UTILIZADOS EN LA METODOLOGIA

4.2.1. Modelamiento UML(Unified Modeling Languaje)

[1]Es un lenguaje de propósito general para el modelado orientado a objetos que mantiene independencia de los lenguajes de programación. UML combina notaciones provenientes desde:

- Modelado Orientado a Objetos.
- Modelado de Datos.
- Modelado de Componentes.
- Modelado de Flujos de Trabajo (Workflows).

[1] REF: http://www.dsic.upv.es/~uml

[2]La notación UML se ha convertido desde finales de los 90 en un estándar para modelar con tecnología orientada a objetos todos aquellos elementos que configuran la arquitectura de un sistema de información y, por extensión, de los procesos de negocio de una organización.

La tecnología orientada a objetos persigue el antiguo principio del divide y vencerás. Su objetivo es descomponer la complejidad en partes más manejables y comprensibles. No parece que esto sea algo novedoso con respecto a la tradicional descomposición funcional de los métodos estructurados. Sin embargo, la gran diferencia reside en aplicar la dualidad estructura - función en pequeñas unidades capaces de comunicarse y reaccionar basándose en la aparición de una serie de eventos. El esquema dominante de la separación de estructuras de datos y funciones (bases de datos y programas) está amenazado pero se resiste a desaparecer.

La dinámica de cambio no se desarrolla en la ingeniería del software con la misma velocidad vertiginosa con que se desarrolla la tecnología del hardware. La clave reside en que a diferencia de la electrónica, en los dominios del desarrollo de software no existe un vocabulario unificado. Desde la fase de concepción de un sistema a la instalación de sus componentes hay que mapear entre sí una gran diversidad de lenguajes orientados al análisis, diseño, código ejecutable, esquemas de bases de datos, componentes de páginas web, entre otros. Esta distancia entre la concepción y la usabilidad de un producto o de un proceso de negocio, exige cada vez más la capacidad de cooperación y comunicación de un equipo interdisciplinario muy especializado.

- **Plan Director de Proyecto**

Debe existir un plan general director, que permita cuantificar los esfuerzos en cada etapa de desarrollo de un Sistemas de Información, en donde se correlacionen las acciones en función de los objetivos del desarrollo de una aplicación, es equivalente a un cronograma de trabajo.

Todo el trabajo del análisis orientado a objetos se puede resumir en la Figura 4.1. donde las acciones a ejecutar o etapas son: Concepción, Formalización, Construcción, Transición, y los resultados que se buscan son: Misión, Modelo, Prototipo, Componentes y Certificación.

- **Esfuerzo en el Desarrollo**

Se miden por un lado las acciones a ejecutar o etapas: Concepción, Formalización, Construcción, Transición, en función de lo que se quiere finalizar: Misión, Modelo, Prototipo, Componentes y Certificación. El esfuerzo global realizado para concluir dicho análisis se muestra en la Figura 4.2.

[2] REF: http://www.vico.org

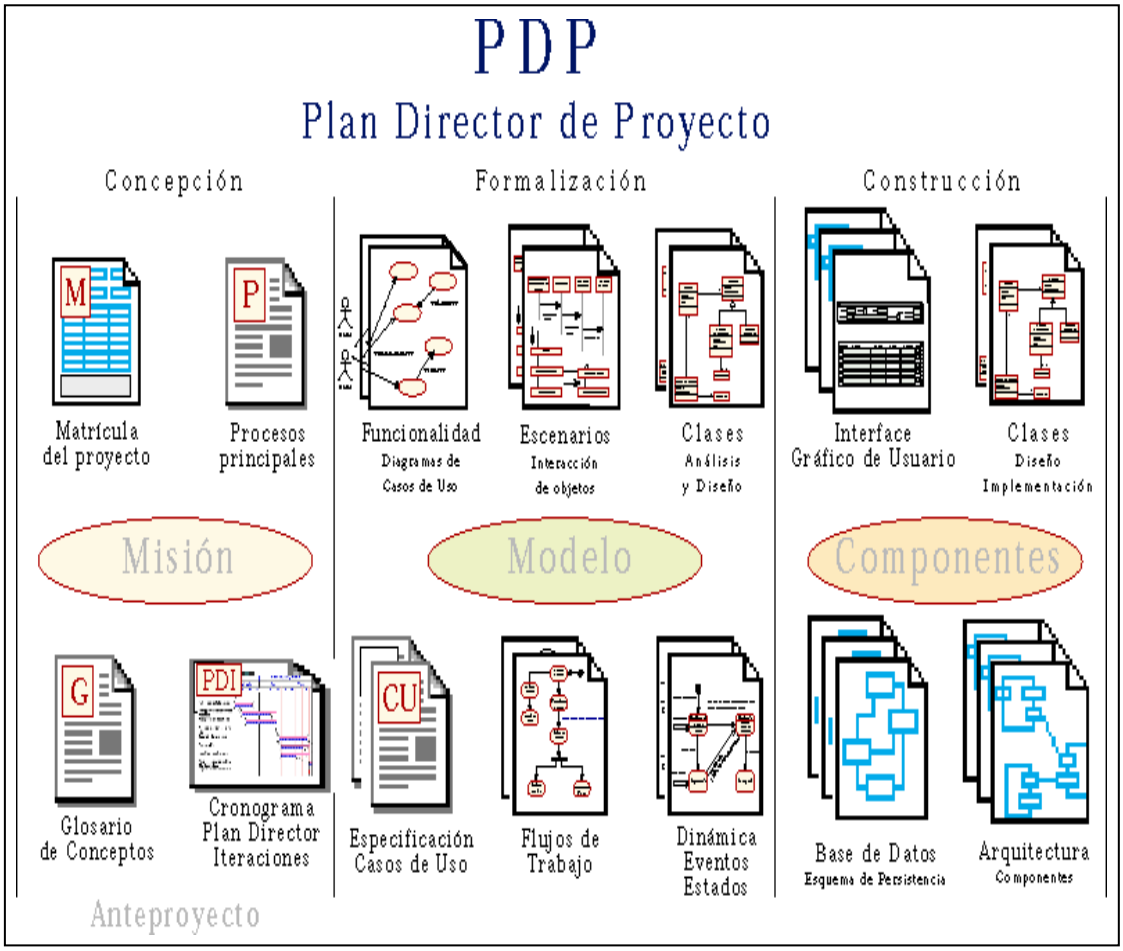


Figura 4.1. Plan Director de Proyecto

Concepción y Formalización: permiten plasmar una realidad, en función de un esquema de trabajo a ejecutar. El esfuerzo es mayor en esta etapa

Misión: la Concepción y Formulación utilizando Patrones de Análisis y Funcionales se unen para generar un Anteproyecto, que plantea una Misión a cumplir.

Modelo: el modelamiento requiere trabajar más detenidamente en la Formalización: Funcionalidad, Escenarios y Clases, y con detalle la especificación de Casos de Uso, Flujos de trabajo y la Dinámica de Eventos y Estados. El esfuerzo es mayor en la etapa de la Formalización.

Componente: los componentes requieren de las especificaciones de diseño dadas en el modelo, fundamental para proceder a la implementación de las clases y su comportamiento. Esta es la etapa de Construcción o desarrollo por lo que se requiere de un minucioso trabajo y mayor esfuerzo, ya que de aquí saldrán las especificaciones del comportamiento de los objetos,

para la elaboración de los componentes también se requiere del esfuerzo en la etapa de Transición.

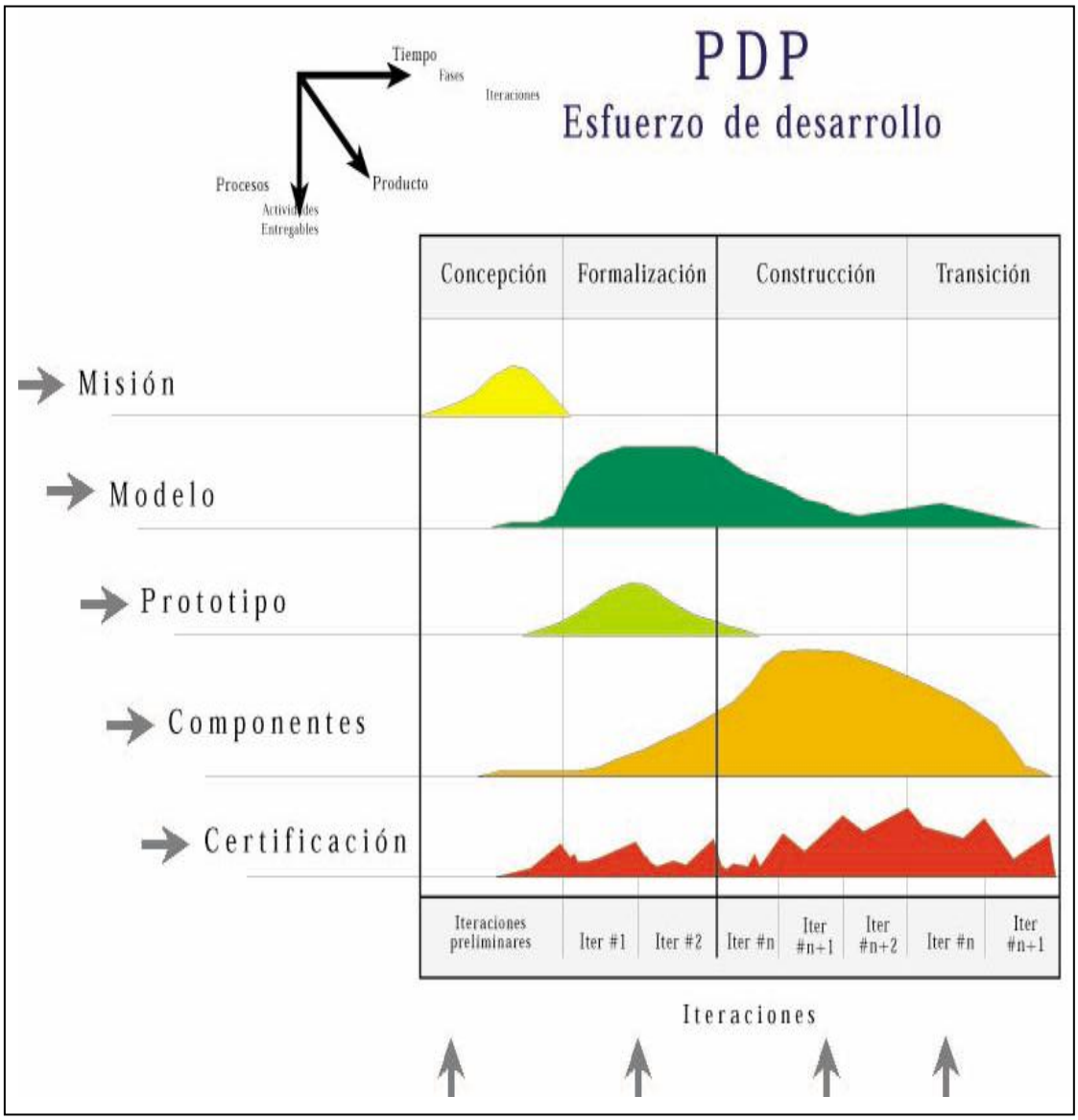


Figura 4.2. Esfuerzo Global del Modelamiento

— [3]Diagramas de UML

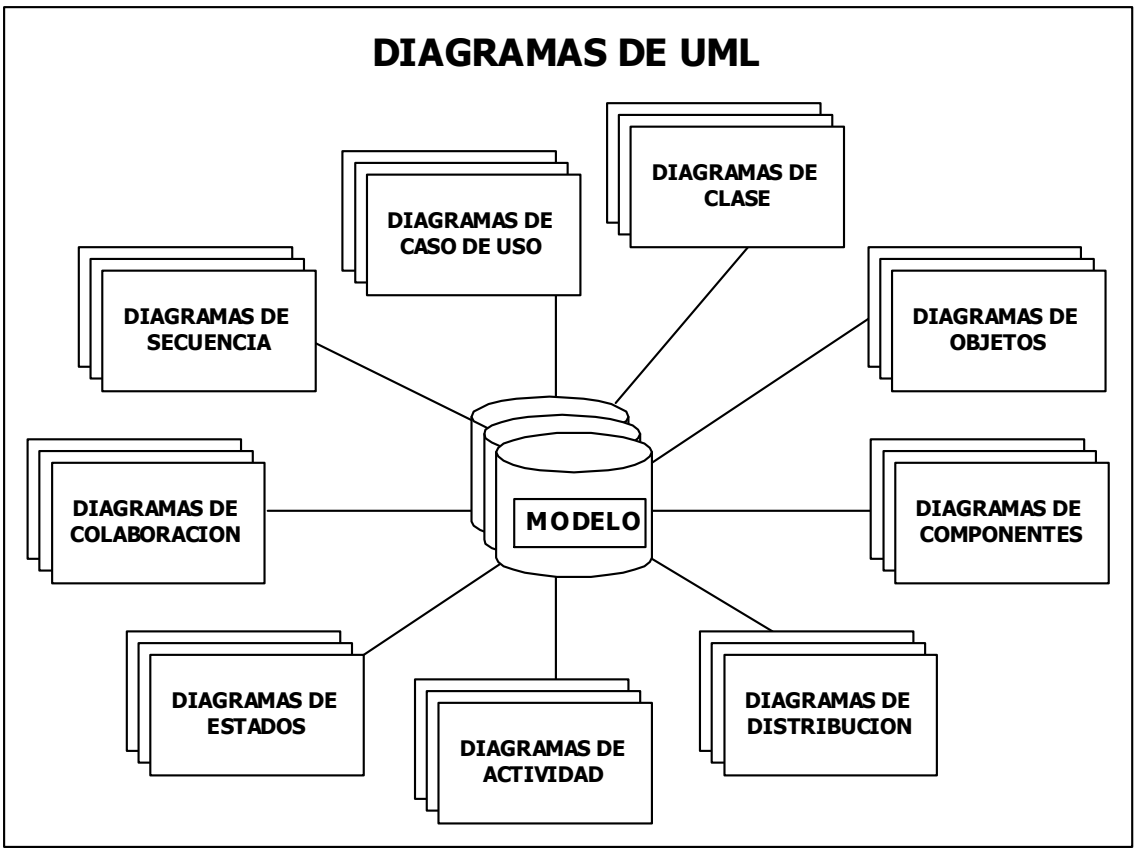


Figura 4.3. Diagramas de UML

Paquetes en UML: los paquetes ofrecen un mecanismo general para la organización de los modelos agrupando elementos de modelado. Se representan gráficamente como:

Cada paquete corresponde a un subconjunto del modelado y contiene, según el modelo, clases, objetos, relaciones, componentes y diagramas asociados. Un paquete puede contener otros paquetes, sin límite de anidamiento pero cada elemento pertenece a (está definido en) sólo un paquete.

[3] REF: http://www.dsic.upv.es/~uml

Una clase de un paquete puede aparecer en otro paquete por la importación a través de una relación de dependencia entre paquetes. Todas las clases no son necesariamente visibles desde el exterior del paquete, es decir, un paquete encapsula a la vez que agrupa.

+ Diagramas de Casos de Uso

Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja actualmente, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objetos, es una técnica para capturar requisitos.

Los Casos de Uso cubren la carencia existente en métodos previos (OMT, Booch) en cuanto a la determinación de requisitos.

Particionan el conjunto de necesidades atendiendo a la categoría de usuarios que participan en el mismo, esta basado en lenguaje natural, es decir, es accesible por los usuarios.

Actores:

- Principales: personas que usan el sistema.
- Secundarios: personas que mantienen o administran el sistema.
- Material Externo: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros Sistemas: sistemas con los que el sistema interactua.

La misma persona física puede interpretar varios papeles como actores distintos. El nombre de actor describe el papel desempeñado.

Los Casos de Uso se determinan observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario.

Construcción: un Caso de Uso debe ser simple, inteligible, claro y conciso. Las Preguntas claves para su construcción son:

- ¿Cuáles son las tareas del actor?
- ¿Qué información crea, guarda, modifica, destruye o lee el actor?
- ¿Debe el actor notificar al sistema los cambios externos?
- ¿Debe el sistema informar al actor de los cambios internos?

La descripción del Caso de Uso comprende:

- El inicio: ¿Cuándo y qué actor lo produce?
- El fin: ¿Cuándo se produce y qué valor devuelve?
- La interacción actor - caso de uso: ¿Qué mensajes intercambian ambos?
- Objetivo del Caso de Uso: ¿Qué lleva a cabo o intenta?
- Cronología y origen de las interacciones
- Repeticiones de comportamiento: ¿Qué operaciones son iteradas?
- Situaciones opcionales: ¿Qué ejecuciones alternativas se presentan en el Caso de Uso?

+ [4]Diagramas de Interacción de Objetos

Descripción de un escenario: un escenario muestra de que manera interactúan los distintos objetos dentro del flujo principal de eventos de un Caso de Uso con alguna variación o extensión concreta del mismo.

Existen dos tipos de diagramas de interacción: los Diagramas de Colaboración y los Diagramas de Secuencia.

Su finalidad es describir los mensajes que intercambian los distintos objetos para cumplir con las responsabilidades definidas en un escenario concreto.

, Diagramas de Secuencia

Representa las interacciones de objetos ordenadas en una serie temporal que muestra su ciclo de vida. Mensajes enviados entre los objetos descritos en el flujo de eventos de un Caso de Uso.

Estos mensajes muestran el nivel de colaboración entre los distintos objetos existentes e indican cuando se requiere generar nuevos objetos para cumplir con las responsabilidades asignadas.

, Diagramas de Colaboración

Representa una posible interacción de los objetos ordenados a partir de la topología que muestra el envío de sus mensajes.

[4] REF: http://www.vico.org

+ Diagramas de Clases y Objetos

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Estos diagramas presentan las clases y objetos del sistema con sus relaciones estructurales y de herencia.

El trabajo realizado en los Diagramas de Casos de Uso, Secuencia y Colaboración aportan información para establecer las clases, objetos, atributos y operaciones.

Clases: desde una perspectiva conceptual, una Clase representa un conjunto de Objetos que comparten:

- Las mismas propiedades (Atributos).
- El mismo comportamiento (Métodos).
- Las mismas relaciones con otros Objetos (Mensajes).
- La misma semántica dentro del sistema.

Desde una perspectiva física, una clase es una pieza de software que actúa como un molde para fabricar tipos particulares de objetos que disponen de los mismos atributos y métodos. Estos elementos que configuran cada tipo de objeto sólo se definen una vez, cuando se especifica la estructura de la clase a la que pertenecen.

Los objetos que se crean a partir de una clase concreta se llama *instancia* de esta clase y se diferencian entre ellos únicamente por los valores de sus atributos(variables).

Cada clase se representa por un rectángulo con tres compartimientos:

- Nombre de la clase
- Atributos de la clase
- Operaciones de la clase.

Motocicleta
Color Velocidad Máxima Cilindrada
Arrancar Acelerar Frenar

Objetos: un objeto representa una entidad del mundo real o algo inventado. Es un concepto, una abstracción algo que dispone de unos límites bien definidos y tiene una significación para el sistema que se pretende modelar.

Estructura y función:

- Identidad ¿Quién soy? = Atributos

- Propósito ¿Cuál es mi misión? = Justificación
- Responsabilidades ¿Qué debo hacer? = Métodos
- Procedencia ¿De qué Clase provengo? = Origen
- Relaciones ¿Qué mensajes entiendo? = Comportamiento

Abstracción: a partir de una abstracción del mundo real, se define objetos que representan micromódulos de software ideales. Desde su creación, se mantienen de manera independiente unos de otros, sólo interaccionan con otros objetos a través de sus mensajes. Cada objeto configura un universo ordenado y autosuficiente.

Mensaje: una aplicación orientada a objetos consiste en un número determinado de objetos que interactúan entre sí enviándose mensajes unos a otros para invocar sus métodos. Este intercambio de mensajes facilita su comportamiento, cambios de estado, destrucción o almacenamiento.

Ya que todo lo que un objeto puede realizar está expresado en sus métodos, este simple mecanismo de mensajes soporta todas las posibles interacciones entre ellos.

Encapsulación: el empaquetado de métodos y atributos dentro de un objeto mediante una interfaz de mensajes, es lo que se denomina encapsulación. La clave está precisamente en este envoltorio del objeto. La interfaz que rodea por completo al objeto actúa como punto de contacto para todos los mensajes que llegan desde cualquier objeto emisor.

La interfaz de mensajes tiene la misma función que la membrana de una célula, disponer de una barrera esencial entre la estructura interna de un objeto y el exterior.

Su propósito es garantizar que todas las interacciones del objeto tengan lugar a través de un sistema de mensajería predefinido con el que es capaz de entenderse y reaccionar adecuadamente. No existe otra manera con la que un objeto externo pueda acceder a los atributos y métodos escondidos dentro de la interfaz de mensajes de otro objeto.

Herencia: es el mecanismo por el cual una clase de objetos puede ser definida como un caso especial de otra clase más genérica. Los casos especiales de una clase se denominan *Subclases*.

La clase más genérica, se conoce como la *Superclase* de todos sus casos especiales. Además de los métodos y atributos que heredan, las Subclases definen sus propios métodos y atributos. También, pueden redefinir algunos de los heredados (overriding - sobremontando).

La interfaz de mensajes definida para una Superclase también es heredada por las subclases. Esto implica que todas las Subclases de una Superclase están preparadas para responder

correctamente a los mismos mensajes que la Clase padre. Esta propiedad es extremadamente útil porque permite tratar de la misma manera a todas las especializaciones de una Clase.

Composición: los objetos que contienen a otros objetos se denominan Objetos Compuestos. Los atributos de un objeto pueden utilizarse de dos maneras. Se puede usarlos para almacenar valores como el número 15, o bien, el texto "Autorizado".

También pueden contener referencias de otros objetos. Las referencias almacenadas en los atributos de un objeto compuesto, permiten a este objeto enviar los mensajes apropiados a todos los objetos contenidos.

Polimorfismo: diferentes objetos pueden responder a un mismo mensaje de diferentes maneras. El polimorfismo permite a los objetos interactuar entre ellos sin necesidad de conocer previamente a que tipo pertenece.

Un objeto puede ser de diferentes tipos. Por ejemplo un vehículo automático de carga puede especializarse para cargar bobinas u otro tipo de items. Los demás objetos del sistema no tienen por que saber cuantos tipos de vehículos existen.

El mismo mensaje cargarItem, acompañado del parámetro que identifica dicho ítem, será interpretado de distinta manera por cada objeto receptor, el cual ya conoce previamente como tiene que tratar la naturaleza de su carga: bobinas, etc.

Hay una deducción de esfuerzo muy significativa por el hecho de que cualquier objeto del sistema puede enviar mensajes a los vehículos automáticos de carga sin necesidad de saber de antemano que tipos de vehículos están en circulación.

No hay necesidad así de picar un código específico para un reporte, con lo cual, se ahorra prolijas sentencias IF o CASE que son complejas de mantener y de actualizar cuando se incorporan nuevos tipos de objetos.

Visibilidad: toda Clase encapsula elementos (atributos y operaciones) que disponen de ciertos criterios de visibilidad y manipulación para otras Clases.

Los elementos públicos pueden ser usados por cualquier otra Clase. Los elementos privados pueden ser usados sólo por la Clase propietaria. Cada plataforma de desarrollo (C++, Smalltalk, Java) desarrolla sus propias reglas con respecto a la visibilidad y manipulación de atributos y operaciones.

La notación UML especifica que todo atributo y operación de una Clase debe disponer de un indicador de visibilidad.

+ Diagramas de Transición de Estados

Descripción de la Dinámica del Sistema: la dinámica de un sistema está determinada por:

- Todos los posibles estados que sus objetos pueden tener.
- Todas las posibles secuencias de eventos que pueden ocurrir.
- Todas las transiciones posibles, de un estado a otro, como consecuencia de los eventos que afectan a los objetos.

Se utiliza el diagrama de estados para describir el comportamiento de una Clase dentro de una serie temporal. Se construye el diagrama de estados a partir de una clase concreta para mostrar el comportamiento de un objeto durante su ciclo de vida.

Un *Evento* no es un objeto. Un Evento es la "causa" que justifica la existencia de una información. Sólo se puede conocer que un Evento ha ocurrido, detectando sus "efectos" en el modelo. Sólo interesa aquellos Eventos que provocan un cambio de estado en los objetos del modelo. Hay que distinguir un Evento como tal, del objeto que representa el registro de los efectos de dicho Evento.

Los diagramas de estados son muy útiles para describir el comportamiento de un objeto a través de múltiples Casos de Uso.

+ Diagramas de Actividad

Descripción de un Flujo de Trabajo: un flujo de trabajo muestra la secuencia de actividades que se desarrollan dentro de uno o varios Casos de Uso, como una pieza de funcionalidad concreta que satisface los requerimientos de un Actor.

Diagrama de Actividad: un diagrama de actividad describe una secuencia de actividades que pueden exhibir un comportamiento en paralelo o estar sujetas a condiciones lógicas.

Los procesos de negocio no muestran siempre una secuencia fija en su desarrollo, es una ventaja así poder modelar bloques de actividades sin restricciones de concurrencia.

Un diagrama de actividad puede mostrar la secuencia de actividades que se desarrolla en un paquete de Casos de Uso que define un proceso de negocio y sus áreas de responsabilidad.

Su objetivo no es relacionar actividad con objetos, sólo comprender qué actividades son necesarias y cuales son sus relaciones de dependencia.

El diagrama de actividad permite definir en qué orden se van a realizar distintas tareas. Los diagramas de flujo (flowchart) sólo permiten modelar procesos secuenciales, mientras que los diagramas de actividad permiten establecer procesos en paralelo.

+ Arquitectura del Sistema

Una Arquitectura es un conjunto organizado de elementos. Se utiliza para especificar las decisiones estratégicas acerca de la estructura y funcionalidad del sistema, las colaboraciones entre sus distintos elementos y su despliegue físico para cumplir responsabilidades bien definidas.

, Modelo de Referencia

La vista del Modelo de Referencia (Reference Information Model), está determinada por la arquitectura lógica. La arquitectura lógica es capturada por los diagramas de Clases que contienen las Clases y relaciones que representan las abstracciones esenciales del sistema a desarrollar.

- Clases.
- Asociaciones.
- Agregaciones.
- Generalización.
- Paquetes.

, Vista de Componentes Modulares

La vista de Componentes Modulares refleja la organización de módulos de software dentro del entorno de desarrollo.

Esta vista de arquitectura toma en cuenta los requerimientos que facilitan la programación, los niveles de reutilización, y las limitaciones impuestas por el entorno de desarrollo. Se dispone de dos elementos para modelar esta vista:

- Paquetes: esta vista representa una partición física del sistema.
- Componentes: esta vista representa la organización de módulos de código fuente.

, Arquitectura del sistema o Vista de Distribución Física de Elementos

Esta vista presenta los componentes de software ejecutables con los nodos de procesamiento (hardware). Esta arquitectura tiene en cuenta los siguientes requerimientos:

- Disponibilidad del sistema.
- Rendimiento.
- Escalabilidad.

Los diagramas de distribución muestran el despliegue de nodos (locales y remotos), en la organización de la empresa. Permite al equipo de desarrollo comprender mejor la topología de un sistema distribuido.

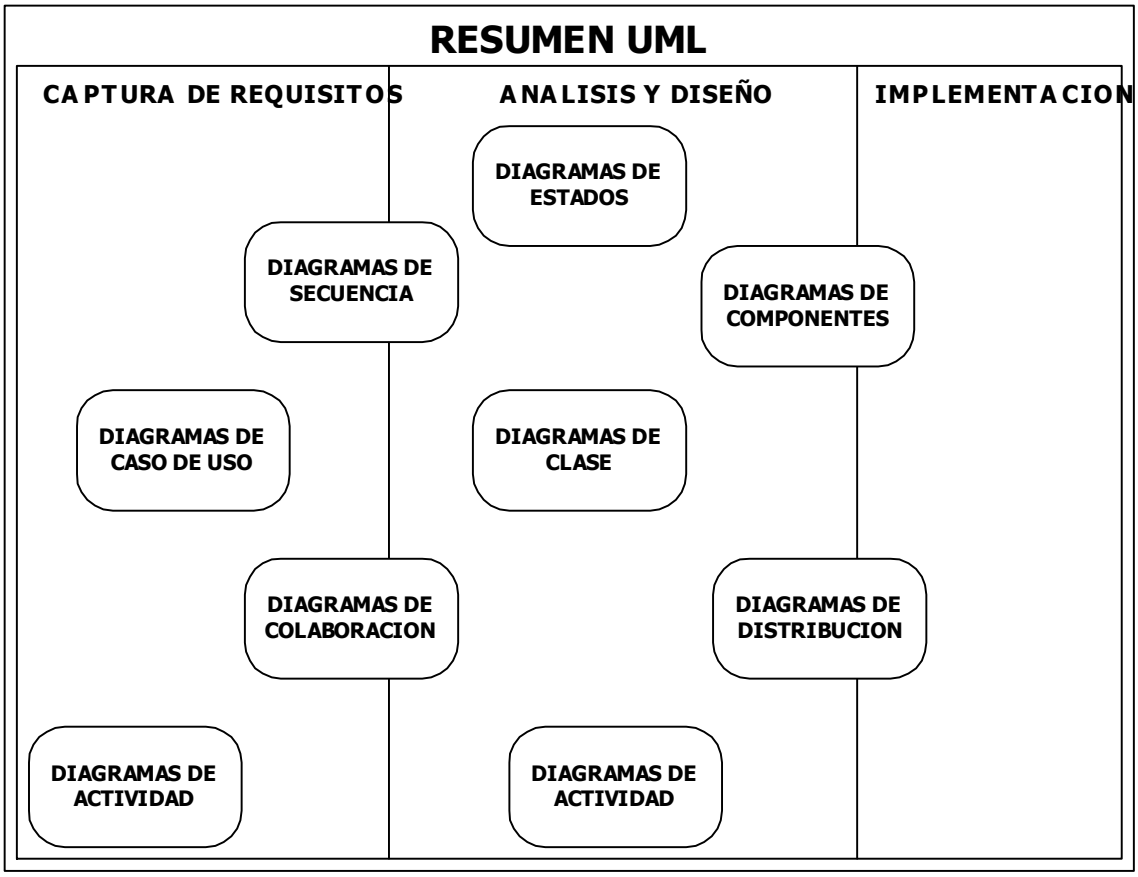
La Arquitectura del Sistema certifica la validez de:

- Modelo de Referencia.
- Componentes Modulares de Software.
- Ejecutables.
- Distribución de recursos informáticos.

Con la funcionalidad requerida del sistema. Se utiliza los siguientes elementos para describir la funcionalidad:

- Diagramas de Casos de Uso.
- Especificación de Casos de Uso.
- Diagramas de Interacción (Escenarios).
- Diagramas de Actividad (Flujos de Trabajo).
- Diagramas de Estados (Dinámica)

- **Resumen UML**



4.2.2. Ingeniería de Métodos

[5]La ingeniería de métodos es el enfoque coordinado y sistemático para establecer métodos de trabajo.

La flexibilidad sin control no se puede considerar como una metodología ya que carece de procedimientos prácticos sistemáticos y coordinados para el establecimiento de métodos de trabajo, en conclusión la ingeniería de métodos ayuda a que este método resulte sistemático y coordinado.

La ingeniería de métodos produce metodologías. En el caso de sistemas de información, una metodología es un conjunto de métodos empleado para el desarrollo de sistemas automatizados. La adaptación de una metodología a las necesidades de un proyecto a veces se conoce como *Ingeniería Situacional de Métodos*.

[5] REF: MARTIN, ODELL, Métodos orientados a objetos, consideraciones prácticas, Prentice Hall.

La ingeniería de métodos tiene varios grados de flexibilidad. Estos son:

- *La utilización de una metodología rígida* no permite prácticamente ninguna flexibilidad, estas metodologías están basadas en una sola filosofía de desarrollo y por esta razón adoptan estándares, procedimientos y técnicas fijas.
- *La selección a partir de metodologías rígidas* es una opción en la que cada uno de los proyectos escoge su metodología a partir de metodologías rígidas.
- *La selección de trayectorias dentro de una metodología* permite más flexibilidad al proporcionar dentro de las mismas metodologías una elección de trayectorias predefinidas (trayectorias que aceptan aspectos de desarrollo, como la selección de paquetes, proyectos pilotos, Cliente/Servidor, tiempo real...).
- *La selección y afinación de un esquema de métodos* permite a cada uno de los proyectos seleccionar métodos a partir de diferentes procedimientos y adecuarlos a las necesidades de los proyectos, es la opción más usada con herramientas automatizadas.
- *La construcción modular de métodos* es una de las opciones más flexibles ya que se genera para un proyecto dado a partir de bloques constructivos predefinidos, cada uno de los cuales se almacena en una base de métodos, dando como resultado una metodología eficaz, completa y consistente.

Método: es el conjunto de procedimientos sistemáticamente diseñados para lograr un objetivo.

Metodología: es el conjunto de métodos empleados por una disciplina.

Metodología de Ingeniería del Software: es un proceso para producir software de forma organizada, empleando una colección de técnicas y convenciones de notación predefinidas. La metodología suele presentarse como una serie de pasos, con técnicas y notaciones asociadas a cada paso.

4.3. CARACTERISTICAS DE LA METODOLOGIA

Para implementar escenarios virtuales es necesario enfrentarlos en forma metódica, para lo cual es necesario vincular los objetivos y conceptos para dar mayor comprensión y entendimiento del enfoque al que se desea llegar.

4.3.1. Objetivos de la Metodología

- Implementar escenarios virtuales no inmersivos con VRML visualizados a través de Internet, minimizando los recursos utilizados y su costo.
- Acortar el tiempo de análisis y diseño, haciendo que los escenarios virtuales se pueda implementar en plazos cada vez más cortos, manteniendo un funcionamiento aceptable y adecuado.
- Disminuir el flujo de datos y el tiempo de espera en Internet al cargar escenarios virtuales.
- Desarrollar escenarios virtuales, crearlo con objetos, definir las relaciones entre ellos y la naturaleza de las interacciones entre los mismos.
- Presenciar un objeto o estar dentro de él, es decir, entrar en el escenario virtual que sólo existirá en la memoria del observador (mientras lo observe) y en la memoria de la computadora.
- Interactuar varias personas en entornos que no existen en la realidad sino que han sido creados para distintos fines. En estos entornos el individuo debe preocuparse por actuar, ya que el espacio que antes se debía imaginar, es facilitado por medios tecnológicos.

4.4. ETAPAS DE LA METODOLOGIA

Una metodología es el conjunto de métodos sistemáticos para producir software o escenarios virtuales de forma organizada, empleando técnicas y convenciones. La metodología se presenta como una serie de pasos, con técnicas y notaciones asociadas a cada paso.

La metodología sé divide en:

- Estudio Preliminar.
- Captura de Requisitos.
- Análisis y Diseño.
- Implementación.
- Publicación, Pruebas y Mantenimiento.

4.5. ESTUDIO PRELIMINAR

Es el primer acercamiento para conocer la manera que está conformada la organización, especialmente en el área de publicidad y promoción de sus productos o servicios y la forma en que se da a conocer a los usuarios, para tener claro lo que se desea dar a conocer a través de la creación de escenarios virtuales.

El conocimiento del área informática de la empresa es importante, pues ellos son los llamados a estar pendientes de los cambios que existan y velar por el adelanto de la empresa para alcanzar las metas trazadas. De no existir el área informática dentro de la organización, debe estar pendiente una persona de alta gerencia con conocimientos en el área.

En esta primera parte de la metodología se define el equipo de trabajo.

4.5.1. Formación del Equipo de Trabajo

Para que todo proyecto salga adelante es necesario siempre formar un grupo de trabajo que se encargue de las diferentes fases del proyecto. La persona que lidera el proyecto debe escoger al personal idóneo y más calificado para formar el grupo de trabajo dependiendo del tipo y tamaño del proyecto.

En este equipo es necesario vincular al gerente de la empresa y al personal del departamento de informática si la empresa cuenta con este departamento.

El equipo de trabajo deberá escoger las estrategias que crea convenientes para la obtención de información necesaria para la realización del proyecto, para lo cual estará asistido y coordinado por el director del grupo. El equipo de trabajo dependiendo de las características y alcance del escenario virtual puede ser:

EQUIPO DE TRABAJO	
Cargo	**Descripción**
Director Proyecto	Dirección, coordinación y gestión del proyecto.
Jefe Laboratorio de Informática	Brindar facilidades, analizar requerimientos y administrar escenario virtual.
Gestores del Proyecto	Análisis, diseño e implementación de los escenarios virtuales.

4.6. CAPTURA DE REQUISITOS

La captura de requisitos es el primer paso para la creación del escenario virtual, en esta etapa es importante la definición del problema en un lenguaje natural, el cual, debe ser fácilmente interpretado por el cliente y desarrollador, es una forma de involucrar al cliente para que no se resista al cambio, ni argumente al finalizar el escenario virtual su inconformidad.

La deficiencia de una buena captura de requisitos del escenario virtual tiene como resultado retrasar la etapa del Análisis y Desarrollo así como la Implementación, especialmente en escenarios virtuales más complejos.

Se deben establecer los requisitos, esto se aplica tanto en la investigación de primera línea como en programas sencillos y personales, así como los esfuerzos realizados por grandes equipos. Cuando se tiene una idea vaga del objetivo, lo único que se está haciendo es posponer las decisiones a una etapa en la cual los cambios serán mucho más costosos.

La captura de requisitos del escenario virtual debe indicar lo que hay que hacer, y no como hay que hacerlo. Debe ser una exposición de las necesidades del cliente, y no de una propuesta de solución. El solicitante debe indicar que características son obligatorias, así como las que sean opcionales.

El desarrollador debe trabajar con el solicitante para refinar los requisitos, de tal forma que estos representen la verdadera intención del solicitante. Esto implica cuestionar los requisitos y buscar la información que falta.

Es importante tener en cuenta el siguiente consejo: "Si realiza exactamente lo que pide el cliente, pero el resultado no satisface sus necesidades reales, es muy probable que le echen la culpa a usted".

La captura de requisitos del escenario virtual no debe carecer de información esencial, debe ser obtenida a partir del solicitante o del conocimiento que el analista tenga del dominio del escenario virtual en el mundo real. El analista debe comunicarse con el solicitante para clarificar ambigüedades y errores conceptuales. El analista debe incluir aquella información que sea significativa desde el punto de vista real, y debe representar el aspecto externo del escenario virtual. De ello la captura de requisitos resulta comprensible para el cliente del escenario.

La Notación UML ha sido escogida en esta metodología, para la Captura de Requisitos y el Análisis y Diseño por ser un lenguaje de propósito general para el modelado orientado a objetos y modelar los sistemas independientemente del lenguaje de implementación.

Para la captura de requisitos primero se debe desarrollar un documento en el cual se plasme lo que se implementará, es decir, un anteproyecto. Para posteriormente crear los Diagramas de Casos de Uso, Secuencia y Colaboración que tienen un lenguaje natural entendible para el solicitante y desarrollador.

4.7. ANALISIS Y DISEÑO

El análisis es una representación precisa y concisa del escenario virtual, que permite responder a preguntas y construir una solución. El propósito del análisis es modelar el escenario virtual del mundo real para que sea posible entenderlo. El análisis facilita una comunicación precisa.

El diseño es el contraste del análisis, es la relevancia a efectos de implementación por computadora, por tanto debe de ser razonable, eficiente y práctico a la hora de codificar.

La etapa de diseño asociada a cualquier escenario virtual, es tal vez la más importante de la solución. Imagínese construyendo una casa sin los planos necesarios. Los resultados serian catastróficos. Lo mismo sucede al desarrollar un escenario virtual sin un buen plan.

Una vez que se ha analizado el escenario virtual, es preciso decidir la forma de aproximarse al diseño. El diseño del escenario virtual es resolver el problema y construir su solución. Este incluye decisiones acerca de la organización del escenario virtual en un diseño general de la escena, un diseño de los objetos que se encuentran dentro de escenario virtual, asignación del hardware, software, decisiones fundamentales y políticas de seguridad que son las que constituyen un marco de trabajo para el diseño detallado.

Durante el análisis, lo fundamental es hacer lo que necesita, independientemente de la forma de hacerlo. Durante el diseño, se toma decisiones acerca de la forma en que se resolverá la implementación del escenario virtual, primero desde un nivel general y después empleando el diseño de objetos en el escenario virtual que es más detallado.

Durante el diseño se decide la estructura y el estilo global. La estructura y el diseño global del escenario virtual, es la organización global del mismo en componentes llamados subescenarios, si es necesario subdividirlo por la complejidad del proyecto. El diseñador desglosa el escenario en subescenarios, de tal manera que sea posible realizar más trabajo por parte de varios implementadores que trabajarán independientemente en distintos subescenarios.

La Notación UML une estos dos conceptos análisis y diseño a través del Diagrama de Clases, que responde a preguntas y permite construir una solución razonable y práctica a la hora de la codificación.

Los Diagramas de Estado describen el comportamiento de las clases y los Diagramas de Actividad describen la secuencia de actividades, el orden en que se realizan las diferentes tareas y permiten establecer tareas en paralelo.

Los Diagramas de Componentes son los paquetes y módulos o componentes que se crean y utilizan para la implementación y los Diagramas de Distribución muestran el despliegue de nodos es decir los servidores y clientes en los cuales está y se ejecuta el sistema.

Estos diagramas son utilizados para el Análisis y Diseño del Escenario Virtual.

Los diagramas de Implementación de ejecutables no son necesarios por cuanto el lenguaje que se utiliza para la implantación es VRML y no genera ejecutables ni librerías, es un lenguaje que se interpreta, no se ejecuta.

4.8. IMPLEMENTACION

La implementación es la codificación del escenario virtual, debería ser una de las actividades más sencillas, siempre y cuando haya realizado un buen trabajo en las etapas de Captura de Requisitos y en el Análisis y Diseño de la solución. La codificación implica la escritura real del escenario virtual en el lenguaje de descripción de escena VRML 2.0.

La escritura de código debe ser sencilla, casi mecánica porque ya debería haber tomado en cuenta las decisiones difíciles durante el Análisis y Diseño. Ciertamente hay que tomar decisiones mientras se escribe el código, pero cada una de ellas debe afectar solamente a una pequeña parte del escenario virtual o de los subescenarios de tal manera que sea fácil cambiarlas. El código del escenario virtual es la personificación última de la solución del escenario virtual, así que la forma en que se escriba será importante para la mantenibilidad y extensibilidad.

El lenguaje utilizado para implementar el escenario virtual es VRML 2.0. (Lenguaje de Modelado de Realidad Virtual), es un lenguaje de descripción de escenas, que describe simulaciones interactivas, escenarios virtuales accesibles a través de Internet y vinculados con el web. Todos los aspectos de presentación de escenarios virtuales, interacción y conexión a la red pueden ser especificados por VRML. Puede ser utilizado para crear representaciones en tres dimensiones de escenas complejas en realidad virtual.

Aunque VRML es un lenguaje de computadora, no es un lenguaje de programación, los archivos VRML no se compilan, son archivos simples de texto que pueden ser leídos por intérpretes de VRML. Estos programas se denominan visores de VRML, ya que el código VRML se interpreta,

los resultados visibles cambian de visualizador a visualizador, de acuerdo a la forma en que se encuentren implementados.

Para la implementación de un escenario virtual se debe tener en cuenta un estilo de programación.

4.8.1. Estilo de Programación

[6]Como puede atestiguar cualquier jugador de ajedrez, cocinero o nadador, una cosa es conocer algo y otra hacerlo bien. La escritura de VRML no se escapa de esta circunstancia. No basta con conocer la estructura básica y como ser capaz de ensamblar para crear un escenario virtual. Los buenos programas hacen algo más que satisfacer simplemente sus requisitos funcionales. Los programas que siguen reglas de diseños correctas tienen más probabilidad de ser correctos, reutilizables, extensibles y fáciles de corregir. Ciertas técnicas que permiten un estilo de programación en VRML se describen a continuación:

- Nodo LOD, nivel de detalle, permite cambiar el detalle de los escenarios virtuales de acuerdo a la cercanía del usuario a los objetos y representaciones.
- Nodo WWWInline, que permite tener varios escenarios pequeños, distribuidos en diferentes lugares sobre Internet y que pueden ser llamados desde el escenario principal, es necesario utilizar estas técnicas, por cuanto para escenarios virtuales grandes y complejos se hace difícil la navegación para el usuario.
- Otra técnica es disminuir la complejidad del escenario VRML, reduciendo la cantidad de polígonos que tenga el escenario, esto permite aumentar la velocidad de navegación del usuario.
- La reutilización de objetos ayuda a mantener el tamaño del archivo, pequeño y puede hacer que el código VRML sea más fácil de escribir y mejoran la extensibilidad.
- Los escenarios VRML pueden ser comprimidos (extensión GZIP), de tal forma que algunos visualizadores VRML reconocen este tipo de archivos comprimidos y los visualizan. Pero esto tiene una desventaja ya que no todos los visualizadores tienen está propiedad.
- El implementador de escenarios virtuales debe tomar las decisiones siguientes:
    - Descomposición del escenario virtual en subescenarios.
    - Implementación de los objetos del escenario virtual.
    - Estimar las necesidades de rendimiento y los recursos necesarios.
    - Diseño de seguridades.

4.8.2. Descomposición del escenario virtual en subescenarios

[6] REF: http://www.vrml.org/VRML/FINAL

En todo escenario virtual, el primer paso para implementarlo consiste en dividirlo en un pequeño número de componentes. Cada uno de los componentes principales del escenario virtual será denominado subescenario. Cada subescenario abarca aspectos del escenario virtual que compartan alguna propiedad común (una funcionalidad similar, la misma ubicación física, o la ejecución en el mismo tipo de hardware). Por ejemplo, un escenario de una nave espacial podría contener un escenario virtual para la parte externa de la nave y otro escenario virtual para el interior de la nave espacial.

Un subescenario es un conjunto de objetos interrelacionados que permiten una interfaz bien definida para su navegación e inmersión. Cada subescenario se puede implementar independientemente, sin afectar a las demás.

4.8.3. Implementación de los objetos del escenario virtual

Los objetos descubiertos en el Análisis y Diseño deben ser implementados de la mejor forma con el objetivo de minimizar el tiempo de ejecución, la memoria y el costo. La optimización de recursos no debe llevarse a extremos exagerados, pues la facilidad de implementación, la mantenibilidad y la extensibilidad son también objetivos importantes.

En la implementación de objetos se deben determinar las representaciones de los objetos a través de fotografías y formas gráficas del objeto. Para ser implementados con VRML 2.0, siendo un lenguaje para la implementación de simulaciones interactivas en tres dimensiones visualizadas a través de Internet, con la habilidad de trabajar adecuadamente con conexiones pequeñas en ancho de banda, minimizando el costo de implementación.

Se debe considerar la cantidad de polígonos que se utiliza en la representación del objeto, así como su complejidad, es decir, no se debe utilizar demasiados polígonos o estructuras complejas que demoren la carga del escenario virtual.

Otra consideración a tomarse en cuenta es la facilidad de implementación y comprensibilidad, se puede implementar un objeto con polígonos y estructuras complejas disminuyendo algo del rendimiento en la carga del escenario virtual.

La flexibilidad es importante, por cuanto, un objeto muy optimizado suele sacrificar la legibilidad y la facilidad de cambio, pero un objeto complejo e ineficiente puede implementarse rápidamente.

4.8.4. Estimar las necesidades de rendimiento y recursos necesarios

Para la implementación de escenarios virtuales, es necesario contar con un espacio físico adecuado para la realización de las actividades, este deberá ser confortable, con ventilación y luz apropiada, permitiendo que ningún miembro del equipo se sienta apretado o encerrado, además se debe contar con suministros indispensables como papel, lápiz, bibliografía, etc.

El hardware para la implementación y visualización es importante. Se puede considerar, un PC con requerimientos mínimos: un procesador Pentium de 75MHz con 32 MB en RAM o su equivalente en otras plataformas, 50 MB libres en el disco duro y una conexión telefónica con un módem de 14.4 Kbps. Este computador puede ser utilizado para la redacción de documentos, implementación y como un cliente para la visualización del escenario virtual. Es preferible utilizar un PC con mayor capacidad de procesamiento, memoria y espacio en disco duro, así como una conexión con mayor ancho de banda, esto ayudará a que la interpretación sea más rápida. Una cámara digital o cámara normal para la recolección de información del escenario virtual. Un escáner para la digitalización de información. Una Intranet o Internet.

Estos son los requisitos mínimos para trabajar en un proyecto para implementar y visualizar un escenario virtual no inmersivo con VRML, dependiendo de la complejidad y extensibilidad del proyecto se debe incrementar el hardware.

Los componentes hardware para la creación de un escenario virtual inmersivo, son obtenidos de diversos fabricantes, para éstos propósitos es suficiente decir que el hardware para la Realidad Virtual va desde periféricos relativamente baratos para una computadora personal hasta sistemas que valen miles de dólares. Un buen número de sistemas independientes se une con la base de hardware, software y electrónica y otros proporcionan efectos auditivos visuales y táctiles como: cascos, guantes, trajes, dispositivos montados en la cabeza etc., este hardware es extremadamente costoso.

El software es una de las herramientas indispensables, consiste en programas que se compran o se los puede bajar de Internet sin costo adicional estos son: Procesadores de Texto, Paquetes de Diseño, Navegadores o Browser y Editores de Escenarios Virtuales.

Existen diferentes visores o browsers para poder experimentar con VRML en Internet, disponibles para diferentes plataformas aunque a veces un tipo de navegador no está disponible más que en una o dos plataformas, y en muchos casos las versiones están en etapas Alfa o Beta. Los diversos visores se distinguen por sus funcionalidades por lo que no es necesario escoger solamente uno. Las funcionalidades que deben tomarse en cuenta son:

- Plataforma.
- Versión.
- Número de colores.
- Velocidad.
- Tipo de sombreado.

- Soporte de texturas.
- Forma de despliegue.
- Posibilidad de modelar escenas.

Otro detalle a tomar en cuenta es que realmente a través de la aplicación se puede navegar en el Web con las ligas definidas en la escena tridimensional, dado que algunas aplicaciones sólo son visores, incapaces de navegar. Algunos ejemplos de navegadores son:

- WebSpace.
- WorldView.
- WebFX.
- Fountain.
- Virtus Voyager.
- VRweb.
- Cosmo Player.
- Live3D.
- Liquid Reality.
- Community Place.

Todos ellos están disponibles en Internet, algunos sin costo adicional y otros con un determinado costo.

Las herramientas de Edición para VRML sirven para crear archivos VRML, existen esencialmente dos maneras:

- La primera es hacerlo a mano, es decir, utilizar cualquier tipo de editor de texto y escribir el código del escenario virtual en VRML.
- La segunda y la más sencilla es usar una herramienta de edición, es decir, crear escenarios virtuales a partir de aplicaciones que generan código VRML. Esto implica que:
    - El autor gana mucho tiempo al no tener que escribir códigos.
    - El autor ve lo que está creando en ese mismo momento sin tener que hacer uso de un visualizador de VRML.
    - Lo más importante, el autor no tiene que conocer a fondo el lenguaje VRML para crear sus escenarios virtuales.

Para la publicación de un escenario virtual realizado en VRML no es necesario tener los diferentes subescenarios en varios servidores web, aunque esta tarea es posible, con los recursos que tiene un servidor web es suficiente para ponerlo en una dirección y acceder al escenario virtual a través de Internet.

Es importante decir que la implementación de escenarios virtuales en VRML 2.0, da como resultado archivos de texto pequeños ocupando espacios más pequeños que las páginas web que se encuentran en servidores web, creando una representación en tres dimensiones, con la cual se puede interactuar a través de Internet o una Intranet.

4.8.5. Diseño de seguridades

Para el diseño de seguridades es importante tener en cuenta qué escenario virtual se va a proteger. Determinar el costo de lo que se va a proteger y el costo que se requiere para implementar la seguridad. Si el costo de la implementación de la seguridad es mayor al costo de la información a proteger no tiene objetivo.

Es importante decir que los escenarios virtuales que se implementan en VRML 2.0, no se pueden realizar un download a través de Internet, pues éstos, son archivos de texto, que son interpretados por un visor y lo que realmente se ve no es real sino representaciones interactivas en tres dimensiones.

Al poner en línea un escenario virtual en Internet, las seguridades para proteger la información es trabajo de los encargados del mantenimiento de los servidores web, al igual que en una Intranet su administrador.

Para una mayor seguridad se puede diseñar una interfaz que pida una determinada clave, aunque esto no es necesario, pues un escenario virtual es un objeto web, como tal debe ser accesible para todos los usuarios de la red.

4.9. PUBLICACION, PRUEBAS Y MANTENIMIENTO

Al realizar la implementación del escenario virtual se debe tener en cuenta los consejos prácticos que se han dando durante la implementación, así como también ir realizando pruebas de velocidad de carga de los diferentes objetos del escenario virtual que se van creando.

[7]Una vez implementado el escenario virtual completamente es importante publicarlo y realizar las siguientes pruebas:

- Velocidad de carga y respuesta.
- Calidad de las imágenes proyectadas.
- Número de sentidos.
- Calidad con que se simulan.

[7] REF: http://www.monografias.com

- Calidad con que se logran los efectos de inmersión y manipulación del escenario virtual.

Estos parámetros son importantes para determinar la calidad de un escenario virtual.

4.9.1. Velocidad de carga y respuesta

La velocidad de carga se refiere a la cantidad de tiempo que utiliza un escenario virtual para visualizarse y ponerse a punto para la inmersión y navegación del usuario y la velocidad de respuesta es la cantidad de tiempo que utiliza para interactuar con los objetos del escenario virtual, es decir, si los objetos tienen eventos, con qué velocidad responden estas interacciones.

Mientras más rápido se cargué el escenario virtual, aunque se trabaje con un ancho de banda pequeño la velocidad de carga es buena, pero no hay que descuidar los otros parámetros de medición.

4.9.2. Calidad de las imágenes proyectadas

Es importante en un escenario virtual medir la calidad de las imágenes proyectadas al usuario, si realmente crean los efectos deseados en la visualización del ojo humano. Si las imágenes creadas tienen reacciones visuales realista en tiempo real.

Siempre existe una correspondencia entre velocidad y calidad de imagen. Los algoritmos que convierten datos de información en pequeños y brillantes píxeles dentro de la pantalla o en una escena virtual consumen, por lo general, gran cantidad de recursos de la computadora.

Las máquinas de más alta calidad pueden producir imágenes complejas y bien definidas, pero lentamente. Por supuesto, también pueden producir otras rápidamente, pero simples y borrosas. Pocas máquinas son capaces de producir rápidamente imágenes complejas de alta resolución.

Se debe tener en cuenta que la calidad y resolución de las imágenes creadas del escenario virtual sean las adecuadas, que no sean muy lentas en su carga y que la resolución no sea tan borrosa, así como también las características de hardware en los clientes que se van a visualizar.

4.9.3. Número de sentidos

La calidad de un escenario virtual también se mide por la cantidad de sentidos que se logre incorporar en la navegación del escenario virtual como: vista, olfato, gusto, tacto y oído.

Los cinco sentidos se logran incorporar solamente en un escenario virtual inmersivo utilizando cascos, guantes, trajes y otros dispositivos de costo elevado.

Los sentidos que se pueden incorporar en un escenario virtual realizado con VRML visualizado a través de Internet son: la vista y el oído, para lo cual no se requiere de equipos para realidad virtual, sino de los equipos tradicionales como: el monitor, el ratón o mouse y teclado bajando considerablemente el costo de implementación y visualización.

4.9.4. Calidad con que se simulan

La calidad con que se simulan los escenarios virtuales es buena y aceptable cuando cumplen con una velocidad de carga y respuesta apropiada, tienen una calidad de imágenes presentadas aceptable, una incorporación de varios sentidos y buenos efectos de inmersión y manipulación logrando un escenario virtual de calidad.

Es importante, el hardware que se utilice en el cliente, mientras más recursos de procesamiento, memoria, espacio en disco duro, cantidad de colores, es mejor; el escenario virtual se visualizará más rápido y con un mejor detalle de sus objetos.

4.9.5. Calidad con que se logran los efectos de inmersión y manipulación del escenario virtual

Un escenario virtual necesita proporcionar a los ojos del usuario imágenes de ángulo abierto, de alta resolución y bien enfocadas. Las imágenes para entornos virtuales deben crear escenas que sean detalladas de forma realista. Estas deben ser presentadas para corresponderse con escenas que el usuario vería si se moviese por ese entorno, como si éste fuese real.

Estas imágenes presentadas deben dar una Inmersión adecuada, mediante la cual el usuario tenga la sensación de encontrarse dentro de un escenario tridimensional.

4.9.6. Mantenimiento

Un escenario virtual se crea con el objetivo de crear objetos cada vez más reales y promocionarlos a través de Internet, aunque es una tecnología novedosa y actual no quiere decir que los escenarios creados no estén sujetos a cambios y mejoras.

Los cambios y mejoras son inevitables cuando se construye un escenario virtual, por tal motivo debe desarrollar mecanismos para evaluar, controlar y realizar mejoras. Se puede definir el mantenimiento de un escenario virtual en tres actividades:

- Mantenimiento Correctivo: se refiere a corregir errores que se puedan encontrar en el escenario virtual, mientras esté publicado.
- Mejoras o Actualizaciones: todos los escenarios virtuales creados no siempre van a ser los mismos, están sujetos a cambios y perfeccionamientos, o también a mejoras o ampliaciones del escenario virtual, creando nuevos subescenarios.
- Mantenimiento Preventivo: se refiere a la verificación y control del escenario virtual aunque esté no falle, con esto se pretende prevenir posibles inconvenientes que puedan presentarse.

4.10. NOTAS BIBLIOGRAFICAS

En este capítulo se exponen criterios que se utilizará en el desarrollo de la metodología, si desea más información visite las siguientes direcciones web y revise los libros listados que le van a ser de mucha utilidad:

- http://www.dsic.upv.es/~uml
  Fecha último ingreso: 2002-09-12
- http://www.vicio.org
  Fecha último ingreso: 2002-09-12
- http://www.gateland.cl/VRML%2022.htm
  Fecha último ingreso: 2002-09-12
- http://www.vrml.org/VRML/FINAL
  Fecha último ingreso: 2002-09-12
- http://www.monografias.com/
  Fecha último ingreso: 2002-09-12

Libros

- James Martín, James F. Odell, "Métodos orientados a objetos, consideraciones prácticas", Ed. Prentice Hall, Primera Edición, 1997.
- Andrew C. Staugaard, Jr., "Técnicas Estructuradas y Orientadas a Objetos", Ed. Prentice Hall, Segunda Edición, 1998.
- James Rumbaugh, Michael Blaha, William Pre Merlani, Frederick Eddy, William Lorense, "Modelado y Diseño Orientados a Objetos. Metodología OMT", Ed. Prentice Hall, Primera Edición, 1996.
- Henry F. Korth, Abraham Silberschatz, "Fundamentos de Bases de Datos", Ed. Mc Graw Hill, Segunda Edición, 1993.
- Roger S. Pressman, "Ingeniería del Software", Editora Mc Graw Hill, Cuarta Edición, 1998.
- Linda Gail, John Christie, "Enciclopedia de Términos de Computación", Ed. Pentice Hall.