Universidad de las Ciencias Informáticas

Facultad 5

Trabajo de Diploma

“Ingeniería de Dominio para la línea de productos Videojuegos del Centro de Desarrollo de Informática Industrial”

Autora: Sindy Chirino Cordero.

Tutor: Ing. Adiel Durán Rodríguez.

La Habana, 2010.

“Año del 53 de la Revolución”

“Dime y lo olvido, enséñame y lo recuerdo,

involúcrame y lo aprendo”.

Benjamín Franklin

Declaración de Autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de _____ del año _____.

Autora: Tutor:

_____ _____

Sindy Chirino Cordero Ing. Adiel Durán Rodríguez

Datos del Contacto

Nombre y Apellidos: Adiel Durán Rodríguez.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias Informática. Categoría Docente: Profesor.

e-mail: aduranr@uci.cu

Dedicatoria

A los que ven en mi compañía un premio,

a los que están a mi lado en los buenos y malos momentos,

a los que estarán siempre en mi pensamiento por ser las personas más importantes en mi vida,

a los que lloran de alegría con mis triunfos,

a aquellos que me apoyan aunque este errada y permiten que aprenda siempre más de la vida,

a las personas que han marcado una pauta en mí,

en fin a mi familia

y especialmente; dedicado a mis padres.

Agradecimientos

Al verme convertida hoy en una Ingeniera en Ciencias Informáticas me siento comprometida en agradecer primeramente a las personas que hicieron posible que esto sucediera al permitirme nacer y darme una adecuada educación inculcando en todo momento la necesidad de estudiar y ser alguien en la vida; a mis padres.

A mi familia en general porque sé que aunque no están presentes aquí, hoy se sienten orgullosos de mí.

A mi novio, que no se si las palabras que le dedique expresen todo lo que se ha esforzado para que este día sea uno de los mejores de mi vida, por aguantarme mis malcriadeces, por su compañía incondicional y su apoyo desinteresado durante todos estos años, por su comprensión, por brindarme tantos momentos de alegría y acompañarme siempre en cada situación que se me ha presentado dándome fuerza para continuar, porque es mi punto de partida para cada decisión. Aunque se pudiera resumir en los años que hemos pasado juntos pero también te agradezco por los que nos faltan por pasar.

A la familia de mi novio que durante mi carrera me ha apoyado en todas las decisiones ayudándome en todo lo que ha estado a su alcance.

A Alexander por acompañarme en muchos momentos difíciles de mi vida y estar presente cuando lo he necesitado. Convirtiéndose en una de las personas más allegadas a mí.

A aquellos profesores que logran convertir la palabra estudiante en amigo especialmente a Adiel, Léster , Sayli, Belkys y Minardo. Agradeciendo además a los que contribuyeron a mi formación como estudiante.

A los compañeros de siempre a Alianna, Bárbara, Mercedes, Naylen, Islema, Yuniel, Maylín, Cervela y Yenita.

A Annia por ser mi más que mi amiga como mi hermana. Gracias por la amistad que permitiste crear día a día sin haber tenido jamás alguna diferencia de criterios.

Resumen

La evolución de las tecnologías ha propiciado actualmente la innovación en la producción de muchas empresas relacionadas al tema. Por tal motivo en la UCI se realiza un continuo perfeccionamiento en la actividad productiva con el propósito de lograr un aumento acelerado de la calidad de sus productos. La presente investigación surgió por la necesidad en el CEDIN, un centro de desarrollo de software de la universidad, de establecer un método para guiar la fase inicial del desarrollo de software en la línea de productos videojuegos.

Como solución se seleccionó un modelo que consta de tres etapas de las cuales se desarrollará en gran medida la primera, que responde al objetivo planteado. Por lo que se identifican los elementos fundamentales de la Ingeniería de dominio que permitirán un alto grado de reutilización en el desarrollo de los videojuegos en el centro. Obteniendo así una especificación global de los requisitos que debe cumplir un videojuego, un modelado de dominio para un mayor entendimiento del problema planteado y los principales elementos para definir una arquitectura que enmarque parámetros esenciales para el futuro desarrollo de cualquier videojuego que se vaya a producir.

Palabras Clave

Arquitectura de software, estilo, modelo, líneas de productos de software, patrón, requisitos.

Tabla de contenido

Tabla de Contenido

Tabla de Figuras

Índice de Tablas

Introducción

La informática abarca un conjunto de conocimientos y herramientas científicas, técnicas y tecnológicas que, para enmarcar su utilidad, se podría decir que se encarga del tratamiento racional y estructurado de la información por medios automáticos electrónicos digitales. En la informática convergen los fundamentos de las ciencias de la computación, la programación y las metodologías para el proceso de desarrollo de software, la arquitectura de computadores, las redes de datos como Internet, la inteligencia artificial, así como determinados temas de electrónica. Se puede entender por informática a la unión de todo este conjunto de disciplinas. En pocas palabras se puede decir que para el mundo de hoy, la informática está siendo el motor impulsor de la más profunda revolución tecnológica conocida, transformando la vida social en lo que se ha dado por llamar: Sociedad de la Información.

Esta revolución tecnológica se debe fundamentalmente a los avances alcanzados en la industria del *software* ya que es posible convertir ideas en productos con pocos recursos, si se compara con otras industrias. La industria del *software* y los servicios informáticos han efectuado un gran salto en la calidad de los productos, que se ve reflejado en el creciente interés de empresas importantes de la mayoría de los países por adquirir tecnología nacional, y que se materializa con el perfeccionamiento creciente de los procesos de desarrollo de *software* para la producción.

Es importante destacar que un proceso de desarrollo de *software* tiene como propósito la producción eficaz y eficiente de un producto *software* que cumpla las expectativas del cliente especificadas en requisitos. Teniendo en cuenta que el proceso de desarrollo de *software* no es único y no existe un proceso de desarrollo de *software* universal que sea efectivo para todos los contextos de proyectos de desarrollo, es difícil automatizar todo un proceso de desarrollo de *software*. Sobre la naturaleza caótica del *software* se encontró una importante referencia en el artículo (Brooks, 1975) donde se analizan las características del *software*. En el mismo, el autor tomaba la realidad de la disciplina mencionando los grandes problemas que planteaba el desarrollo de *software*. Motivo por

el cual surge la necesidad de la estandarización, materializada con la aparición de modelos de desarrollo de *software*.

Según bibliografía consultada se define modelo de proceso de desarrollo de *software* como: “Una representación simplificada de un proceso de *software*, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del *software* es una abstracción de un proceso real.”(Sommerville, 2002)

En la actualidad múltiples empresas se especializan en el desarrollo de software, específicamente la Universidad de las Ciencias Informáticas representa una potente industria de software en el país. En ésta, la creación de software surgió de forma empírica por lo que continuamente se propone mejorar la calidad de sus productos para alcanzar un lugar en el mercado mundial. Una de las técnicas empleadas para lograr su propósito ha sido sin dudas la creación de centros de desarrollo de software especializados. Se realizó una restructuración en la actividad productiva en todas las facultades que la conforman. La facultad 5 fue dividida en 2 centros de este tipo uno de estos es el Centro de Desarrollo de Informática Industrial que está formado por 5 líneas de productos; una de ellas es la línea de videojuegos en la cual se han detectado diferentes anomalías referentes al proceso de desarrollo de este tipo de software.

Es importante aclarar que el proceso de creación de los videojuegos en particular, es similar a la creación de software en general. Este difiere en la cantidad de aportes creativos necesarios. El desarrollo de estos también varía en función a la plataforma objetivo, el género y la forma de visualización. Teniendo en cuenta lo anterior es notable la inexistencia en el centro de una forma de producir software que se ajuste a las características particulares de los videojuegos. Para la línea de videojuegos no están identificadas todas las posibles funcionalidades que pueda tener un videojuego para satisfacer las necesidades de los clientes. Además se nota la ausencia de una estructura que soporte el desarrollo de estos para organizar las etapas tempranas de su creación.

Analizando la situación problémica anteriormente descrita se plantea como **problema científico**.

¿Cómo guiar el proceso inicial de realización de los productos de la línea de videojuegos del CEDIN?

Partiendo de este problema se hace imprescindible que el **objeto de estudio** sea: ¨los modelos de desarrollo de software¨.

Teniendo esta investigación el siguiente **objetivo general**:

Definir los elementos principales de un modelo de desarrollo de software para guiar el flujo de producción en la línea de productos de videojuegos del CEDIN.

Para centrarse en el **campo de acción** ¨ Ingeniería de dominio en los modelo de desarrollo de software para la línea de productos videojuegos del CEDIN¨.

Tomando como **idea a defender** que:

Con la definición de los activos más importantes de la ingeniería de dominio de un modelo de desarrollo de software para la línea de productos videojuegos, se debe lograr mayor organización en el proceso inicial de la realización de este tipo de software en el CEDIN.

Para dar solución al problema planteado y cumplir el objetivo trazado se asumen como **tareas investigativas** las siguientes:

- Estudio de los modelos de procesos de desarrollo de software para realizar la selección de uno ellos.
- Estudio del estado del arte de los videojuegos para identificar aspectos comunes en el proceso de desarrollo de estos.
- Aplicación de elementos del modelo seleccionado en la línea de videojuegos.
- Estructuración de un modelo de dominio para tener un mayor conocimiento de de la lógica del proceso de desarrollo de los videojuegos en el centro.
- Identificación de requisitos comunes en los videojuegos.

- Estudio del estado del arte de la arquitectura de software para proponer una arquitectura candidata a aplicar en la línea de videojuegos.
- Validación de la propuesta mediante técnicas de validación.

Los **Métodos de Trabajo Científico** utilizados son los siguientes:

Métodos teóricos

El método **histórico-lógico** y el **dialéctico** para el estudio crítico de los antecedentes de los elementos a desarrollar utilizando estos como punto de referencia y comparación de los resultados a alcanzar.

El método **analítico-sintético** al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta, logrando definir los principales aspectos para la primera fase de desarrollo de los videojuegos del centro.

Métodos Empíricos

El método de **la entrevista** para obtener los principales problemas que existen en los proyectos de la línea de productos videojuegos. Además para conocer el criterio de los especialistas del tema sobre la propuesta brindada.

Resultados Esperados

La **actualidad de la investigación** se puede apreciar en que es la primera vez que se propone un modelo de procesos de desarrollo de software para la línea de productos de videojuegos del CEDIN y además son estructurados los elementos principales de la Ingeniería de dominio para la fase inicial del desarrollo de los mismos.

La **novedad científica** de la investigación radica en el hecho del amplio análisis bibliográfico que la autora ha realizado y sobre todo que en ninguno de los videojuegos desarrollados en el CEDIN se ha utilizado en su proceso de desarrollo un modelo de procesos de desarrollo de software.

La **significación práctica** estará dada en que se ofrecerá una guía para organizar el proceso inicial del desarrollo de los productos de la línea de videojuegos del CEDIN.

El presente trabajo de diploma se estructura de la siguiente forma:

En el **Capítulo I "Fundamentación Teórica"** se exponen los postulados, principios, categorías y conceptos que sirven de referentes teóricos a la investigación. Con el fin de lograr un mayor entendimiento de los temas a tratar en la propuesta.

En el **Capítulo II "Propuesta de solución"** se propone un modelo de procesos de desarrollo de software aplicando de este elementos de la Ingeniería de dominio a la producción de videojuegos, se conforma una lista de requisitos funcionales comunes a todos, un modelo de dominio para la comprensión del problema y como otro elemento importante, se propone una línea base de arquitectura como estructura candidata para el desarrollo de videojuegos en el centro. Siendo estos aspectos la solución propuesta.

En el **Capítulo III "Validación de la propuesta"** se presentan los resultados de la validación de la solución propuesta. Se selecciona una técnica de validación para obtener la aceptación de la propuesta realizada. Los resultados obtenidos se reflejaran graficados para brindar una mayor visibilidad del análisis realizado. Dichos resultados se obtendrán mediante cálculos probabilísticos.

Capítulo 1. Fundamentación Teórica

Introducción

En la actualidad la evolución continua de la ingeniería de software ha logrado estandarizar el desarrollo de software mediante la adopción de modelos de procesos.

El presente capítulo está enfocado a profundizar temas relacionados con la existencia de modelos de procesos de desarrollo de software para guiar el ciclo de vida de los mismos, el funcionamiento de las líneas de productos como nuevo paradigma para las empresas productoras de software, así como los principales elementos de la ingeniería de dominio en una familia de software.

1.1 Líneas de productos de software (LPS)

Partiendo de las diferentes conceptualizaciones existentes de líneas de productos de software:

"... es un conjunto de sistemas de software que comparten un conjunto común y gestionado de aspectos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales de software de una manera prescrita". (Clements and Northrop, 2002)

"...consiste de una familia de sistemas de software que tienen una funcionalidad común y alguna funcionalidad variable". (Gonma, 2004)

"...se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción". (Krueger, 2006)

Muchos han sido los autores que han realizado investigaciones referentes a aumentar la calidad del proceso de desarrollo de software y que esto influya en el resultado final. Es notable que el interés por variar la forma tradicional de hacer software despertara hace algún tiempo. Se puede afirmar, teniendo en cuenta los antecedentes expuestos, que una LPS es un conjunto de sistemas de software compartiendo características comunes y administradas, que satisface las necesidades específicas de un segmento de mercado particular y que son desarrolladas de forma prescrita a partir de un conjunto común de elementos clave. En la figura 1 se esquematiza la relación existente entre las partes que forman una línea de producto de software.

1.1.1 Representación básica de una línea de productos de software.

Figura 1.Representación Básica de una LPS.(Montilva,2006)

Activos de software: Una colección de partes de software (requisitos, diseños, componentes, casos de prueba, etc.) que se configuran y componen de una manera prescrita para producir los productos de la línea.

Producción:

- Establece los mecanismos o pasos para componer y configurar productos a partir de los activos de software de entrada.
- Las decisiones del producto se usan para determinar que activos de entrada utilizar y como configurar los puntos de variación de esos activos.

Decisiones de Productos:

- Los Modelos de Decisiones describen los aspectos variables y opcionales de los productos de la línea.
- Cada producto de la línea es definido por un conjunto de decisiones (decisiones del producto).

Productos de software: productos que pueden o son producidos por la línea de productos.

El paradigma línea de productos de software está dividido en dos partes fundamentales: la ingeniería de dominio e ingeniería de aplicaciones:

Ingeniería de Dominio: Conjunto de actividades relacionadas con la obtención de una infraestructura que proporcione las bases para la reutilización sistemática. (fraques et. Al, 1998)Provee a las LPS de los activos de software reutilizables para el desarrollo de sus productos.

Ingeniería de Aplicaciones: Se encarga del desarrollo de los productos de la LPS a través de los activos de software reutilizables aplicando planes de producción. (Montilva, 2006)Toma como referencia de diseño la arquitectura de dominio.

Para el análisis de la ingeniería de dominio es necesario comprender en qué consiste un dominio.

Dominio: Es un área o ámbito de aplicación de productos software. (Rebeca, 2002) Constituye un conjunto de especificaciones y elementos susceptibles de ser tratados de manera global bajo un proceso industrial, que engloba las similitudes y diferencias. Un Dominio contiene una maquinaria específica que produce software según normas y

destinada a resolver un determinado conjunto de problemas dentro de las especificaciones técnicas o funcionales que caracterizan lo caracterizan.

El análisis de un dominio es realizado por los analistas e ingenieros de dominio partiendo de implementaciones o aplicaciones ya existentes y toda la información que éstas conllevan (código, documentación de requisitos y funcionalidad, diseño, manuales de usuario, etc.), de requisitos actuales y futuros y de otras documentaciones técnicas relacionadas con el ámbito de aplicación. Siguiendo unas pautas establecidas para el análisis, se extrae toda la información relevante y, tras realizar tareas de abstracción, se encapsula y organiza para obtener estándares, arquitecturas de dominio y modelos de dominio. (Rebeca, 2002)

El desarrollo de software basado en líneas de productos evidencia un alto grado de reutilización ofreciendo mejoras en los tiempos de entrega y en la calidad de los productos. Para realizar una eficiente adopción de este nuevo paradigma es necesario realizar un estudio de la familia de software.

1.2 Proceso de desarrollo de los videojuegos

Para tener una idea más clara sobre los videojuegos se debe conocer el significado de los mismos: un videojuego se define como un programa informático que sirve para entretener a sus usuarios, se basa en la interacción entre una o varias personas y un dispositivo electrónico. (Frasca, 2003) El dispositivo que ejecuta el juego, puede ser una computadora, una videoconsola o un celular. No todos los videojuegos sirven para entretener; también los hay que pueden crear toda una variedad de emociones, como, por ejemplo: miedo.

El proceso de desarrollo de un videojuego no es tarea fácil debido al número de perfiles que engloba, por lo tanto, es necesario el desarrollo de herramientas de comunicación eficientes que aglutinen estos elementos de forma organizada. Por otra parte, el establecimiento de una metodología para desarrollar videojuegos es de suma importancia. Sobre esto no se ha estandarizado nada de forma general solo se han realizado trabajos que lo hace un juego en particular.

1.3 Modelos de procesos de desarrollo de software

1.3.1 Antecedentes

En los inicios de la producción de software el orden de los pasos era fabricar primeramente algún código y luego pensar sobre los requerimientos, diseño, prueba y mantención del producto. Esta forma traía consigo la dificultad de presentar una baja estructuración del código luego de alguna cantidad de fijaciones; pese a que se podía desarrollar un software de calidad, era posible que éste tuviera una correspondencia muy pobre con las reales necesidades del usuario y, finalmente, si no existía la conciencia de la necesidad real, así como de las pruebas y modificaciones, el costo del trabajo sería muy alto.

Al analizar diferentes criterios relacionados con la utilización de los modelos de procesos en el desarrollo de software, se puede comprender que una aplicación eficiente de estos provee un aumento considerable en la calidad de los productos de software. Puesto que constituyen una guía que logra organizar los procesos que intervienen en la realización de cualquier producto de software. Actualmente coexisten múltiples modelos de procesos, a continuación se expondrán algunos de los más usados por grandes, medianas y pequeñas empresas de software.

1.3.2 Modelos de procesos de desarrollo de software tradicionales.

En esta sección se da una visión general de los modelos de procesos de desarrollo de software que se utilizan en la forma tradicional de hacer software.

Modelo Secuencial

El modelo de secuencias sigue una sistemática y secuencial aproximación que comienza en el nivel de sistema y progresa exitosamente desde el análisis hasta las pruebas. Cada actividad es considerada como concluida antes de que la próxima actividad comience. La salida de una actividad es la entrada para la otra. Este acercamiento al desarrollo de software es el más viejo y el más ampliamente usado. Se basa en la noción de que es posible definir y describir todos los requerimientos del sistema y características del

software de antemano, o al menos bastante temprano en el proceso de desarrollo. Muchos problemas mayores en la ingeniería de software surgen del hecho de que es difícil para los clientes definir todos los requerimientos explícitamente. Un modelo de secuencia requiere una completa especificación de requisitos y ahí puede haber dificultades en acomodar la incertidumbre natural que existe en el comienzo de muchos proyectos. Además, es difícil agregar o modificar requerimientos durante el desarrollo porque, una vez realizado, las actividades se consideran como terminadas. En la práctica los requerimientos cambiarán y nuevas funcionalidades serán definidas; una aproximación puramente secuencial al desarrollo de software puede ser difícil y en muchos casos insatisfactorio. Otro problema que se encuentra a veces cuando se aplica un modelo de secuencia es la tardía respuesta de los clientes. Una versión que trabaje del software no estará disponible hasta tarde en el proceso de desarrollo del sistema y un mayor desconocimiento, si no se detecta hasta que el programa final sea revisado, puede ser desastroso.

Luego de un análisis de este modelo y teniendo en cuenta las características del desarrollo de videojuegos no sería factible utilizarlo porque contiene algunos elementos incompatibles con los mismos. Se pudiera citar principalmente que presenta problemas en acomodar la incertidumbre que se genera normalmente en el inicio de todo software, dificultando la posibilidad de agregar o modificar nuevas funcionalidades pedidas por el cliente.

Modelo Incremental

El modelo incremental combina elementos del modelo secuencial (aplicado repetitivamente) con la aproximación iterativa. El modelo incremental aplica el modelo de secuencias en etapas como el progreso del tiempo del calendario. Cada secuencia produce un incremento entregable del software con nuevas funcionalidades adicionadas al sistema. Cuando el modelo incremental es usado, el primer incremento es a menudo el núcleo del sistema de software. Los requisitos básicos son direccionados, pero muchas características suplementarias se mantienen sin entregar.

Este proceso es repetido siguiendo la entrega de cada incremento, hasta que el sistema de software completo sea desarrollado. El modelo incremental es particularmente útil para

manejar cambios en los requerimientos. Tempranos incrementos pueden ser implementados para satisfacer requerimientos, dichos requerimientos pueden ser mejorados en incrementos posteriores. Si el núcleo del software es bien recibido, el próximo incremento puede ser adicionado de acuerdo con el plan. Los incrementos pueden ser planeados además para administrar riesgos técnicos.

El modelo incremental a pesar de estar combinado con el secuencial no responde a la totalidad de particularidades que presenta el desarrollo de videojuegos pasando por alto esencialmente el alto grado de reutilización que necesitan estos productos.

Modelo de Prototipo

El modelo evolutivo para desarrollo de software está basado en la creación de un prototipo bastante temprano en las actividades de desarrollo. El prototipo es una versión preliminar, o intencionalmente incompleta, del sistema completo, el prototipo es entonces gradualmente mejorado y desarrollado hasta que alcanza un nivel de aceptación decidido por el usuario. Se trata de un ciclo de vida de software basado en componentes. Se recomienda usar este modelo cuando:

- No se entienda la aplicación del software a desarrollar, por eso este modelo ayuda a experimentar y/o probar.
- A veces suelen terminar el contrato cuando el proceso de producción está bien avanzado, para evitar esto se realiza un prototipo de bajo costo y en corto tiempo.
- A veces no se conoce la reacción de los usuarios para con el nuevo software, entonces un prototipo ayudaría para probar diferentes diseños.

Teniendo en cuenta que el proceso de realización de los videojuegos puede ser desde lo más sencillo hasta lo más complejo este modelo no se adapta al mismo porque interfiere en el parámetro tiempo del desarrollo de estos productos, que actualmente se ve afectado.

Modelo Iterativo

El acercamiento iterativo al desarrollo de software está basado en el modelo secuencial suplido con la posibilidad de retornar a actividades previas. Cada iteración dirige el sistema completo e incrementa la funcionalidad de las partes del sistema. Este acercamiento permite futuros refinamientos de los requerimientos del sistema, lo que incrementa la posibilidad de administrar los requerimientos de bajo nivel. Una desventaja específica de usar este acercamiento es la imposibilidad de predecir la fecha de realización y el costo de la implementación final.

En el estudio de este modelo se tuvo en cuenta los aspectos positivos que favorecerían al desarrollo de los videojuegos pero al analizar su principal desventaja es imposible la adopción de este modelo por su directa influencia en la fecha de entrega de los productos.

Modelo en Cascada

Se centra principalmente en la separación de las distintas fases la especificación y desarrollo del software. Estas fases son:

- Análisis de requerimientos y definición.
- Diseño del sistema y del software.
- Implementación y prueba de unidades
- Integración y prueba del sistema.
- Operación y mantenimiento.

Según Sommerville, 2002 la desventaja principal de este modelo reside en la dificultad de realizar cambios entre etapas, aspectos que hace que no sea adaptable este modelo en la elaboración de los videojuegos pues sería más factible la utilización de uno que brinde mayor flexibilidad.

Modelo de Desarrollo Evolutivo

Este modelo de procesos de desarrollo de software se aplica a sistemas interactivos generalmente pequeños o medianos en ocasiones partes de sistemas de software

grandes .Esto se debe a la poca visibilidad que brinda en el proceso, los sistemas están pobremente especificados, requiriéndose habilidades adicionales.
Como se evidencia en la breve descripción anterior este modelo no reúne las condiciones deseadas para estandarizar los videojuegos teniendo en cuenta que estos pueden ser de cualquier dimensión dependiendo solamente del guión utilizado.

Modelo en Espiral

Centra su atención en la reutilización de componentes y eliminación de errores en información descubierta en fases iniciales. Los objetivos de calidad son prioridad en este modelo para lo cual integra desarrollo del software con el mantenimiento del mismo. Además provee un marco de desarrollo de Hardware/Software. En el estudio de este se puede identificar que requiere de experiencia en la identificación de riesgos y refinamiento para su uso generalizado, aspecto que puede resultar como inconveniente al ser utilizado por los videojuegos teniendo en cuenta que la universidad presenta un alto grado de vinculación docencia-producción y en ocasiones no se cuenta con personal especializado.

1.3.3 Modelos de procesos para líneas de productos de software

TWIN: Modelo de desarrollo de software basado en componentes.

Integra tres procesos complementarios asociados al desarrollo de software basado en reutilización de componentes. (Montilva, 2006)Comprende la ingeniería de dominio, la ingeniería de componentes y finalmente la ingeniería de aplicaciones. Estas etapas funcionan de forma coordinada, favoreciendo a un alto grado de reutilización lo que es uno de los aspectos que se persigue en la producción de videojuegos. Luego de haber analizado a este tipo de software y su desarrollo en el centro existe una característica de este modelo que no se corresponde a lo estudiado. Esta no es más que la orientación que tiene el mismo a la fabricación de componentes y que la reutilización ocurra basada en este sentido. En desarrollo de videojuegos en el centro está centrado en la recopilación de activos de software reutilizables como base a futuras aplicaciones.

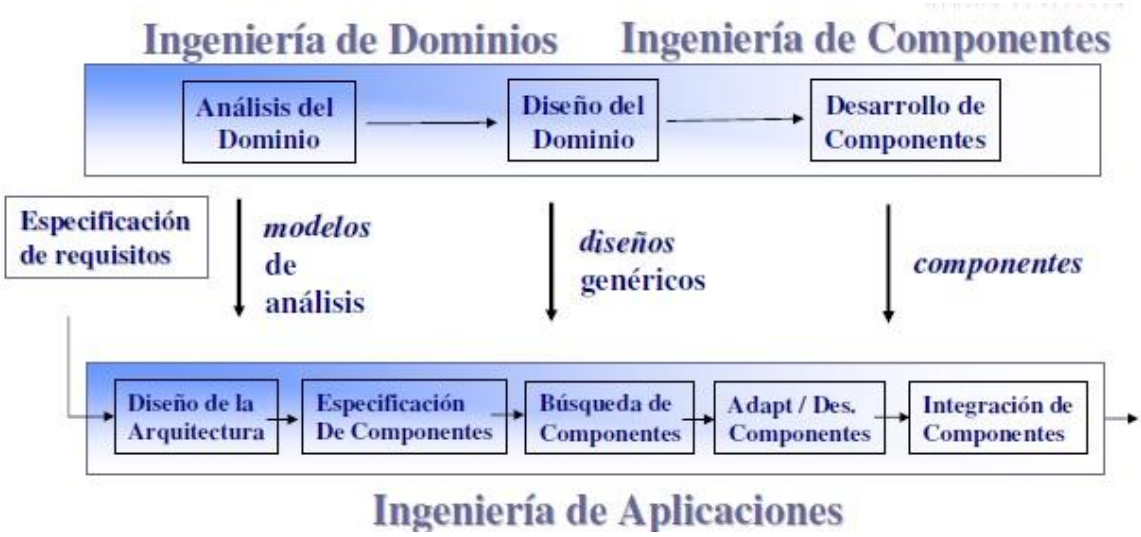

Figura 2.Modelo de procesos TWIN (VI Congreso Expotecnología, 2000)

WATCH: Modelo propuesto en la Universidad de Los Andes en Venezuela para el desarrollo de aplicaciones empresariales. Incluye IEEE 1074 en la creación de su estructura, emplea UML como lenguaje de modelado. Entre sus principales características se encuentran simple, completo y adaptable uso. (Montilva, 2006) Consta de dos componentes metodológicos:

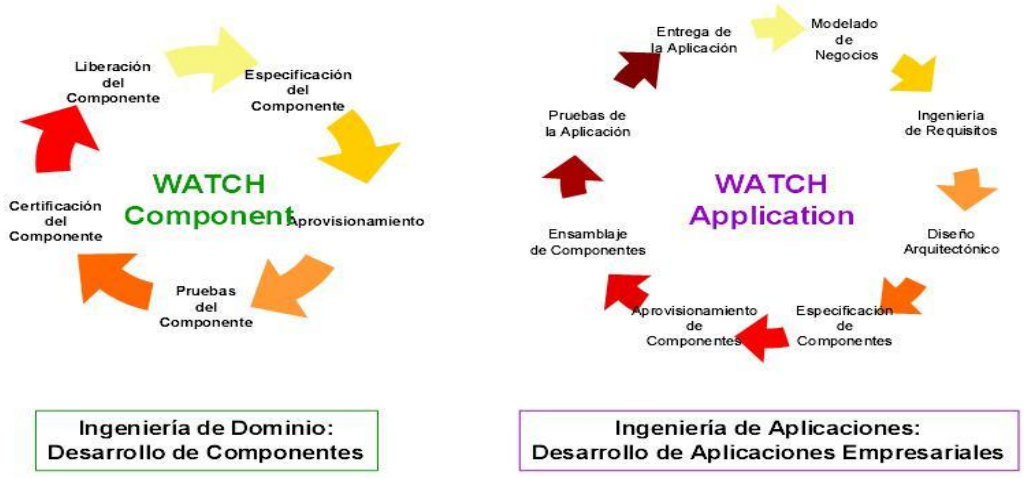

Figura 3.Modelo de procesos WATCH (Montilva, 2006)

A pesar de las características señaladas que resaltan el simple y adaptable uso de este modelo, no sería factible su utilización en los videojuegos porque centra su ingeniería de dominio en el desarrollo de componentes y no de activos de software reutilizables. Además que este está diseñado propiamente para aplicaciones empresariales.

Modelo de procesos de LPS desarrollado en el Instituto de Ingeniería de Software

El SEI (Instituto de Ingeniería de Software) propone un framework para SPLE, en donde se involucran tres actividades principales ilustradas en la figura 4: desarrollo de activos base, desarrollo de productos y la gestión respectiva de la línea. En el desarrollo de activos base se deben definir e implementar el conjunto de activos que serán utilizados durante la generación de cada producto. En la fase de desarrollo de productos se derivan los respectivos productos haciendo uso de los activos base. Finalmente, debe existir una actividad relacionada con la gestión integral de la línea, en donde se administren los activos y se tomen decisiones con respecto a su evolución.

Figura 4.Modelo de procesos SEI (Montilva, 2006)

Teniendo en cuenta que este modelo se centra en el desarrollo de activos de software que puedan ser reutilizados por futuros productos y además no impone limitaciones en las variadas formas en que el equipo de desarrollo desee adquirirlos se pudiera utilizar este

modelo en el desarrollo de los videojuegos del centro. Utilizando siempre las técnicas existentes para la realización de cada proceso.

Para adaptar un proceso de desarrollo de software que está basado en líneas de productos a un modelo que regule sus procesos es imprescindible realizar previamente un análisis del dominio específico mediante los métodos existentes.

1.4 Metodologías de análisis de dominio

Para la realización del análisis del dominio de videojuegos se realizó un estudio de las principales metodologías que existen como guía a este proceso. Dentro de las cuales se profundizó en las siguientes:

- **FODA (Análisis de dominio orientado a funciones):** FODA es una metodología desarrollada por el SEI para la aplicación del análisis de dominio, definiendo las etapas del método y los resultados obtenidos en cada una de ellas. Divide este proceso en tres etapas fundamentales que los resultados de cada una de ellas responden a la solución del problema planteado. Define un modelado del dominio, un especificación de los requisitos presentes en los videojuegos y una base arquitectónica que soporte a futuros productos.(Rebeca, 2002)

- **DARE (Entorno de reutilización y análisis y dominio):** DARE es una herramienta CASE que proporciona un entorno de trabajo y una metodología que facilita las tareas propias del análisis de dominio: extracción de información sobre el ámbito de aplicación y adquisición y almacenamiento de dicha información. Las principales ventajas respecto a otros entornos de trabajo como FODA y ODM es su elevada modularidad que la hace compatible con dichos entornos y un mayor esfuerzo en la automatización de estas actividades.(Rebeca, 2002)

- **FORE (Familia de requisitos):** FORE establece una metodología para el análisis de requisitos dentro de un ámbito de aplicación o área con el objetivo de crear requisitos generales que puedan ser instanciados para la obtención de aplicaciones concretas dentro del dominio. Es decir, se trabaja con la idea de productos o aplicaciones genéricos que engloben los rasgos comunes e instanciables de un subtipo de sistemas dentro del ámbito. A pesar que responde

a gran parte de la problemática planteada no existe un cubrimiento total de los elementos a generar para los videojuegos. Provee un detallado análisis de los requisitos aspecto importante en la investigación la que incluye además otros aspectos.(Rebeca, 2002)

1.5 Ingeniería de Requisitos

Para realizar con calidad el tratamiento de los requisitos necesario para la selección de las funcionalidades que servirán como orientación en la producción de los videojuegos, se debe conocer los fundamentos básicos de la ingeniería de requisitos.

La Ingeniería de Requisitos cumple un papel primordial en el proceso de desarrollo de software, ya que se enfoca a un área fundamental: la definición de lo que se desea producir. Según Sommerville 1995, su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes; de esta manera, se persigue minimizar los problemas relacionados con la gestión de los requerimientos en el desarrollo de sistemas. Está dirigida a mejorar la capacidad de predecir cronogramas en los proyectos así como sus resultados; proporcionando un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios. Favorece a lograr un consenso entre clientes y desarrolladores; evitando rechazos finales de los productos.

1.6 Arquitectura de software

La arquitectura de software se centra tanto en los elementos estructurales significativos del sistema, como subsistemas, clases, componentes, nodos y las colaboraciones que tienen lugar entre estos elementos a través de las interfaces. Los casos de uso dirigen la arquitectura para hacer que el sistema proporcione la funcionalidad y uso deseado, alcanzando a la vez objetivos de rendimientos razonables. Una arquitectura debe ser completa, pero también debe ser suficientemente flexible como para incorporar nuevas funciones, y debe soportar la reutilización del software existente. (Ivar, 2000)

Para el correcto funcionamiento de una arquitectura es necesario el empleo de estilos y patrones arquitectónicos.

1.6.1 Estilos arquitectónicos

Se define un *estilo arquitectónico* como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes. (Buschmann, 1996)

1.6.2 Patrones Arquitectónicos

Buschmann define un *patrón* como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución. Partiendo de esta definición, propone los *patrones arquitectónicos* como descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma en que estos colaboran entre sí.

1.6.3 Patrones de Diseño

Un *patrón de diseño* provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. Son menores en escala que los patrones arquitectónicos, y tienden a ser independientes de los lenguajes y paradigmas de programación. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero sí sobre la de un subsistema, debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema.

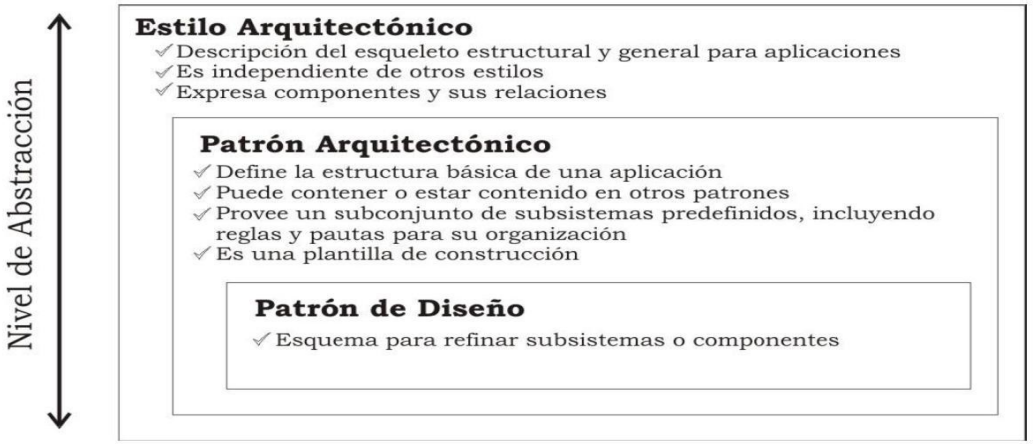

Figura 5. Relación entre estilos, patrones arquitectónicos y patrones de diseño.

Los estilos y patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad. Asimismo la organización del sistema de software debe estar disponible para todos los involucrados en el desarrollo del sistema, ya que establece un mecanismo de comunicación entre los mismos. Estas descripciones pueden establecerse a través de las vistas arquitectónicas, las notaciones como UML (Lenguaje Unificado de Modelado) y los lenguajes de descripción arquitectónica.

1.7 Metodologías de desarrollo de software

Siempre que se vaya a desarrollar un software es necesario usar una metodología que guíe el proceso. A continuación se muestran algunas características de las metodologías más utilizadas en la actualidad, para seleccionar la que se propondrá a ser utilizada en la producción de videojuegos en el centro.

Rational Unified Process (RUP): La metodología RUP es un proceso de desarrollo de software y junto con el UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Esta metodología se caracteriza por ser centrado en la arquitectura, ya que la organización del sistema depende de los casos de usos claves y debe tener en cuenta la comprensibilidad, la

facilidad de adaptación al cambio y la reutilización. Los casos de uso claves son aquellos que más le interesan al cliente.

Extreme Programming (XP): Es una de las metodologías de desarrollo de software más exitosas y utilizadas en la actualidad para el desarrollo de proyectos de corto plazo y pequeños equipos. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

1.8 Lenguaje Unificado de Modelado

El **Lenguaje Unificado de Modelado (UML)** se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema de notación destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Proporciona a los desarrolladores un vocabulario que incluye tres categorías: elementos, diagramas y relaciones. .

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

1.9 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Sirven de apoyo en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Rational Rose: Esta es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases.

Visual Paradigm Suite : Visual Paradigm en la versión 3.4, es una poderosa herramienta CASE de modelación visual. Utiliza UML como lenguaje para el modelado, además se puede decir que es un conjunto de herramientas de modelado que permiten realizar el modelado dentro del proceso de desarrollo de software.

Consideraciones Parciales

En este capítulo se han expuesto conceptos relacionados con los diferentes temas que serán desarrollados en el transcurso de la investigación. Los principales aspectos fundamentados estuvieron enfocados en la conceptualización de los modelos de desarrollo de software, analizando sus principales características y el funcionamiento de los mismos. Se realizó además un estudio detallado del modelo básico de desarrollo en una línea de productos de software localizando la esencia de cada una de sus partes. En este punto se centralizó la atención en la ingeniería de dominio definiendo sus activos fundamentales. Dentro de estos se tuvo en cuenta para el análisis de la línea base de arquitectura una descripción de los postulados esenciales de la arquitectura de software.

Capítulo 2. Propuesta de solución.

Introducción

En el transcurso del capítulo se expone la aplicación de elementos de la primera etapa del modelo de desarrollo de software propuesto, en la línea de videojuegos del centro. Con el propósito de brindar una guía para el proceso de desarrollo de los videojuegos en el centro se realizó un estudio de los activos de software reutilizables fundamentales que intervienen en la producción de estos. Teniendo en cuenta el estudio realizado se destacan los principales aspectos de la ingeniería de dominio que serán desarrollados y expuestos, así como la descripción de una línea base de arquitectura para estructurar el desarrollo de los videojuegos en el CEDIN.

2.1 Selección del modelo de procesos de desarrollo de software

Los modelos de procesos funcionan como procedimientos empleados por los equipos de desarrollo para organizar las tareas que se deben ejecutar en el ciclo de vida de un software. Analizando el proceso de desarrollo de los videojuegos en el centro, es evidente que este carece en su totalidad de este importante factor. Con el objetivo de proveer una guía a la producción de este tipo de software, se ha realizado un profundo estudio de diferentes modelos de procesos de desarrollo de software. Teniendo en cuenta las características de cada uno de los modelos analizados y las particularidades del proceso de realización de los videojuegos, se seleccionó como propuesta para guiar la producción de estos en el centro el modelo de procesos del SEI.

La selección de este modelo está basada fundamentalmente en la idea de ajustar la forma de desarrollar los videojuegos en el centro al paradigma LPS, como nueva alternativa en la Ingeniería de Software. Este es uno de los modelos de procesos para líneas de productos de software, que favorece en alta medida la reutilización de activos de software y no la integración de componentes reutilizables. En el desarrollo de los videojuegos en el centro, sería más factible la adopción de un modelo que garantice la identificación de los activos base de la línea de productos, para luego estos ser reutilizados. Con un modelo que presente tal característica se posibilita la reducción de costos de desarrollo en la producción, mejoras en los tiempos de entrega de los productos y aumento de la calidad, lo que conduce a un considerable incremento de la satisfacción de los clientes y usuarios que operarán con el software. Con lo que obtiene mayores ganancias la empresa de software.

A continuación se explicará detalladamente el funcionamiento que propone el modelo para la ingeniería de dominio.

Figura 6. Primera etapa del modelo propuesto. Ingeniería de dominio. (Montilva, 2006)

La figura muestra los principales elementos que recibe como entrada el proceso de desarrollo de activos base en la ingeniería de dominio para satisfacer las funcionalidades de futuras aplicaciones.

2.2 Lenguaje y herramienta de modelado a utilizar

Como se mencionaba en el capítulo anterior el lenguaje de modelado a utilizar será UML y como herramienta de modelado Visual Paradigm.

¿Por qué Visual Paradigm?

Porque Visual Paradigm es una poderosa herramienta CASE que al igual que el Rational Rose utiliza UML para el modelado, es la herramienta por excelencia para ser utilizada en un ambiente de software libre. Permite crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. Posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas, como el de componentes, despliegue, secuencia, casos de uso, clase, actividad, estado, entre otros. A continuación se exponen otras características que este ofrece:

- Entorno de creación de diagramas para UML.

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs (Integrated Development Environment).
- Disponibilidad en múltiples plataformas.
- Ofrecen un mecanismo general para la organización de los modelos/subsistemas/capas agrupando elementos de modelado.
- La licencia es pagada por la universidad.

2.3 Metodología de desarrollo de software seleccionada

Para la selección de la metodología se analizaron varias de las existentes para el desarrollo de software. La siguiente tabla muestra los aspectos fundamentales que condujo a la selección de RUP como metodología a seguir.

Metodologías ágiles	Metodologías robustas
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.

Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con menos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y Trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1. Comparación entre las metodologías ágiles y las robustas

2.4 Identificación de los activos fundamentales de la etapa Ingeniería de Dominio

Enfocando el sentido de la investigación al objetivo planteado a continuación se procederá a aplicar del modelo propuesto, elementos de su primera fase; para obtener como activos fundamentales de la línea de productos de videojuegos del centro los siguientes:

- Modelo de dominio.
- Especificación de requisitos presentes en los videojuegos.
- Arquitectura de dominio.

Para el análisis de la Ingeniería de Dominio en la línea de videojuegos teniendo en cuenta los aspectos propuestos por el modelo SEI vistos en la figura 6, fue necesario apoyarse en las metodologías existentes para análisis de dominio de las cuales se seleccionó la metodología FODA, que define las etapas del método y los resultados de cada una de ellas, centrado en el análisis del dominio orientado a sus funciones. La siguiente figura muestra cómo se desarrolla el análisis del dominio y sus resultados.

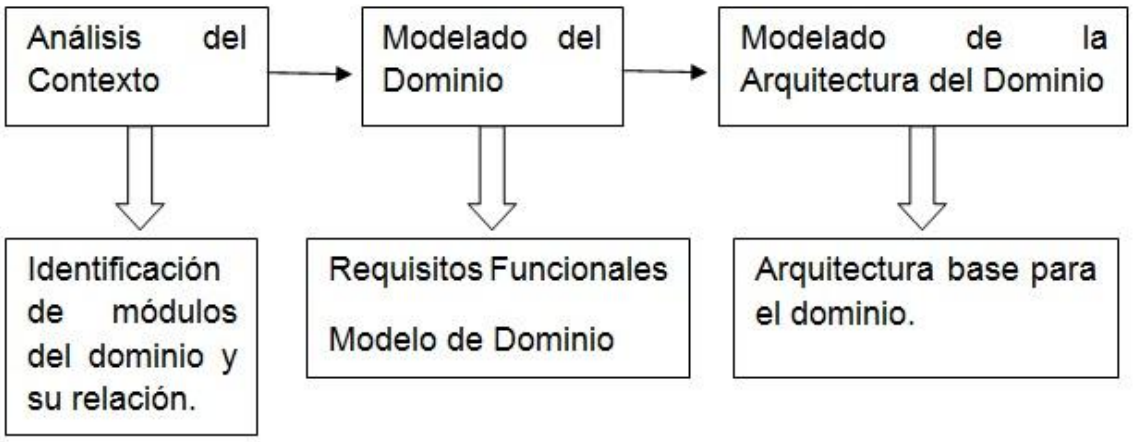

Figura 7.Método para el análisis de dominio.

2.4.1 Análisis del contexto

Para la identificación de los módulos se analizaron varios grupos de desarrollo de videojuegos además de consultar múltiples bibliografías de este tipo de software para enriquecer la propuesta. A continuación se ofrece una breve descripción de los diferentes equipos de desarrollo mencionados anteriormente:

Juegos CNeuro: La falta de recursos pedagógicos y de técnicas para el tratamiento de la Discalculia fue el móvil principal para la creación de este proyecto, el cual tuvo como objetivo: crear un paquete de Video-Juego para el tratamiento de la Discalculia en niños que cursan desde el 1ro hasta el 6to grado.

Desarrollo de juegos en K Dimensiones (DJKD): El proyecto Desarrollo de Juegos en K Dimensiones estuvo centrado en el desarrollo de juegos infantiles para el apoyo de la campaña "Misión Energética", que se llevó a cabo en Venezuela, brindando información sobre la conservación del Medio Ambiente, el Cambio Climático, el uso de Energías Alternativas y la Revolución Energética. Temas que por su importancia cognitiva y social son de interés del Estado que el pueblo venezolano conozca, se instruya y alcance una cultura basta el tema.

Grupo para el desarrollo de videojuegos (GDEVI): Debido a la carencia de Video-Juegos relacionados con los dibujos animados cubanos, surgió el Proyecto GDEVI, con el objetivo de comenzar a desarrollar juegos que tuvieran una temática más cercana a la vida cotidiana de los niños y jóvenes de nuestro país. GDEVI además fue un grupo que pretendió extender sus producciones a todas las ramas del ámbito de los Video-Juegos, y desde esta esfera, contribuir al desarrollo de la Informática en Cuba.

Proyectos juegos online (en línea): El Juego País de Nimo fue el primer producto producido por este equipo de desarrollo. El centro de atención está en la comercialización de Juegos en Línea.

Entrenadores Aduaneros: Tuvo la tarea de crear una aplicación con el objetivo fundamental de buscar una manera de sustituir el entrenamiento de los inspectores aduaneros para minimizar los gastos que se realizan en este proceso y minimizar los peligros en la seguridad nacional.

Videojuego de conducción de automóviles: Grupo de trabajo centrado en un género especifico de videojuegos. Uno de sus resultados fue el juego “Rápido y Curioso”. Con el análisis de este proyecto se logró identificar el módulo para la Gestión de Perfiles.

Con el análisis de las características comunes entre todos los videojuegos desarrollados por algunos de estos grupos y un intenso análisis de la documentación existente se logró definir los siguientes módulos que pueden evidenciarse en los videojuegos:

Módulos Principales

Módulo Lógico: Es el encargado de la lógica del juego. Posee todas las actividades que desarrollan la lógica de la aplicación representado por el motor lógico del juego.

Módulo gráfico: Este módulo es muy importante, pues la retroalimentación visual le llega al usuario gracias a los gráficos. Mientras más atractivos resulten, mayor sensación de inmersión se conseguirá, permitiéndole al jugador adentrarse cada vez más en el juego.

Módulo Físico – Matemático: Este módulo constituye un pilar fundamental en la confección de un videojuego, es el encargado de manejar toda la matemática y la física, como por ejemplo la detección de colisiones y las reacciones de los objetos del videojuego ante la acción de fuerzas.

Módulo de Inteligencia Artificial: La inteligencia artificial es un módulo realmente importante a la hora de diseñar cualquier videojuego, ya que es la encargada de dar capacidad de decisión y acción a los jugadores no controlados por el usuario. Mientras más “inteligentes” sean los oponentes artificiales, más difícil y entretenido será el juego, ya que los jugadores tendrán que emplearse a fondo para poder avanzar. Estos elementos se garantizan en el módulo de inteligencia artificial.

Módulo de Sonido: El sonido es otra importante forma de interacción en un videojuego. Muchas de las grandes empresas desarrolladoras de videojuegos invierten considerables sumas de dinero en la banda sonora y los efectos de sonido, contratando incluso a reconocidos compositores y orquestas para que interpreten sus canciones. Esto no lo

hacen por gusto, sino porque se ha comprobado que aumenta el suspenso, la emoción y por consiguiente la aceptación del videojuego por parte del público.

Módulo de Gestión de Datos: Brinda la posibilidad a los jugadores de personalizar el juego a su antojo, y además poder guardar los avances logrados al adentrarse en la trama del mismo.

Módulo Gestión de Perfiles: Da la posibilidad de crear, eliminar y modificar los perfiles a los jugadores en el caso que estos lo deseen.

Módulo de Red: El módulo de red en un videojuego se encarga de establecer una conexión rápida y estable entre el cliente y el servidor, garantizando la satisfacción de los usuarios que usan el modo multijugador, mediante el que pueden enfrentarse a otros competidores de cualquier lugar, con el único requisito de que estén conectados mediante una red con un ancho de banda adecuado.

Relación entre los módulos identificados en los videojuegos

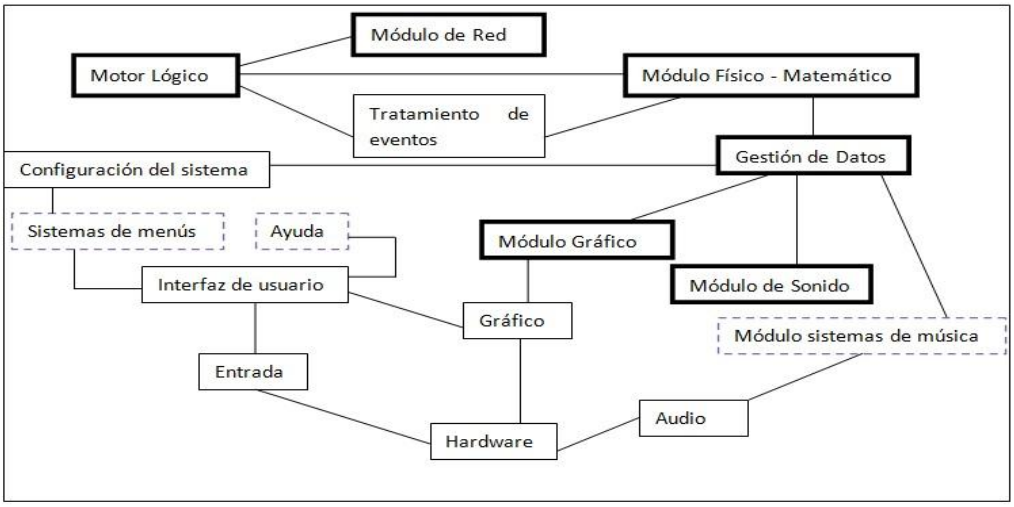

Figura 8.Relación entre módulos.

2.4.2 Modelado del Dominio

En esta etapa se obtiene como resultado el modelo de dominio y los requisitos funcionales comunes a todos los videojuegos.

Modelo de Dominio

Debido al bajo conocimiento de los procesos del negocio se plantea un modelo de dominio para lograr una mejor compresión de los conceptos del sistema. Para esto se realiza la descripción del modelo del dominio a través de un diagrama de clases UML, en el cual se definen los principales conceptos que intervienen de forma genérica en el desarrollo de los videojuegos.

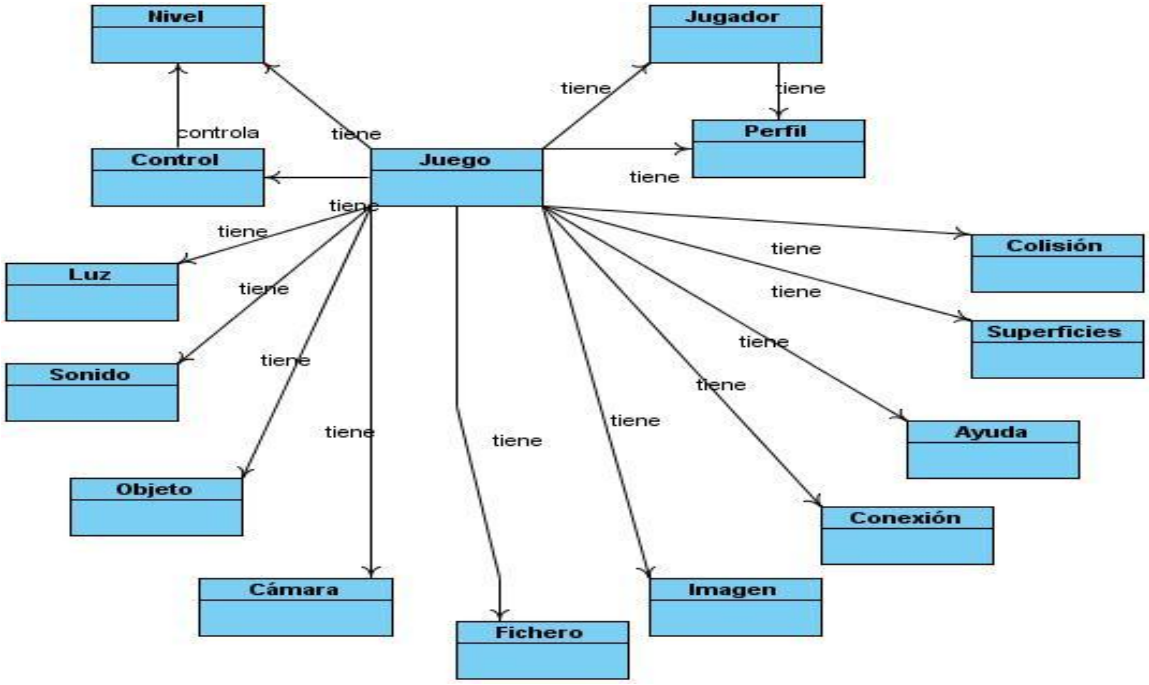

Figura 9.Modelo de Dominio.

Descripción de los conceptos del dominio.

Juego: Concepto principal de un videojuego que está formada por instancias de las demás elementos que intervienen en el desarrollo de estos. Tiene de 1 a varios jugadores en dependencia si el videojuego es multijugador o no.

Jugador: Interactúa con las funcionalidades que debe brindar el videojuego.

Perfil: Agrupa los datos del desempeño de un jugador en el juego.

Control: Coordina lógicamente todas las reglas de juego para avanzar en los niveles, realizando un previo tratamiento de eventos en cada instante del juego. Permite tener el control de tiempo.

Nivel: Interviene en la lógica del juego dando un sentido al avance del jugador en el mismo.

Luz: Permite la iluminación de las escenas del juego favoreciendo a la apariencia de la interfaz de usuario.

Cámara: Permite la visualización de objetos dentro de las escenas del juego.

Imagen: Componen las escenas del juego.

Superficies: Espacio destinado al desarrollo del juego.

Sonido: Permite la asignación de sonidos a diferentes eventos del juego.

Ayuda: Permite la orientación en el juego.

Colisión: Detecta elementos físicos dentro del juego por ejemplo la cinemática.

Conexión: Posibilita el juego en red.

Fichero: Guarda todos los datos persistentes del juego.

Objeto: Son componentes que pueden ser figuras geométricas o modelo tridimensionales.

2.4.3 Identificación de los requisitos funcionales

El tratamiento de requisitos realizado para la identificación de los requisitos funcionales que están presentes en los videojuegos fue necesario analizarlos según las 4 etapas definidas en la Ingeniería de Requisitos.

1. Obtención o captura de requisitos

Para capturar los requisitos funcionales comunes en los videojuegos se utilizaron técnicas de obtención de requisitos como la **entrevista** a personal especializado en los temas de realización de videojuegos e Ingeniería de Requisitos, la minería de requisitos en **sistemas existentes**, la **tormenta de ideas**, **Interfaz y Navegación, Mapas conceptuales** y la **Arqueología de documentos.**

Módulos	Requisitos Funcionales
Módulo Gestión de Perfiles	RF1: “Crear perfil de usuario” RF2: “Editar perfil de usuario” RF2.1: “Modificar el nombre del jugador” RF2.2: “Modificar cantidad de jugadores” RF3: “Cargar perfil de usuario” RF4: “Eliminar perfil de usuario”
Módulo Gráfico	RF5: “Configurar opciones gráficas” RF5.1: ”Modificar brillo” RF5.2: “Modificar contraste” RF5.3: “Modificar color” RF6: “Configurar Pantalla” RF6.1: ”Visualizar pantalla completa” RF6.2: “Visualizar pantalla media”

Módulo Sonido	RF7: “Configurar audio” RF7.1: “Modificar volumen de sonido” RF7.2: “Habilitar \Deshabilitar sonido”
Módulo Gestión de Datos	RF8: “Gestionar Partida” RF8.1: “Iniciar nueva partida” RF8.2: “Eliminar partida guardada” RF8.3: “Cargar partida guardada ” RF9: “Abortar partida” RF10: “Guardar partida” RF12 : “Pausar partida” RF13 : “Reanudar Partida”
Módulo Red	RF11: “Especificar IP del servidor” RF12: “Modificar tiempo de espera del servidor” RF13: “Modificar puerto de conexión”
Módulo Lógico	RF14: “Mostrar resultados del nivel” RF15: “Mostrar instrucciones del nivel” RF16: “Mostrar tiempo”
Módulo Físico	RF17: “Configurar controles” RF17.1:”Modificar teclas de movimiento“

Tabla 2. Requisitos por módulos

2. Análisis de requisitos

En el desarrollo de la etapa de análisis de requisitos se realizó un profundo estudio de los mismos basándonos en que es aquí donde se pretende abstraer y estructurar los aspectos que caracterizan a los videojuegos. Con este fin se efectuaron reuniones con personas que han estado involucradas en la producción de videojuegos y con otras que han desarrollado el rol de analista en alguno de los realizados con anterioridad en el CEDIN. Para de esta manera garantizar que los requisitos propuestos sean los más adecuados luego de atravesar esta etapa.

En esta etapa son utilizadas diferentes técnicas. La utilizada en este proceso fue la **orientada a puntos de vista** debido a que para la realización del análisis de los diferentes requisitos fue necesario el apoyo de variados puntos de vista de todos los implicados y así tener una visión general y hacer un estudio específico de los mismos.

3. Especificación de requisitos

La especificación, independientemente del modo como se realice puede verse como un proceso de representación. Los requisitos se representan de manera que lleven al éxito de la implementación del software. Para especificar los requisitos funcionales comunes a todos los videojuegos fue necesario apoyarse en las técnicas existentes de las cuales se hizo mayor uso de las que se mencionan a continuación:

Plantillas o patrones: Se utilizó con el propósito de describir los requisitos obtenidos mediante el lenguaje natural pero de una forma estructurada y predefinida por el equipo de desarrollo de los videojuegos, usando para ello el lenguaje del usuario. A través de ésta se eliminaron las ambigüedades que existían del lenguaje natural estructurando la información.

Glosario de términos: En la etapa de extracción de requisitos se realiza una amplia comparación de terminologías, esto hace necesario el uso de un marco de términos que sea entendido por la diversidad de personas que forman parte de los proyectos que desarrollan los videojuegos. Por tanto durante la etapa de especificación de requisitos se

identificaron algunas palabras técnicas que se hizo necesario la aclaración de su significado.

A continuación se realiza una breve especificación de los requisitos funcionales identificados.

RF1:"Iniciar juego": El usuario accede al juego mediante un ejecutable.

RF2: "Crear perfil de usuario": El jugador puede crearse un perfil, cuenta con un área donde se escribe el nombre que recibirá el perfil, después de escribir el nombre el jugador debe presionar el botón Aceptar para completar la acción de creación del perfil.

RF3: "Editar perfil de usuario":

RF3.1: "Modificar el nombre del jugador": Brinda la posibilidad al jugador de cambiar el nombre que identifica su perfil de los demás que están guardados en el juego.

RF3.2: "Modificar cantidad de jugadores": Permite seleccionar la cantidad de jugadores que intervendrán en una partida determinada.

RF4: "Cargar perfil de usuario": Muestra al jugador una lista de los perfiles creados en el juego, brindándole la posibilidad de seleccionar el perfil que desea activar. Después debe presionar el botón Aceptar para completar operación.

RF5: "Eliminar perfil de usuario": Muestra al jugador una lista de los perfiles creados en el juego, brindándole la posibilidad de seleccionar el que desea eliminar. Después de seleccionar el perfil debe presionar el botón eliminar para completar la operación.

RF6: "Configurar opciones gráficas"

RF6.1:"Modificar brillo": Brinda la posibilidad de cambiar la intensidad de brillo de la visualización del juego.

RF6.2: "Modificar contraste": Permite al jugador variar las diferencias de color en la visualización del juego.

RF7: “Configurar Pantalla”

RF7.1:”Visualizar pantalla completa”: Permite al jugador la visualización del juego en pantalla completa.

RF7.2: “Visualizar pantalla media”: Permite al jugador la visualización del juego sin ocupar toda la pantalla.

RF8: “Configurar sonido”

RF8.1: “Modificar volumen de sonido”: Permite al jugador aumentar o disminuir el volumen del sonido.

RF8.2: “Habilitar \Deshabilitar sonido”: Brinda la posibilidad al jugador de quitar o activar los sonidos del juego.

RF9: “Gestionar Partida”

RF9.1: “Iniciar nueva partida”: Permite al jugador comenzar un juego nuevo partiendo desde cero.

RF9.2: “Eliminar partida guardada”: Muestra al jugador la opción de eliminar la partida guardada y comenzar un nuevo juego.

RF9.3: “Cargar partida guardada”: Permite al jugador continuar una partida iniciada con anterioridad.

RF10: “Abortar partida”: Brinda la posibilidad al jugador de salir del juego cuando lo desee.

RF11: “Guardar partida”: Muestra al jugador la posibilidad de guardar una partida al salir del juego.

RF12:”Ir al menú principal”: El usuario debe tener la posibilidad de ir al menú de juego cuando lo desee.

RF13: “Pausar partida”: Muestra al jugador la opción de parar el juego cuando lo desee.

RF14: “Reanudar Partida”: Permite al jugador continuar un juego pausado con anterioridad.

RF15: “Configurar las opciones de red”: Muestra la posibilidad de variar la configuración establecida para establecer la conexión.

RF16:”Configurar música de fondo”

RF16.1:”Modificar volumen de la música del juego”: Permite al jugador variar el volumen de la música de fondo del juego.

RF16.2:“Habilitar\ Deshabilitar música del juego”: Permite al jugador quitar o activar la música de fondo del juego.

RF17: “Mostrar resultados del nivel”: Muestra al jugador los resultados obtenidos en el nivel alcanzado.

RF18: “Mostrar instrucciones del nivel”: Muestra al jugador lo que se debe hacer en el nivel si este lo desea.

RF19: “Mostrar tiempo”: Muestra al jugador el tiempo que demoró en un nivel.

RF20: “Configurar controles”

RF20.1:”Modificar controles de movimiento“: Permite cambiar las teclas que permiten el movimiento en el juego.

Glosario de Términos

Perfil: Información personalizada del jugador.

Nivel: Ascenso en la complejidad del juego.

4. Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que

el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias.

Para la validación de los requisitos capturados se analizaron varias técnicas como el empleo de **prototipos**, **casos de prueba**, **matrices de trazabilidad**, **auditorías** y **revisiones de requisitos**. De las cuales se utilizó las **revisiones de requisitos** y el uso de **prototipos** teniendo en cuenta que estos son una propuesta genérica de requisitos para todos los videojuegos y no para un sistema en particular.

Las **revisiones de requisitos** consisten en una o varias reuniones planificadas, donde se intenta confirmar que los requisitos poseen los atributos de calidad deseados. Estas reuniones son llevadas a cabo por el analista encargado del proyecto y un conjunto de colegas que, preferiblemente, no están relacionados con el proyecto y, además, son competentes en la actividad de requisitos. El resultado final de las reuniones de revisión es un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas.

En consecuencia, los pre-revisión son un buen medio de:

• **Permitir que los participantes se concentren en los defectos difíciles de identificar.** Esto es una consecuencia directa de eliminar los defectos más sencillos y evidentes.

• **Disminuir el número de errores y acciones a realizar** ya que, previsiblemente, el número de defectos identificados será mucho menor.

El uso de **prototipos** no es más que para cada funcionalidad detectada se modela una propuesta de como quedaría a grades rasgos dicha función en el software lo que disminuye en gran medida las no conformidades de los clientes.

2.4.4 Modelado de una propuesta arquitectónica para los videojuegos

A continuación se explica la línea base de la arquitectura para un mejor entendimiento del sistema, las herramientas y tecnologías propuestas seleccionadas para realizar los videojuegos y los subsistemas por los que está formado el proceso de desarrollo de un software de este tipo. Esta propuesta estructurará la forma de producción de los videojuegos en el centro.

Línea base de la arquitectura

La Línea Base de la Arquitectura contiene elementos de gran importancia para lograr la máxima abstracción en el diseño arquitectónico de la aplicación. En la misma se exponen los estilos arquitectónicos de la aplicación, así como los principales elementos de la arquitectura, se describen los principales patrones de arquitectura utilizados, las metodologías, herramientas y tecnologías de software que se utilizarán en los videojuegos a desarrollar en el centro.

Propósito

El propósito de la línea base es proporcionar la información para estructurar el sistema desde el más alto nivel de abstracción, se define la estructura del sistema en cuanto a los elementos, la configuración y sus restricciones.

Alcance

Describe detalladamente al organigrama de la arquitectura encarnada en los estilos arquitectónicos que se utilizarán, se propone la utilización de un conjunto de patrones que van a resolver futuros problemas que se van a presentar a lo largo del desarrollo de los software. Se proponen las principales tecnologías y herramientas que soportan los estilos y patrones especificados y cumplen con las restricciones del sistema.

Rol del arquitecto

Indudablemente el rol del arquitecto de software es crítico y sumamente importante, puesto que requiere de una gran variedad de conocimientos, tales como: ingeniería de requisitos, teoría de arquitecturas de software, codificación, tecnologías de desarrollo, plataformas de software y hardware etc. Por otra parte, el arquitecto de Software debe

dominar la mayor cantidad de tecnologías de software y prácticas de diseño, para así poder tomar decisiones adecuadas para garantizar el mejor desempeño de las aplicaciones.

Además, requiere de saber negociar intereses encontrados de múltiples involucrados en el desarrollo de un sistema de software, promover la colaboración entre el equipo, entender la relación entre atributos de calidad y estructuras, ser capaz de transmitir claramente la arquitectura a los equipos, escuchar, y entender múltiples puntos de vista. El arquitecto de software además debe interaccionar con todos los involucrados en el desarrollo de un sistema de software, y ser capaz de dialogar con el analista para obtener los requerimientos significativos, diseñarlos y transmitirlos al programador para su correcta codificación.

Estilo arquitectónico propuesto

Como se describió en el capítulo anterior la arquitectura de software es la estructura más importante de un sistema, puesto que es la encargada de la estructura y relaciones de los componentes de dicho sistema. Partiendo de que los estilos arquitectónicos representan el nivel de abstracción mayor para estructurar el sistema, la elección del mismo está dada por el tipo de aplicación que se vaya a desarrollar. Teniendo en cuenta los elementos mencionados anteriormente, la arquitectura que se propone está basada en el estilo arquitectónico llamada y retorno utilizando el patrón **Arquitectura en capas.** Este es un patrón clásico de aplicaciones web pero analizando el funcionamiento de los videojuegos su desarrollo se ajusta a las características de este patrón. En la siguiente figura se muestra la materialización de una arquitectura en 3 capas en el desarrollo de los videojuegos.

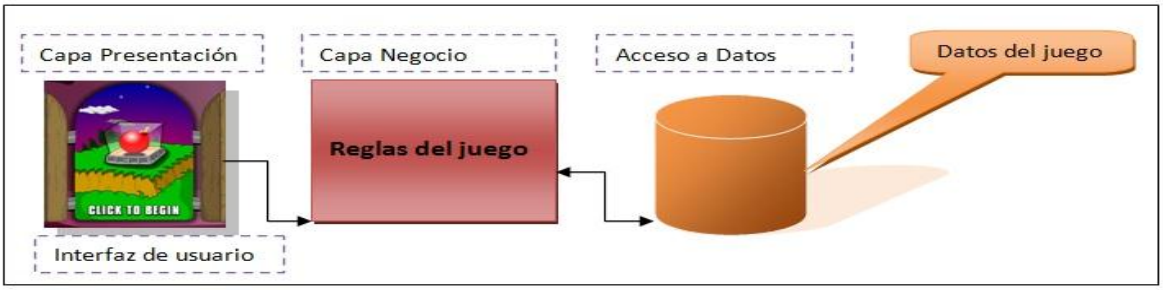

Figura 10.Arquitectura en 3 capas en los videojuegos.

En la capa de presentación es donde están contenidas todas las clases de la interfaz de usuario. Estas se comunican con la capa de negocio tras cada evento realizado por los jugadores en el juego. La capa de negocio siguiendo las reglas del juego es la encargada de obtener o guardar los datos en dependencia de la acción ejecutada.

Patrones GoF

El concepto de patrón se viene utilizando en varias disciplinas, pero no es utilizado en la ingeniería del software hasta la publicación del libro “Desing Patterns”, escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides (también conocidos por “Gang of Four”, o GoF). En este libro se definen veintitrés patrones recurrentes y proponen soluciones sencillas.

De estos veintitrés, se explicarán los más destacados y que se pueden utilizar a la hora de diseñar e implementar código de un videojuego. Los patrones seleccionados son:

- **Singleton**

Este patrón garantiza que una clase solo pueda tener una instancia dentro de la aplicación y establece su punto de acceso global. Un ejemplo de su uso puede ser el controlador de algún elemento de Hardware que se necesite nuestra aplicación, por ejemplo, el ratón, el teclado, el sonido entre otros. Otro uso es mantener el estado del videojuego (posiciones de los elementos móviles, estado de los avatares...). En la API del motor Ogre3D es muy común el uso de este patrón.

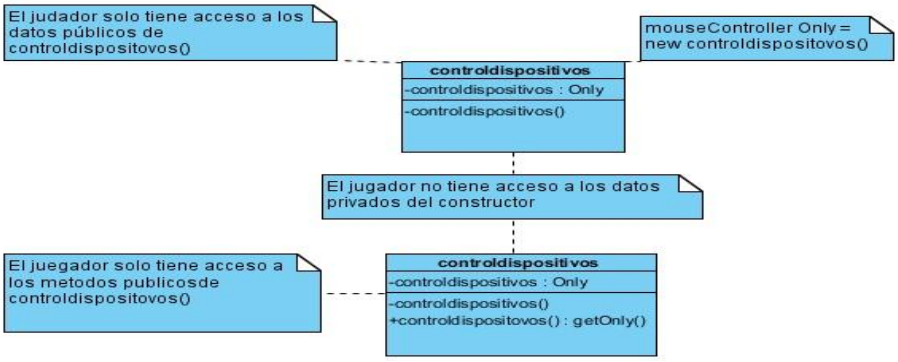

Figura 11. Patrón Singleton

- **Factory Method**

Este patrón define la forma en que deben de crearse determinados objetos a partir de un objeto llamado fábrica (o Factory). Por ejemplo, al leer una cierta imagen desde un fichero, podemos delegar la elección del algoritmo de descompresión (gif, png, jpeg...) a un objeto Factory y que éste nos devuelva una instancia a un objeto imagen.

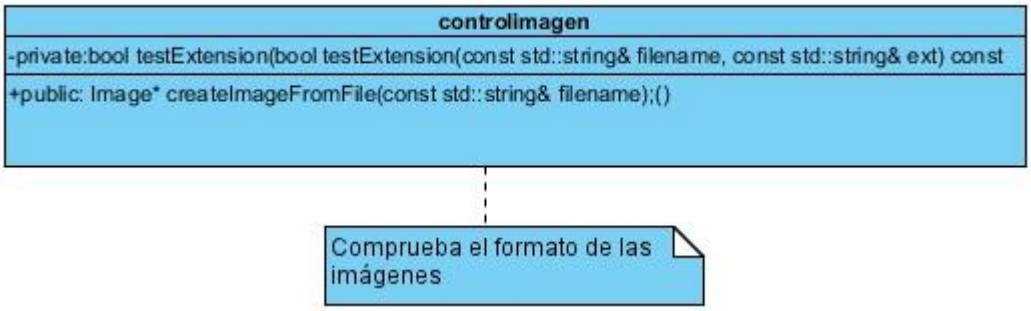

Figura 12. Patrón Factory Method

- **Observer**

El patrón Observador también conocido como "*spider"* el cual define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes, en el juego el observador sería una clase o un objeto que cuando el jugador cambie por ejemplo de nivel o realice una operación este actualizaría o informaría a todo el sistema de lo ocurrido.

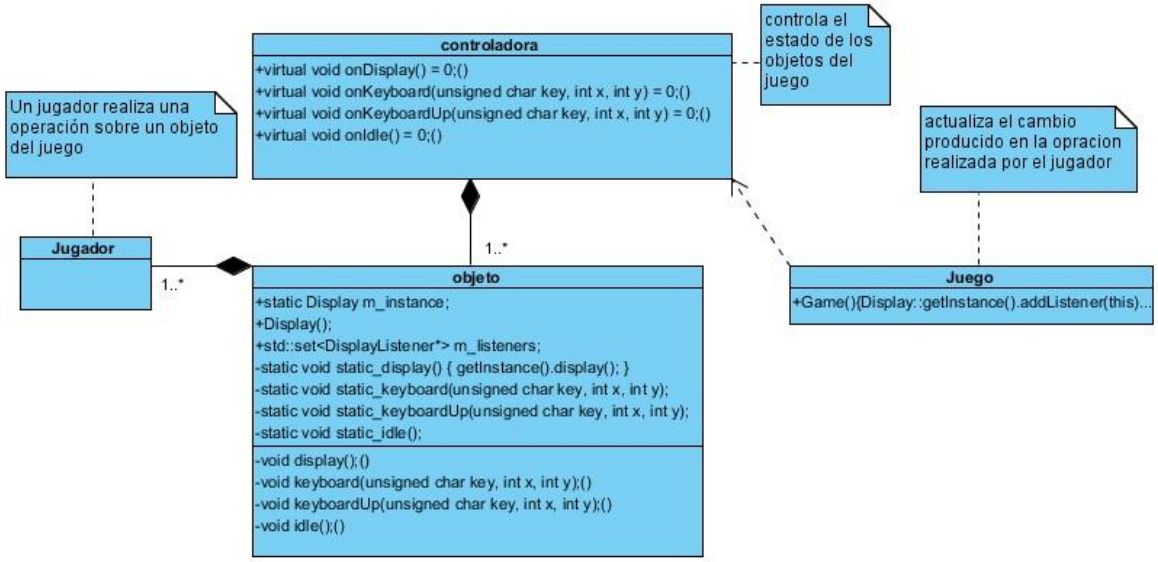

Figura 13. Patrón Observer

Patrones GRASP

Los patrones GRASP describen una serie de principios fundamentales para la asignación de responsabilidades a objetos, expresados en forma de patrones. Por lo que es muy necesario e importante entender y poder aplicar estos principios durante la preparación de un diagrama de interacción. A continuación se hace una breve descripción acerca de los patrones utilizados durante esta propuesta para la asignación de responsabilidades.

Experto: Este patrón consiste en asignarle una responsabilidad al experto en información, en el juego cuando el usuario interactúe con la aplicación, el experto en la información seria la clase controladora, que se encargaría de gestionar el evento correspondiente a la acción del jugador.

Alta Cohesión: Cuando se dice que una clase tiene alta cohesión, se quiere decir que el objeto tiene bien delimitadas sus responsabilidades. En la programación orientada a objetos, cada objeto tiene una responsabilidad que ha de cumplir dentro de un programa, en este caso se pretende que todas las responsabilidades de las clases del juego estén bien definidas. En una aplicación de juego este patrón estará presente al aplicar el patrón arquitectura en capas pues cada una trabajara de forma independiente sucediendo que al modificar una de ellas no se afectarán las demás.

Bajo acoplamiento: El bajo acoplamiento hace referencia a las relaciones que tienen los objetos entre si dentro de un sistema. Teóricamente, cuando una serie de objetos tienen

una relación con varios objetos, cuando estos últimos son cambiados, los objetos relacionados han de verse afectados necesariamente. El alto acoplamiento se da normalmente, cuando un objeto ha de saber demasiados detalles internos de otro para su funcionamiento, es decir, se rompe el encapsulamiento de otro objeto. Por ello cuando menos acoplamiento, mejor diseñado estará el sistema y es precisamente lo que perseguimos dentro de este tipo de aplicación, ya que cualquier juego encapsula una serie de objetos que tendrán que estar actualizándose a medida que transcurra el desarrollo de por ejemplo (una partida).

Metas y Limitaciones de la Arquitectura

A continuación se plantean las metas y restricciones con las que se deberá cumplir a cabalidad para la correcta puesta en funcionamiento de los videojuegos, y que además son significativas para la arquitectura. Además, se hace mención de los requerimientos no funcionales así como de los requerimientos funcionales arquitectónicamente más significativos para el desarrollo de este tipo de aplicaciones.

Requisitos Funcionales

RF1:”Iniciar juego”

RF2: “Crear perfil de usuario”

RF3: “Editar perfil de usuario”

RF3.1: “Modificar el nombre del jugador”

RF3.2: “Modificar cantidad de jugadores”

RF4: “Cargar perfil de usuario”

RF5: “Eliminar perfil de usuario”

RF6: “Configurar opciones gráficas”

RF6.1:”Modificar brillo”:

RF6.2: “Modificar contraste”:

RF7: “Configurar Pantalla”

RF7.1:”Visualizar pantalla completa”

RF7.2: “Visualizar pantalla media”

RF8: “Configurar sonido”

RF8.1: “Modificar volumen de sonido”

RF8.2: “Habilitar \Deshabilitar sonido”

RF9: “Gestionar Partida”

RF9.1: “Iniciar nueva partida”

RF9.2: “Eliminar partida guardada”

RF9.3: “Cargar partida guardada”

RF10: “Abortar partida”

RF11: “Guardar partida”

RF12:”Ir al menú principal”

RF13: “Pausar partida”

RF14: “Reanudar Partida”

RF15: “Configurar las opciones de red”

RF16:”Configurar música de fondo”

RF16.1:”Modificar volumen de la música del juego”

RF16.2:“Habilitar\ Deshabilitar música del juego”

RF17: “Mostrar resultados del nivel”

RF18: “Mostrar instrucciones del nivel”

RF19: “Mostrar tiempo”

RF20: “Configurar controles”

RF20.1:"Modificar controles de movimiento"

Requisitos no funcionales

1. Usabilidad.

RnF 1.1 **Tipo de usuario final:** Los usuarios del sistema serán los clientes del mismo y/o jugadores.

RnF 1.2 **Tipo de aplicación informática:** Los videojuegos podrán ser multijugador, para un solo jugador o en red.

RnF 1.3 **Finalidad:** Los objetivos que persiguen estos sistemas pueden ser de informar y/o entretener al usuario.

RnF 1.4

2. Confiabilidad.

RnF 2.1 **Disponibilidad del sistema:** El sistema se encontrará 100% disponible, se encontrará disponible las 24 horas del día, no tendrá un límite de horas de uso y el cliente podrá utilizarlo el tiempo que estime conveniente.

RnF 2.2 **Tiempo medio entre fallos y reparación:** El tiempo medio entre fallo del sistema será de 5 años para proveer cambios que ocurran en la universidad, la aplicación tendrá un tiempo medio de reparación de 1 año solamente.

RnF 2.3 **Exactitud:** Se podrá salir del sistema en el momento preciso que el usuario estime conveniente, así mismo el sistema efectuará cualquiera de sus otras funciones con la mayor exactitud posible de acuerdo a las peticiones del mismo.

3. Eficiencia.

RnF 3.1 **Tiempo de respuesta por transición:** El tiempo de respuesta ante peticiones debe dar la sensación de inmediatez, con un tiempo máximo de respuesta por transacción de 5 seg.

RNF 3.2 **Rendimiento:** Se deben representar las imágenes o cuadros (frame) a una velocidad mínima de 30 frame por segundo.

RnF 3.2 **Capacidad:** Deberá soportar un alto número de clientes dentro del paseo virtual, máximo por definir aún

RnF 3.3 Utilización de recursos: El sistema debe hacer uso de la menor cantidad de recursos posibles, dígase discos externos, memorias flash, etc.

4. Soporte.

RnF 4.1 Plataformas: El sistema deberá ser multiplataforma, inicialmente para los sistemas operativos Linux y Windows.

5. Restricciones de diseño:

RnF 5.1 Lenguajes de programación: Se utilizara el lenguaje de programación C++.

RnF 5.2 Herramientas de diseño gráfico: Se utilizará la herramienta Blender ,Maya y 3DStudio para la modelación de gráficos.

RnF 5.3 Hardware: Las características de hardware que debe tener la PC donde se vaya instalar o utilizar la aplicación debe cumplir con los siguientes requisitos mínimos:

- Memoria RAM de 1 Gb o superior.
- Computadora P4 o superior.
- Velocidad de Microprocesador a 2.0 o superior.
- Recursos de Video a 128 Mb o superior.

6. Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RnF 6.1 Manual de usuario disponible: El sistema deberá contar con un manual de usuario con las instrucciones para el uso del mismo.

7. Interfaz

RnF 7.1 Interfaz de Usuario: La interfaz del videojuego será sencilla y agradable para facilitar a los usuarios el uso de la misma, mostrará botones e información imprescindible en una localización adecuada según las reglas del juego que se vaya a desarrollar. Se hará uso además de textos con tamaño razonable, fácilmente legibles e identificables, usando la mayor coherencia posible entre las fuentes y las interfaces utilizadas.

2.5 Representación de la arquitectura

La arquitectura no requiere un estilo único. A veces esta tiene secuelas de un diseño donde se particionó prematuramente el software o se hizo un énfasis excesivo en la ingeniería de los datos, la eficiencia en tiempo de ejecución, o estrategias de desarrollo y organización de equipos. A menudo, la arquitectura tampoco aborda los intereses de todos sus "clientes". El modelo de 4+1 vista fue desarrollado para remediar este problema.

Siguiendo la metodología RUP tal y como se expuso en el Capítulo 1, a continuación se desarrolla la descripción de la arquitectura a través de las 4+1 vistas propuestas por Kruchten, quien describe la arquitectura del software usando cinco vistas concurrentes, donde cada vista se refiere a un conjunto de intereses de diferentes trabajadores del sistema.

2.5.1 Vistas arquitectónicas

Figura 14.Modelo 4+1 vista

1. **Vista de Casos de Uso**

Esta vista representa un subconjunto del artefacto Modelo de casos de uso y lista los casos de usos o escenarios más significativos, con las funcionalidades centrales del sistema.

Los Casos de Uso arquitectónicamente significativos, son aquellos que describen funcionalidades imprescindibles para el sistema, y que a través de estos se valida la arquitectura propuesta para el mismo. En esta vista se representa el diagrama de casos de uso con los casos de uso arquitectónicamente significativos para el sistema

y una descripción de cada uno. Teniendo en cuenta que la propuesta está dirigida a un grupo genérico de software se representarán aquellos casos de uso que delimiten las principales funcionalidades que brindan los videojuegos a un jugador.

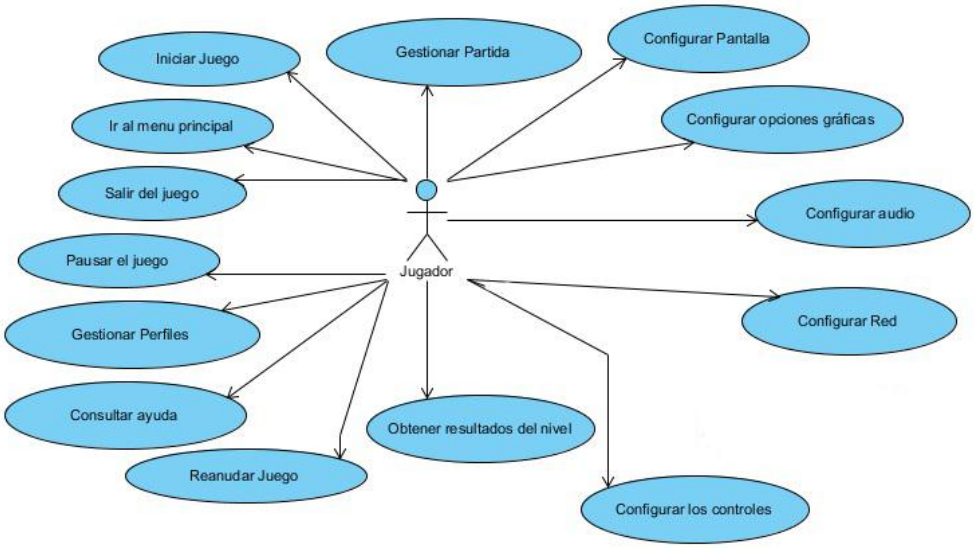

Figura15. Diagrama de casos de uso.

Descripción de los Casos de Uso (Anexo # 2)

2. Vista Lógica

Esta vista representa un subconjunto del artefacto Modelo de Diseño, representando los elementos de diseño más importantes para la arquitectura del sistema, aquí se describen las clases más importantes, su organización en paquetes y subsistemas. Esta descripción se realiza a través de diagramas de clases para ilustrar la relación entre las clases arquitectónicamente significativas, subsistemas y paquetes.

Teniendo en cuenta, que el objetivo del trabajo no persigue describir los elementos del diseño de una aplicación en específico, sino que está centrado en dar una vista lo más abstracta posible de cómo quedaría distribuida lógicamente una aplicación de juego de tal manera que sea posible utilizarla en cualquiera de los diseños orientados a este tipo de aplicaciones, por supuesto siempre dejando claro donde se encontrará cada una de las clases involucradas en las mismas.

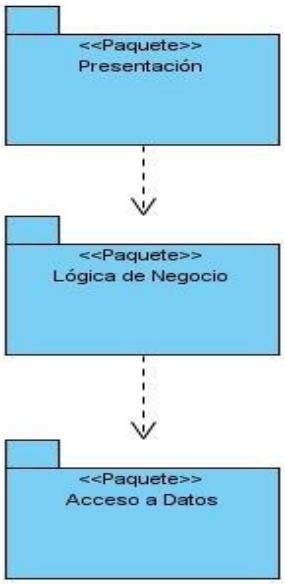

Figura 16.Estructura en capas

Descripción de las Capas

Presentación: Aquí se encontrarán el conjunto de clases que componen la interfaz del sistema.

Lógica de Negocio: Aquí estarán las clases encargadas de gestionar todos los procesos de la lógica de negocio.

Acceso a Datos: Aquí estará ubicado el conjunto de clases que controlan el tratamiento de los datos.

3. Vista de Procesos

Esta vista suministra una base para la comprensión de la organización de los procesos de un sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos concurrentes o hilos. En nuestro caso la solución propuesta no presenta procesos concurrentes por tanto no se hace representación de esta vista.

4. Vista de Implementación

En esta vista se describe la descomposición del software en capas y subsistemas de implementación. Además, la misma provee una vista de la trazabilidad de los elementos de diseño de la vista lógica ahora para la implementación.

Muy similar a la vista lógica ocurre a la hora de describir los componentes que conformarán cada una de las capas o subsistemas de implementación dentro de esta vista, ya que no perseguimos describir la descomposición de un software en específico, sino que tratamos de dejar bien claro donde se encontrará cada uno de los componentes generados en la conformación de un videojuego, haciéndolo lo más abstracto posible de tal forma que pueda ser reutilizado en este tipo de aplicaciones. En la figura que se muestra a continuación, la descomposición de la vista de implementación en subsistemas de implementación, notar que la relación entre los componentes no está reflejada en el mismo puesto, dicha relación depende del tipo de componente que se genere de acuerdo con el objetivo que se persiga en la aplicación. Cada componente responde a los módulos identificados para el proceso de realización de los videojuegos.

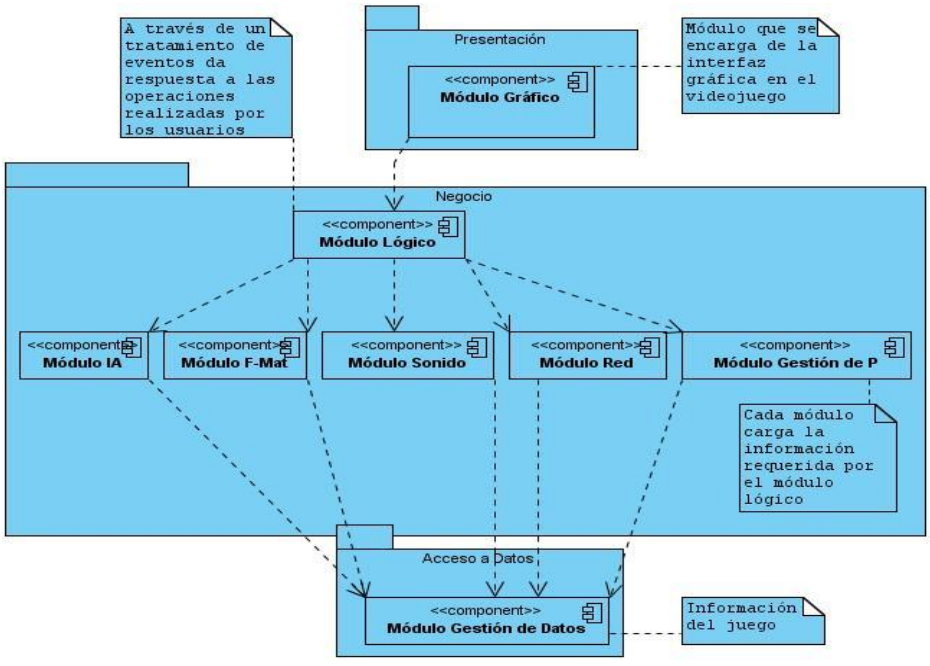

Figura 17. Vista de Implementación

5. Vista de Despliegue

Esta vista suministra una base para la comprensión de la distribución física de un sistema a través de nodos. Suele utilizarse cuando el sistema está distribuido. Y hay una traza directa del modelo de implementación, puesto que cada componente físico debe estar almacenado en un nodo.

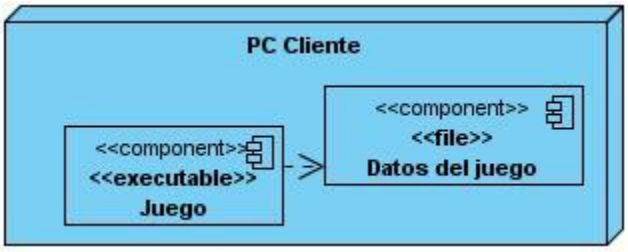

Figura 18.Vista de Despliegue

Nodo Cliente: Donde estaría instalado el juego

La figura representa lo más abstracto posible la distribución física que se evidencia en un videojuego centrando la atención en el elemento básico para que pueda ser reutilizable.

Consideraciones Parciales

En este capítulo se planteó la descripción de los activos fundamentales de la Ingeniería de Dominio, como primera fase del modelo de procesos de desarrollo de software propuesto. Se hizo una descripción de la línea base de la arquitectura mostrándose las metas y restricciones arquitectónicas, además de la representación arquitectónica a través del modelo 4+1 vista, y de forma general dando una explicación concreta de cómo formular completamente la arquitectura propuesta para el desarrollo de los videojuegos.

Capítulo 3. Validación de la propuesta.

Introducción

El presente capítulo muestra la validación de la investigación a partir del estudio de las técnicas que para este fin existen. Se aprecian los resultados obtenidos en el uso de la técnica seleccionada, se grafican estos para permitir una mejor visibilidad de los mismos. Para lo que es necesario un previo análisis de los criterios emitidos por los especialistas entrevistados según la técnica seleccionada teniendo en cuenta varios parámetros que definirán la aceptación de la propuesta realizada.

3.1 Tipos de evaluación

Las investigaciones científicas deben ser evaluadas de tal manera que se demuestre su validez práctica y uso posterior. Con este objetivo existen técnicas de validación las cuales fueron analizadas logrando la selección de una de ellas para validar el presente trabajo.

Seguidamente se mencionan varios tipos de evaluación especificando los resultados de la técnica utilizada.

3.1.1 Método de consulta a expertos. Método Delphi

Este método tuvo sus inicios en la década de los 50, y a partir de aquí fue utilizado frecuentemente como sistema para obtener información sobre las ocurrencias de un fenómeno en el futuro. Consiste en la selección de un grupo de expertos a los que se les encuesta su opinión sobre cuestiones referidas a sucesos del futuro Se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos, obtenido, encuestando a este grupo mediante un cuestionario. Es un método fiable y muy utilizado actualmente.

No existe posibilidad de aplicarlo a la propuesta realizada debido a que no se cuenta en la universidad con el número de personas con experiencia en el tema que exige este método.

3.1.2 Grupo focal

Consiste en la selección de grupo de personas con conocimientos sobre el tema, deben ser especialistas, expertos, de distintos niveles y categorías, que se reúnen en un lugar a una hora determinada, donde se discute en forma de grupo debate sobre el procedimiento, siendo este debate dirigido por los autores, y centrado en lo que se quiere conocer sobre el procedimiento.

Este método necesita de mucho personal y los que podrían ser seleccionados en estos momentos están cumpliendo otras labores por lo que se hace imposible realizarlo.

3.1.3 Método de consulta a especialistas

El criterio de especialistas es un instrumento rápido y eficaz por el potencial que contiene para conformar, valorar y enriquecer criterios, concepciones, modelos, estrategias o metodologías. Existen varias técnicas: encuestas, cuestionarios, entrevistas, estados de opinión, Positivo-Negativo–Interesante y sugerencias.

Se escoge este método por las ventajas que posee y por las características que presenta el marco donde se realiza la investigación. Aplicando la técnica de entrevistas.

3.1.4 Triangulación

La triangulación entendida como técnica de confrontación y herramienta de comparación de diferentes tipos de análisis de datos (triangulación analítica) con un mismo objetivo puede contribuir a validar un estudio de encuesta y potenciar las conclusiones que de él se derivan. Se hace a través de datos estadísticos recogidos sobre resultados de la puesta en práctica o de los estados de opinión.

Este método tampoco se puede aplicar a la propuesta pues los datos deben ser recogidos aplicando la misma, o recogiendo estados de opinión de los involucrados, cosa que tampoco se hace posible por el momento, dado a lo explicado previamente.

3.1.5 Validación práctica

Consiste en la obtención, comparación y análisis de resultados obtenidos al aplicar prácticamente el proceso en varios proyectos.

No se escoge este método porque no existen proyectos que se le pueda aplicar el proceso y que se pueda ver resultados en un corto período de tiempo.

3.1.6 Recopilación de información

Se basa en la recopilación de estados de opinión. Encuestas, cuestionarios o entrevistas a los clientes o a las personas que tengan que ver de una forma u otra con el proceso, o con la puesta en práctica de este de forma general.

No se puede aplicar porque depende de la realización práctica de la propuesta.

Luego del análisis de los métodos de evaluación, se decidió utilizar el **método consulta a especialistas** por ser el que más se ajusta a la propuesta de acuerdo con las necesidades y oportunidades de la misma.

3.2 Actividades para la validación

La propuesta se validará mediante el método de criterios de especialistas. El tema de la investigación abarca dos tópicos importantes que necesitan de una verificación de especialistas, ya sean especialistas en Ingeniería de Requisitos y además los especialistas en el desarrollo de videojuegos. De los mismos se debe recoger la información siguiente:

- Nombre y Apellidos
- Grado científico
- Ocupación
- Vinculación a proyecto
- Rol que desempeña
- Breve currículo

Para seleccionar los especialistas que participarán en la validación se tuvo en cuenta los requisitos siguientes:

- Poseer grado científico de ingeniero o superior.
- Tener conocimiento y experiencia en el tema.
- Pertenecer a la universidad y conocer el proceso de desarrollo de los videojuegos, para valorar correctamente el ajuste de las necesidades del mismo que posee la propuesta.

Debido a los requisitos de selección anteriores, de un total de 30 especialistas reconocidos y consultados en el centro, se seleccionaron solamente 6, representando un 20%, se definió que para el tema de Ingeniería de Requisitos se entrevistaran a tres especialistas y para el tema de videojuegos se consultaran tres especialistas. Con el objetivo de conocer sus criterios sobre la propuesta presentada.

Una vez seleccionados los especialistas en el tema de la Ingeniería de Requisitos, se les realiza una **entrevista** que contiene las siguientes preguntas:

1. ¿Los requisitos funcionales propuestos presentan ambigüedad?

2. ¿Está correctamente realizado el análisis de requisitos a los propuestos en la investigación?

3. ¿Está correctamente elaborada la especificación de requisitos que se muestra en la propuesta?

4. ¿Existen inconsistencias en los requisitos propuestos?

5. ¿Están completos los requisitos funcionales propuestos?

6. ¿Los requisitos propuestos son concisos?

Teniendo los especialistas en el tema de desarrollo de videojuegos, los cuales constituyen los más adecuados a tener en cuenta, por ser personas que desarrollan videojuegos en el centro y poseer, además, conocimientos avanzados de ingeniería de software, se les realiza una entrevista con las siguientes preguntas:

1. ¿Los requisitos funcionales que se proponen pueden ser reutilizados por cualquier videojuego que se desarrolle en el centro?

2. ¿Qué importancia le atribuye a la reutilización de la especificación de requisitos propuesta?

3. ¿Cree que la línea base de la arquitectura propuesta se ajusta al desarrollo de los videojuegos en el centro?

4. ¿Qué opina sobre la genericidad de los requisitos planteados, cree que cubren totalmente el desarrollo de los videojuegos?

5. ¿Cree que sea necesario el empleo de un modelo para estandarizar en desarrollo de los videojuegos en el centro?

6. ¿Cree que con la propuesta realizada se podrá organizar la fase inicial de la realización de los videojuegos en el centro?

En cada entrevista realizada se verifica, además, la opinión de los especialistas teniendo en cuenta los parámetros siguientes:

- Importancia

- Posibilidad de aplicación
- Eficacia
- Orden y estructura
- Satisfacción de las necesidades

3.3 Criterio de los especialistas

Especialista # 1:

Opinión: De forma general para los proyectos de videojuegos que se desarrollen en la universidad, esta propuesta es importante pues considera concentrar las características comunes a todos los videojuegos. Aunque se deba aplicar para conocer si esta correctamente elaborada, posee las actividades necesarias para que el desarrollo del proceso se realice con la calidad requerida. Es aplicable a los proyectos videojuegos y en sus actividades resumen los objetivos necesarios de estos productos. Establece un orden lógico con que se deben desarrollar las actividades y en general resulta ser una propuesta de mucha significación para los sistemas de entrenamiento favoreciendo a la reutilización

Especialista # 2:

Opinión: La propuesta posee gran importancia, brinda una ayuda para que el proceso del desarrollo de los videojuegos. A su vez es necesaria para elaborar un proceso de Especificación de Requisitos en los productos de videojuegos con las características fundamentales para lograr alta calidad. Cumple con las actividades que se deben desarrollar en los videojuegos y es aplicable a los mismos. Puede que no se pueda asegurar que se cumplan todos los objetivos de un proyecto específico hasta que no se aplique al mismo, a simple vista no se puede asegurar. Su orden lógico establece correctamente las prioridades que posee un videojuego, satisfaciendo las necesidades de los mismos. De manera general la propuesta es simple y sencilla, aunque se le pueden agregar más elementos, de acuerdo con las necesidades de cada producto.

Especialista # 3:

Opinión: Cumple con las actividades a desarrollar durante la administración de requisitos. Resulta ser conveniente, necesaria e importante para los proyectos que dedican su funcionamiento al desarrollo de videojuegos serios. Es aplicable a los proyectos y contiene actividades necesarias que de forma directa garantizan la eficacia en el proceso de administración de requisitos. Está estructurada de manera lógica y entendible, fundamentando el cómo desarrollar las actividades. Es conveniente que dentro del Guión de Juego también se especifique hacia que público se dirigirá el juego.

Especialista # 4

Opinión: La propuesta está muy buena, bastante completa y posee buena estructura. Constituye un trabajo con gran valor para el desarrollo del proceso de especificación de requisitos en los videojuegos y es aplicable a los mismos. Las actividades definidas satisfacen los objetivos esenciales a desarrollar en los proyectos específicos y los artefactos que se generan pueden ser reutilizado y a la vez se ajustan a las necesidades de los mismos. Se adecua a lo establecido en el PM ayudando al mismo a ser más entendibles para los proyectos que desarrollen software del tipo de videojuegos. En general la propuesta es un paso para la estructuración del proceso de desarrollo de los productos en la línea de videojuegos.

Especialista # 5

Opinión: Existe un correcto desarrollo de las actividades propuestas, la estrategia de ajuste está bien elaborada, se realizó compatible a los formatos y descripciones del PM. Posee un orden lógico y buena estructura. La propuesta puede tener grandes posibilidades de uso como una guía de adaptación para los proyectos que desarrollen videojuegos. Se ajusta al desarrollo del proceso de administración de requisitos, determinando las principales necesidades que se deben desarrollar en los proyectos estudiados. Encuentro muy interesante el. Creo que puede convertirse en una guía que apoye la actividad de especificación de requisitos pues cuenta con un conjunto de

elementos que orientan al analista proporcionando una alta reutilización.

Especialista # 6:

Opinión: Considera importante la aplicación de la propuesta, además considera que cumple con las necesidades de estos tipos de proyectos y su futura aplicación solucionaría muchos problemas que se tienen a la hora de desarrollar un videojuego. La propuesta abarca de manera ordenada y lógica el cómo realizar las principales actividades en el proceso. El desarrollo de la misma en general podría convertirse en un paso de avance para ajustar todos los procesos a los tipos de proyecto que se desarrollen en el centro y en la universidad.

3.4 Análisis de los resultados

Teniendo en cuenta los criterios emitidos por los especialistas sobre la propuesta brindada se procede a realizar un análisis de los criterios emitidos por estos mediante estadísticas para verificar el nivel de aceptación de la propuesta con este fin se definen los siguientes valores para evaluar cuantitativamente la propuesta

Criterio	Valores
Muy Alta	5 puntos
Alta	4 puntos
Media	3 puntos
Baja	2 puntos
Muy baja	1 punto

Tabla 3. Para evaluar cuantitativamente la aceptación.

Parámetros\ Especialistas	1	2	3	4	5	6	Promedio
Importancia	5	5	5	5	5	5	5.00
Posibilidad de aplicación	5	5	4	5	5	5	4.83
Eficacia	5	4	4	5	4	5	4.50
Orden y Estructura	5	5	5	4	5	5	4.83
Satisfacción de las necesidades	5	5	5	5	4	5	4.83
Promedio	5.00	4.00	3.83	4.00	3.83	5.00	4.80

Tabla 4. Resultado de las evaluaciones de los especialistas.

El promedio general del nivel de aceptación de la propuesta es 4.80 puntos, este valor se encuentra entre la variable Alta y la variable Muy Alta, es decir, la aceptación de la propuesta por parte de los especialistas ha resultado satisfactoria.

3.4.1 Cálculo de la concordancia entre los especialistas entrevistados

Para brindar una mayor validez a la propuesta se necesita determinar el coeficiente de concordancia o correspondencia de Kendall, estadígrafo que permite comprobar el grado de coincidencia entre los criterios o valoraciones dados por los especialistas en las entrevistas realizadas.

Para calcular el coeficiente de Kendall se hace uso de la tabla , determinando además los valores siguientes que se necesitan para el cálculo del coeficiente:

K: número de especialistas que intervienen en el proceso de validación, K=6.

N: cantidad de parámetros a evaluar, N=5.

Rj: es la suma de los valores asignados a cada parámetro por parte de los especialistas.

Rj: es la media de los valores.

$$\overline{Rj} = \frac{\sum_{j=i}^{n} Rj}{N}$$

Parámetros\ Especialistas	1	2	3	4	5	6	Rj
Importancia	5	5	5	5	5	5	30
Posibilidad de aplicación	5	5	4	5	5	5	29
Eficacia	5	4	4	5	4	5	27
Orden y Estructura	5	5	5	4	5	5	29
Satisfacción de las necesidades	5	5	5	5	4	5	29

Tabla5. Resultados de Rj

Calculando el valor Rj = 30+29*3+27/5 = 28.8

Calculando el coeficiente de Kendall:

$$W = \frac{12 * S}{K^2(N^3 - N)}$$

W = 12*4.8/6² (5³ -5) = 0.01333

Calculando Chi-Cuadrado: X² = K(N-1) W = 6(5-1)0.01333 =0.32

Se procede a comparar el X² real con el X² de una tabla de probabilidad (Anexo #4) con un error de 0.05 y si es menor que este entonces hay concordancia entre los especialistas.

X² real = 0.32 y X² de la tabla = 9.49

X² real< X² de la tabla por tanto hay concordancia entre los especialistas.

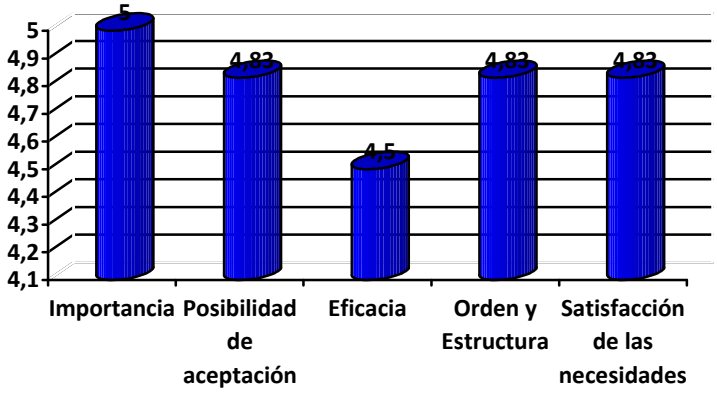

Figura 19. Representación del promedio de evaluación por parámetro.

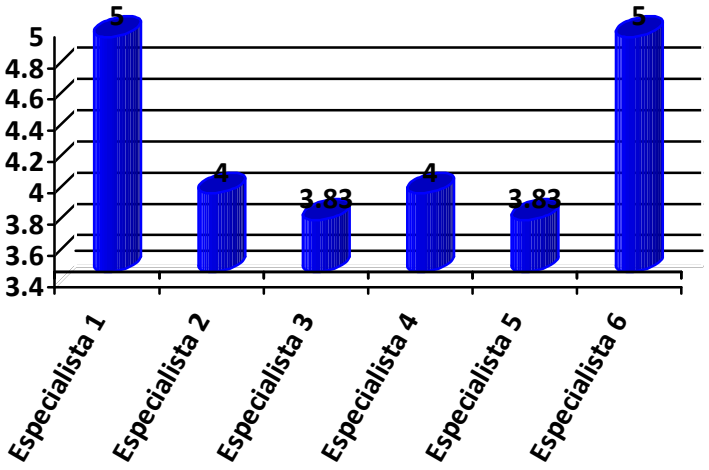

Figura 20. Representación de la concordancia entre especialistas.

Consideraciones Parciales

En el desarrollo de este capítulo se realizó la validación de la propuesta a partir de la técnica: consulta de especialistas, mediante la cual se realizaron entrevistas a especialistas en los temas fundamentales que abarca la investigación realizada. Se hizo un análisis de la información capturada mediante los cálculos probabilísticos para saber el nivel de concordancia entre los especialistas entrevistados y determinar el grado de aceptación de la propuesta. Para brindar una mayor visibilidad de los resultados alcanzados se realizaron graficas representado los parámetros analizados y los especialistas entrevistados. Finalmente se obtuvo la aceptación de la propuesta.

Conclusiones

El desarrollo de software se divide en varias etapas fundamentales. La correcta realización de las mismas depende en gran medida del empleo de modelos y metodologías que las estandaricen y logren guiar la ejecución de estas. En la presente investigación se propone una solución a los problemas detectados en la línea de videojuegos del CEDIN. Luego de culminada la investigación se logro arribar a las siguientes conclusiones:

- Con el uso de un modelo de procesos basado en líneas de productos de software para estandarizar el proceso de realización de los videojuegos se logra un alto grado de reutilización de activos de software.
- Con la realización de un modelado del dominio se logró conocer las principales conceptualizaciones que intervienen en el negocio. Dando una idea clara de su funcionamiento.
- El conocimiento de los requisitos por módulos que pueden estar presentes en los videojuegos permite al equipo de desarrollo contar con una orientación específica en los inicios de cualquier futura realización de este software.
- Con la estructuración de una propuesta arquitectónica enfocada al desarrollo de los videojuegos se logra brindar un soporte para la correcta realización de los mismos. La utilización del patrón arquitectura en 3 capas permite aprovechar la estrecha relación que existe entre las diferentes clases y módulos de un videojuego.

La investigación expone además la validación de los resultados para lo que se seleccionó la técnica consulta a especialistas mediante entrevistas relacionadas con los temas abordados, obteniendo como resultado la aceptación de la propuesta.

Recomendaciones

De forma general, la solución propuesta describe toda una serie de aspectos significativos referentes al futuro desarrollo de una aplicación de videojuegos. No obstante como parte un proceso de mejoras continuas se proponen las siguientes recomendaciones:

- Desarrollar los productos de la línea de videojuegos siguiendo los aspectos que en la solución se proponen.

- Mantener esta propuesta en constante refinamiento.

- Capacitar a los analistas del centro que estén vinculados a los proyectos de videojuegos para que desarrollen la propuesta con la calidad requerida.

- Desarrollar otros activos de software reutilizables no contenidos en la propuesta.

Bibliografía y Referencias bibliográficas

1. **Montilva, Jonas A.** Desarrollo de Software Basado en Líneas de Productos de Software. IEEE Computer Society Región 9. Universidad de los Andes, Facultad de Ingeniería. Dpto. de Computación. Mérida, Venezuela 2006.

2. **Díaz Redondo, Rebeca P.** "Reutilización de requisitos funcionales de sistemas distribuidos utilizando técnicas de descripción normal". Dpto. de Enxeñería telemática ETSE de Telecomunicación, Tesis doctoral, Febrero 2002.

3. **[Sommerville, 1995] Sommerville, Ian.** "Ingeniería del software", Séptima Edición,1995

4. **[Ivar, 2000] Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* La Habana : Editorial Félix Varela, 2004. ISBN: 978-84-7829-0369.

5. **Brooks Jr, Frederick.** *The mythical man-month.* s.l. : Addison-Wesley, 1975.

6. **Buschmann, F., R. Meunier, et al. (1996).** Pattern – Oriented Software Architecture. A System of Patterns.

7. **Alexander, C., S. Ishikawa, et al. (1977).** A Pattern Language: Towns, Buildings, Construction.

8. **FREE DOWNLOAD MANAGER.** 2007. Visual Paradigm for UML . [En línea] 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p.

9. **ATI.** Asociación de Técnicos de Informática. 2009. http://www.ati.es/spip.php?article1136.

10. **Montilva, Jonas A.** Desarrollo de Software Basado en Líneas de Productos de Software. IEEE Computer Society Región 9. Universidad de los Andes, Facultad de Ingeniería. Dpto. de Computación. Mérida, Venezuela 2006.

11. **Montilva, Jonas A.** Modelos de Procesos para el Desarrollo de Software Orientado a Objetos. Noviembre, 2000.

12. **Martínez, Alfonso & Cervantes, Humberto.** Diseño y construcción de una arquitectura de línea de producto de sistemas PACS, 2006.

13. **4. Ocharán Hernández, Jorge O. & Cortés Verdín, María K.** Documentando Arquitecturas Orientadas a Aspectos para Líneas de Productos de Software. México, 2006.

14. **Mellado Torío, Julio.** "Estrategias de pruebas de líneas de producto de sistemas de tiempo real especificados con diagramas de estados jerárquicos". Tesis (Doctoral), 2004.

15. **Especificación FODA:** Disponible en: www.sei.cmu.edu/domain-engineering/FODA.html.

16. **Hamar, Vanessa.** Aspectos metodológicos del desarrollo y reutilización de componentes de software. Universidad de los Andes, Facultad de Ingeniería, Post-grado de computación. Tutor: Jonas Montilva. Mérida, Noviembre 2003.

17. **Prieto-Diaz, R.** Domain Analysis for Reusability. En: Proceedings of COMPSAC'87, Tokyo, Japan, (23-29), October 1987.

18. **Prieto-Diaz, R.** Domain Analysis: An Introduction. En: Software engineering Notes, ACM SIGSOFT, 1990, vol.15, n...02, p.47-54.

19. **Barrios Albornoz, Judith.** El proceso de Desarrollo de Software. Modelos, Enfoques y Métodos. Escuela de Ingeniería de Sistemas. Universidad de los Andes. Enero, 2006.

20. **Eito Brun, Ricardo.** Técnicas de análisis de dominio: organización del conocimiento para la construcción de sistemas de software. 2006.

21. **Díaz Redondo, Rebeca P.** "Reutilización de requisitos funcionales de sistemas distribuidos utilizando técnicas de descripción normal". Dpto. de Enxeñería telemática ETSE de Telecomunicación, Tesis doctoral, Febrero 2002.

22. **Mena López, Santiago.** Simulación virtual & SIG. Centro de Entrenamiento Operativo Táctico Simulado. Quito, Ecuador. 2008.

23. **Jacobson, Ivar y Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. Madrid, España : Addison-Wesley, 2000. 84-7829-036-2.

24. **Kruchten, Philippe.** El Modelo de "4+1" Vistas de la Arquitectura del Software. s.l. : IEEE Software, Noviembre, 1995.

25. **Kruchten, Philippe.** *The Rational Unified Process: An Introduction. s.l.: Addison-Wesley, Marzo 2000. 0-201-70710-1.*

26. **Mazzini, Daniel.** Patrones de Diseño. ppt. Ubica Solutions.

27. **Montaldo, Diego Fernando.** Patrones de Diseño de Arquitecturas de Software Enterprise. Tesis. Noviembre 2005. Departamento de Computación. Facultad de Ingeniería. Universidad de Buenos Aires.

28. **Silva, Andrés.** Ingeniería de Requisitos. ppt.

29. **Mendoza Sánchez, María A.** Metodologías De Desarrollo De Software. 2004.

30. **Carmona Ruiz, Álvaro Ernesto.** *De los patrones de análisis y de integración a los componentes de negocio. [ppt]. Bogotá, Colombia: s.n., 2005. Software Architect Heinsohn Software House S.A.*

31. Las 4+1 Vistas.http://synergix.wordpress.com/2008/07/31/las-4-mas-1-vistas/.

32. **Obeso, Ivan Agüera. 1999.** Estudio de herramientas de análisis y gestión de requisitos. 1999.

33. **Eva. 2005.** Entorno virtual de aprendizaje. Eva. [En línea] Ingeniería de software, 5 de Junio de 2005. http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=22434.

34. **Hill, Mc Graw. 1999.** INGENIERÍA DEL SOFTWARE Un enfoque práctico. . EEUU : Sexta edición, Capítulo 1, pág. 5., 1999.

35. **Kruchten P.** *Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software, pp.42-50. 1995. en línea* http://www.computer.org/portal/web/csdl/doi/10.1109/52.469759.

36. **Bosch.** *Design Patterns as Language Constructs: Journal of Object Oriented Programming (JOOP). 1998.*

37. **IEEE. (2000). "IEEE Std 1471-2000 IEEE** Recommended Practice for Architectural Description of Software-Intensive Systems -Description." from http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html

38. **Shaw, M. (1996).** Introduction to Software Architectures New perspectives on an emerging discipline.

Anexos

Anexo #1: Tabla Probabilística

	PROBABILIDAD			
Df	0,10	0,05	0,01	0,001
4	7,78	9,49	13,28	18,46
5	9,24	11,07	15,09	20,52
6	10,64	12,59	16,81	22,46
7	12,02	14,07	18,48	24,32
8	13,36	15,51	20,09	26,12
9	14,68	16,92	21,67	27,88
10	15,99	18,31	23,21	29,59
11	17,28	19,68	24,72	31,36
12	18,55	21,03	26,22	32,91
13	19,81	22,36	27,69	34,53
14	21,06	23,68	29,14	36,12
15	22,31	25,00	30,58	37,70
16	23,54	26,30	32,00	39,29
17	24,77	27,59	33,41	40,75
18	25,99	28,87	34,80	42,31
19	27,20	30,14	36,19	43,82
20	28,41	31,41	37,57	45,32
24	33,20	36,42	42,98	51,18
25	34,38	37,65	44,31	52,65