**Universidad de las Ciencias Informáticas**

**Facultad 5 "Entornos Virtuales"**

# Guía Metodológica para la Estimación del Tiempo de Duración de Proyectos de Realidad Virtual

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas**

**Autores:** Leuris Sánchez Bell.
Eugenia Castillo Villares.

**Tutor**: Ing. Jandrich Domínguez Fortién.

**Asesor**: MSc. Pedro Carlos Pérez Martinto

Ciudad de la Habana, Julio 2007

*"... es característico de una mente instruida, descansar satisfecha con el grado de precisión permitido por la naturaleza de cada asunto y no buscar la exactitud cuando sólo es posible una aproximación de la verdad..."*

*Aristóteles (330 a. C.)*

Declaración de Autoría.

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

**Leuris Sánchez Bell**

_____

Firma del Autor

**Jandrich Domínguez Fortién**

_____

Firma del Tutor

**Eugenia Castillo Villares**

_____

Firma del Autor

**Índice:**

Agradecimientos

*Por esta obra muestra de años de esfuerzos y sacrificios, donde no pocos han colaborado, ya sea estando a mi lado cuando los necesite o tan solo por servirme de impulso para seguir adelante, le agradezco a todos mis familiares, profesores, compañeros y amigos.*

**Índice:**

**Índice:**

## Resumen:

El presente trabajo está enmarcado en la Facultad 5 “Entornos Virtuales” de la Universidad de las Ciencias Informáticas; la investigación sienta sus esfuerzos en el estudio del comportamiento del Proceso de Desarrollo de Proyectos de Realidad Virtual de la facultad antes mencionada, donde se detecto una serie de problemas en la aplicación en la práctica de la Disciplina Ingeniería y Gestión de Software específicamente en el área de la Estimación del Tiempo en el Proceso de Desarrollo.

Para la solución de los problemas se propone una de guía metodológica para la estimación del tiempo de Duración del Proceso de Desarrollo de Aplicaciones de Realidad Virtual. Basándose en técnicas de estimación como la Desagregación Estructural del Proceso, el Juicio de Expertos y la comparación por Analogías.

Para su aplicación se tiene en cuenta dos momentos dentro del proceso de desarrollo, un primer momento se define para obtener una estimación temprana y un el segundo momento es para una estimación más madura que prediga un valor del tiempo mucho mas confiable. Para la primera estimación se propone utilizar las descripciones de los escenarios como métrica del proceso y para la otra estimación los puntos de casos de usos.

**Términos Claves:**

Proceso de Desarrollo, Gestión de Software, Diseño Tridimensional, Medidas de Software, Estimación.

## Índice.

## Introducción

La Industria del Software se desarrolla a un ritmo vertiginoso, aunque la producción sigue siendo aún baja y los costos muy elevados. Debido, según [FRANCO, 2006], en la mayoría de los casos, a la no aplicación de técnicas de Ingeniería y Gestión de Software.

A pesar de estas dificultades se hace notar en el contexto internacional, la consolidación de una industria que genera anualmente cifras astronómicas de efectivo. Por lo cual, está citada a convertirse en unos de los pilares del crecimiento económico de no pocos países. En Cuba la Industria del Software, sin estar ajena a los problemas antes mencionados, está llamada a convertirse en una significativa fuente de ingresos para el mejoramiento social.

Hechos, demuestran que el país observa como una posibilidad, la incorporación de sus productos en el mercado mundial de software: la creación del Ministerio de la Informática y las Comunicaciones, la creación de la Universidad de las Ciencias Informáticas (UCI), la revitalización y potenciación de los tecnológicos de informática y la universalización de la carrera de Ingeniería Informática son ejemplos ilustrativos.

Sobre la creación de la UCI se ha dicho: “La visión del futuro de la UCI es la de realizar producciones intelectuales que serán el sustento fundamental de Cuba, la idea es convertir la informática en Cuba en una de las ramas más productivas y apartadoras de recursos para la nación.” [RUIZ, 2007]

La UCI nace como una universidad de nuevo tipo que une formación y producción. La producción se orienta a las necesidades del mercado, debido a esto, hay un incremento de la actividad productiva y por consiguiente se hace necesario investigar procesos que den soporte al control de la calidad en los procesos de desarrollo de software llevados a cabo en ella. [RUIZ, 2007]

La infraestructura productiva se divide por facultades, como estructura fundamental de producción y docencia en la Universidad, donde se acopian las diferentes líneas de investigación de interés para el país. Una de las estas líneas de investigación, con mucha relación en los intereses del país se encuentra las aplicaciones de Realidad Virtual, por poder ser desarrollados para casi todas las esferas económicas y sociales.

A partir de este creciente interés del país se hace necesario el estudio de buenas prácticas de Ingeniería y Gestión de Software de Realidad Virtual y de esa manera contribuir a las aspiraciones, de insertase en el mercado mundial de software con productos de Realidad Virtual que tengan calidad.

Considerando que el interés por la calidad del software crece de forma continua, debido a que los clientes se han vuelto más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades, se ha sumado la exigencia de certificaciones para comercializar productos. Hoy día, es obligatorio en la industria de software la aplicación de uno o varios modelos de calidad. [FEBLES, PIÑERO, FERNÁNDEZ, NAPAL 2006]

Para la Gestión de Calidad de Software a nivel mundial se han seguido dos tendencias. La primera a seguir, son las reglas implantadas por las oficinas internacionales de estandarización para los productos y servicios a través de las normas ISO y la IEEE y la segunda a seguir son las creadas específicamente para el mundo del software como CMM y CMMI. [FEBLES, PIÑERO, FERNÁNDEZ, NAPAL 2006]

Los beneficios de aplicar un Modelo de Gestión de la Calidad son muchos, los cuales se pueden alcanzar con el establecimiento de procesos como la planificación, el control, el mejoramiento para el aseguramiento de la calidad del proceso. En este último se ha de tener en cuenta que en su concepto, la calidad no radica únicamente en la satisfacción del cliente, sino también de la calidad del proceso; es decir, de las interrelaciones entre las tecnologías, los recursos humanos y su estructura.

Si se recurre a observar los beneficios del establecimiento de un Modelo de Calidad, se puede percatar que se podría, obtener mejor cumplimiento de los tiempos de entrega, estimando a través de un riguroso proceso.

Para conocer el comportamiento de esta área del conocimiento en los proyectos productivos de Realidad Virtual, se realizó un estudio, donde se reveló que en no pocos proyecto de este tipo, llevados a cabo por la Facultad ”Entornos Virtuales” de la UCI, cuentan con un Procedimiento definido para la Estimación del Tiempo de Duración, los encargados de esta actividad en mucho de los casos no poseen mucha practica para desempeñarla, hay desconocimiento de las variedad de técnicas y métodos de estimación, consecuentemente a estas carencias se evidencia en ocasiones el incumplimientos de los compromisos con los clientes y el atropello de los implicados en los proyectos a través de jornadas

laborales intensas, afectando considerablemente la satisfacción de los clientes (por la demora de los proyectos) y de los trabajadores del proyecto (estudiantes y profesores).

Si se tiene en cuenta que: los departamentos económico-financieros han considerado imprescindible esta actividad (la estimación de la duración de los proyectos) para el estudio de la viabilidad de un proyecto y debido a la Situación Problémica antes esbozada; cabe plantear la siguiente interrogante como problema a resolver en este trabajo: ¿Cómo proporcionar a la Facultad "Entornos Virtuales" de la Universidad de las Ciencias Informáticas un medio para la Estimación del Tiempo de Desarrollo de un proyecto de Realidad Virtual?

El trabajo enmarca su estudio en el Proceso de Desarrollo de Software de Realidad Virtual aplicado en la Facultad 5 de la UCI, el cual constituye el objeto de estudio de la investigación; acciona fundamentalmente en las Metodologías de Estimación de Software como campo de acción de la investigación. Los autores considera que si los jefes de proyectos estuvieran mejor entrenados en técnicas y metodologías de estimación y pudieran ser guiados a través este Proceso, sus estimaciones podrían aportar a la obtención de un producto de calidad dentro de un rango aceptable de duración. Por lo que para dar solución al problema científico planteado, los autores propone un grupo de tareas, entre los que se encuentra proporcionar el conocimiento de las metodologías de estimación para facilitar su empleo en el Proceso de Estimación de Tiempo de Desarrollo de Proyectos de Realidad Virtual.

Como objetivo general del trabajo se plantea: Proponer una Guía Metodológica para la Estimación del Tiempo de Desarrollo de un proyecto de Realidad Virtual.

Entre las tareas a realizar para darle cumplimiento al objetivo de la investigación se encuentran:

- Analizar el entorno del conocimiento que explora la tesis.
- Analizar el Proceso de Desarrollo Software de Realidad Virtual que aplica la Facultad 5.
- Indagar y analizar sobre las Técnicas más utilizadas para la estimación de proyectos.
- Proponer las métricas relacionadas con el tamaño que pueden ser utilizadas en la estimación en un Proyectos de Realidad Virtual.
- Exponer la técnica de estimación que mejor se ajusta al problema.

- Describir el proceso de Estimación del Tiempo para Proyectos de Realidad Virtual.

El trabajo fue producto de la aplicación de los métodos investigativos que se describen a continuación:

- Métodos teóricos: El método histórico /lógico y el dialéctico para el estudio crítico de los antecedentes de la investigación, y para utilizar estos como punto de referencia: El método analítico/sintético al descomponer el problema de investigación en elementos por separado, para luego proponer una solución.

- Métodos empíricos: el método de la entrevista para obtener los problemas presentes en la investigación.

Como fruto de la aplicación de estos método investigativos; el documento de tesis quedo dividido en 4 capítulos, en el primero se documentan los fundamentos teóricos que constituyen las bases de la investigación, en el segundo capítulo se describe e identifican los problemas dentro del proceso de desarrollo de Aplicaciones de Realidad Virtual que implementa la Facultad 5 de la Universidad de las Ciencias Informáticas, en el tercer capítulo se describen las métricas de software y técnicas de estimación más utilizadas así como los modelos de estimación propuestos por conocidos autores que servirán de base para la validación de la propuesta recogida en el capitulo cuatro.

## Capítulo 1 Fundamentación Teórica

### 1.1 Proceso de Desarrollo de Software

Desde el momento en que las empresas, comenzaron a considerar las aplicaciones informáticas como productos industriales, se comenzó un proceso de mejora de la producción y del rendimiento; como consecuencia de la introducción de la Ingeniería de Software al Proceso de Producción.

Como una aproximación al significado de esta afirmación se encontró que "Ingeniería de Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadoras y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software. [BOEHM, 1976]

El propósito es generar y mantener sistemas de software dentro de las restricciones de tiempo, funcionalidad y costos acordados con el cliente. Las metas de esta disciplina tecnológica son mejorar la calidad de los productos desarrollados y aumentar la productividad de los ingenieros de software. La aplicación de las buenas prácticas de ingeniería de software es esencial para lograr un producto con calidad. [PRESSMAN, 2002]

Esta definición como acercamiento, hace una descripción bastante acertada de lo que quieren los autores idealizar como Proceso de ingeniería de software o desarrollo de software, como también se le llamara indistintamente en este trabajo. Para ampliar la veracidad de lo ante planteado, uno de los concepto que abordan los desarrolladores del trabajo es el dado por el prestigioso Instituto de Ingeniería Eléctrica y Electrónica, al cual se les identifican por las siglas de Institute of Electrical and Electronics Engineers (IEEE).

El mismo ha desarrollado una definición en cuanto a este tema, alegando que Ingeniería de software es (1) La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software. (2) El estudio de enfoques como en (1). [HEREDIA, 2003]

Pressman hace un conceso de estas definiciones, con la cual los autores coinciden, al plantear que el Proceso de Desarrollo de Software se puede caracterizar como un marco común que define un núme-

ro de actividades comunes que son aplicables a todos los proyectos, con independencia de su tamaño o complejidad. Es decir una colección de tareas de trabajo de ingeniería de software, hitos de proyectos, productos de trabajo y puntos de garantía de calidad, que permiten que las actividades dentro del marco de trabajo se adapten a las características del proyecto del software y a los requisitos del usuario.

Entre las prácticas más difundidas de desarrollo de software se encuentran el Proceso Unificado de Desarrollo (RUP) y las Metodologías Agiles de Desarrollo, estas dos grandes tendencias de desarrollo se podrían categorizar como tendencias opuestas de desarrollo por la forma de ver el proceso pero ambas concuerdan en que del propósito final es la obtención de un producto de calidad que satisfaga las necesidades de los clientes.

### 1.1.1 Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo, se define como un proceso de desarrollo de software, es decir es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. [JACOBSON, BOOCH, RUMBAUGH, 2000]

El proceso RUP combina un conjunto básico de mejores prácticas con una serie de complementos opcionales del proceso a fin de dar cabida y soporte a proyectos de cualquier envergadura o alcance. Dicho proceso utiliza el Lenguaje Unificado de Modelado (UML), para preparar todos los esquemas de un sistema software. Donde el UML es una parte esencial de RUP. Oras de las características fundamentales de este proceso se define una de las mejores prácticas de desarrollo de software como es ser iterativo e incremental, estar dirigido por casos de usos y centrado en arquitectura. [JACOBSON, BOOCH, RUMBAUGH, 2000]

El proceso iterativo e incremental consta de una secuencia de iteraciones. Iterativo significa que el software evolucione a través de un número de ciclos. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en

curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

El ser dirigido por casos de uso es una característica propia de RUP. Esto significa que se utilicen casos de uso para hacer la especificación funcional, en la etapa de análisis se compone de la realización en análisis de cada caso de uso, en la de diseño se compone de la realización en diseño de cada caso de uso, y en la etapa de implementación los desarrolladores desarrollan casos de uso, que son probados con casos de prueba derivados de los casos de uso.

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo.

Según [JACOBSON, BOOCH, RUMBAUGH, 2000] el equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

RUP estructura el proceso en cuatro fases (Inicio, Elaboración, Construcción y Transición), la cual denomina estructura dinámica debido a que varía en el tiempo y dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

**Inicio:** Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar qué recursos deben ser asignados al proyecto.

**Elaboración:** El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

**Construcción:** La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

**Transición:** La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

La otra estructura del proceso es la denominada Estructura Estática del Proceso donde se definen en el proceso en los roles, actividades, artefactos y flujos de trabajo.

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una actividad en concreto es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto. Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final. Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables**.**

### 1.1.2 Metodologías Ágiles de Desarrollo

A principios de la década del ’90, surgió un enfoque de desarrollo de software que fue bastante revolucionario para su momento ya que iba en contra de la creencia de que mediante procesos altamente definidos se iba a lograr obtener software en tiempo, costo y con la calidad requerida. El enfoque fue planteado por primera vez en el libro Rapid Application Development (RAD). RAD consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código de forma automática tomando como entradas sintaxis de alto nivel. [SCHENONE, 2004]

Las Metodologías Ágiles no fueron reconocidas, como tales en la comunidad de los desarrolladores de software hasta el surgimiento de una de las metodologías utilizada como arquetipo: La Programación Extrema (XP de Extreme Programming) surge de la mente de Kent Beck, junto a Ward Cunningham. [BECK, 2000]

Los principios de XP aludidos por Beck fueron enunciados por [SCHENONE, 2004] de la siguiente forma:

- El juego de Planeamiento: Rápidamente determinar el alcance del próximo realce mediante la combinación de prioridades del negocio y estimaciones técnicas. A medida que la realidad va cambiando el plan, actualizar el mismo.

- Pequeños Raleases: Poner un sistema simple en producción rápidamente, luego liberar nuevas versiones en ciclos muy cortos.

- Metáfora: Guiar todo el desarrollo con una historia simple y compartida de cómo funciona todo el sistema.

- Diseño Simple: El sistema deberá ser diseñado tan simple como sea posible en cada momento. Complejidad extra es removida apenas es descubierta.

- Pruebas: Los programadores continuamente escriben pruebas unitarias, las cuales deben correr sin problemas para que el desarrollo continúe. Los clientes escriben pruebas demostrando que las funcionalidades están terminadas.

- Descomposición en factores: Los programadores reestructuran el sistema sin cambiar su comportamiento para remover duplicación, mejorar la comunicación, simplificar, o añadir flexibilidad.

- Programación de a Pares: Todo el código de producción es escrito por dos programadores en una máquina.

- Propiedad Colectiva del Código: Cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento.

- Integración Continua: Integrar y hacer builds del sistema varias veces por día, cada vez que una tarea se completa.

- Semana de 40 horas: Trabajar no más de 40 horas semanales como regla. Nunca trabajar horas extras durante dos semanas consecutivas.

- Cliente en el lugar de Desarrollo: Incluir un cliente real en el equipo, disponible de forma full-time para responder preguntas.

- Estándares de Codificación: Los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo.

De las características que promueve XP esta el involucramiento del usuario en el proceso, la integración de la prueba de software a través de todo el ciclo de vida y un enfoque colaborativo y cooperativo entre todos los interesados. [BECK, 2000]

Muchos de estos principios son la base para caracterizar a las metodologías ágiles las cuales presentan un modelo de desarrollo iterativo- incremental, con pequeñas entregas, ciclos rápidos y cooperativos donde los desarrolladores y los usuarios trabajan juntos en estrecha comunicación, y donde es capaz de adaptarse a los cambios, siendo un método muy simple y fácil de aprender. Tienen como

características el ser adaptativas en vez de predictivas, centradas en la gente o en los equipos, iterativas, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva, y que requieren que el negocio se involucre en forma directa.

### 1.1.3 ¿Qué es el Proceso de Desarrollo de Proyectos de Realidad Virtual?

Para darle respuesta a esta interrogante, se hará ante de todo un bosquejo del término Realidad Virtual, que es en sí mismo es paradójico ya que se compone de dos conceptos prácticamente opuestos, la Real Academia Española define la realidad como algo que se entiende a la existencia real y efectiva de una cosa, o sea aquello que tiene existencia verdadera y efectiva. Y Virtual: Que tiene virtud de producir un efecto. Frecuentemente en oposición a lo efectivo o real. El término Realidad Virtual por tanto puede dar un significado confuso “Realidad No Real”. Por esta razón algunos investigadores prefieren usar otros términos como: Ciberespacio, Realidad Artificial, Ambientes Sintéticos [PARRA, GARCÍA, SANTELICES, 2001]

La Enciclopedia Británica, define este término (Realidad Virtual) como: La utilización de la modelación y simulación que habilitan a la persona a interactuar con una visión tridimensional o a través de sensores ambientales. De tal forma que las aplicaciones sumergen al individuo en ambientes generados por computadora que simulan la realidad a través del uso interactivo de estos dispositivos, los cuales envían y reciben información.

Esta enunciación se acerca más a la definición que se necesita comprender para este trabajo ya que el mismo también es usado para llamar una rama de las ciencias de la computación que involucra al desarrollo de tales sistemas.

Partiendo de los conceptos anteriores se puede identificar claramente cuatro características de la Realidad Virtual:

- **Simulación:** Capacidad de replicar aspectos suficientes de un objeto o ambiente de forma que pueda convencer al usuario de su casi realidad.

- **Tridimensionalidad**: Tiene que ver directamente con la manipulación de los sentidos del usuario, principalmente la visión, para dar forma a el espacio virtual; los componentes del mundo vir-

tual se muestran al usuario en las tres dimensiones del mundo real, en el sentido del espacio que ocupan, y los sonidos tienen efectos estereofónicos (direccionalidad).

- **Interacción:** Rasgos que permiten al usuario manipular el curso de la acción dentro de la aplicación de realidad virtual, permitiendo que el sistema responda a los estímulos de la persona que lo utiliza; creando interdependencia entre ellos.

- **Percepción**: Permite la interacción con los sentidos del usuario (vista, oído y tacto). Según la complejidad del sistema los elementos externos utilizados para producir estas sensaciones serán más o menos simples pudiendo ser un simple ratón de ordenador o unos cascos, más sensores de posición en una cabina virtual.

Haciendo una combinación de los conceptos tratados, hasta ahora durante el transcurso de la investigación se puede llegar a la conclusión de que Proceso de Desarrollo de Proyecto de Realidad Virtual es: una colección de tareas de trabajo de Ingeniería y Gestión de Software, que tiene por objeto crear un producto de Realidad Virtual único en un tiempo finito, que permita la visualización, manipulación e interacción con ordenadores a trabes de un ambiente artificial.

La interrelación de estos conceptos provoca una nueva paradoja para los profesionales de la informática dedicados a desarrollar este tipo de aplicación, consecuencia de las características que se enunciaron anteriormente con relación al término de Realidad Virtual. Conseguir que las aplicaciones cumplan con estos principios, demanda la implementación de Procesos de Desarrollos muy particulares debido a que solo se alcanzan a través de procesos incomparables con los que se están acostumbrados a lidiar en el mundo del desarrollo de software como Procesos de Desarrollo como RUP o MAD, por ejemplo el Diseño de los Escenarios Tridimensionales, en los que se le da soporte al cumplimiento de la característica de los mundos virtuales.

Las características expuestas de las Aplicaciones de Realidad Virtual se han de tener en cuenta para definir los Proceso de Desarrollo de los Proyectos de Realidad Virtuales. Por ejemplo la simulación y la iteración son características que definen mucho el Paradigma de Programación. La realidad está compuesta por la interrelación de una gran variedad de objetos y estas aplicaciones tienen el objetivo de reproducirlos. Por cuanto no se presupone mucho trabajo abstraerse de la realidad al aplicar el

Paradigma Orientado a Objeto ya que los modelos tridimensionales corresponden a escenarios con muchas clases diferentes de objetos.

Un programa de Realidad Virtual presenta una estructura de mayor complejidad de lo normal en un software computacional, debido a que, por un lado, constituye un ambiente tridimensional, a la vez que dispone de una programación específica y el control de múltiples dispositivos externos, todo funcionando y modificándose en tiempo real.

Esto establece una amplitud sofisticada de características, descritas por dos procesos principales: el diseño de la aplicación, en el cual se utiliza el programa para diseñar el escenario 3D, incorporar el audio, programar los comportamientos y configurar dispositivos, y otra actividad es la navegación interactiva, en que se puede visualizar, recorrer y manipular el ambiente virtual de acuerdo con lo preparado en el diseño a través de los que se denomina, “guión del mundo virtual” el cual define las acciones que los objetos realizan por ellos mismos y cuando interactúe el usuario con éstos.

### 1.2 ¿Proceso de Gestión de Proyectos de Software?

Para ingresar en el tema cabe plantear la siguiente interrogante: ¿Qué es un Proyecto de Software? A lo largo de historia de la ingeniería informática se han adoptado un sin número de definiciones para esta interrogante entre los que se destacan definiciones no muy lejos de las utilizadas en otros campos del conocimiento como las siguientes:

“Conjunto de actividades, planificadas, ejecutadas y supervisadas que, con recursos finitos, tiene por objeto crear un servicio o producto único”. [AJENJO, 2000].

“Combinación de todos los recursos necesarios, reunidos en una organización temporal, para la transformación de una idea en realidad”. [COS, 1999]

“Trabajo en el que se define un principio y un fin (el tiempo), se especifica un resultado deseado (alcance), cumpliendo con unos requisitos de calidad (acabado) y coste (presupuesto)”. [COURTER, 2000]

Desde estos, puntos de vistas, conceptuales los autores identifican las siguientes características que hacen a un proyecto distinto a cualquier otra actividad dígase proceso o tarea:

- Actividades planificadas, ejecutadas y supervisadas
- Disponibilidad limitada de recursos
- Limitado en el tiempo
- Con un resultado único

Con el planteamiento de las características se puede dar entrada a la respuesta de la interrogante que le da nombre a este epígrafe. Para los autores es la actividad encaminada a garantizar el cumplimiento de las tareas expuestas en el Proceso de Desarrollo de Software para darle cumplimiento a un Proyecto de este tipo. Esta definición es muy sencilla, por lo que los autores considerando, que no satisfaga las necesidades de lectores experimentados en el tema, se dan a la tarea de abordar conceptos dados por prestigiosas organizaciones, como los que le sigue a continuación.

“Es la aplicación de conocimientos, aptitudes, herramientas y técnicas a las actividades del proyecto, encaminados a satisfacer o colmar las necesidades y expectativas de las entidades y organizaciones involucradas en un Proyecto”. [HEREDIA, 2003]

“Es la planificación, organización, seguimiento y control de todos los aspectos de un proyecto, así como la motivación de todos aquellos implicados en el mismo, para alcanzar los objetivos del proyecto de una forma segura, y satisfaciendo las especificaciones definidas de plazo, coste y rendimiento. Ello también incluye el conjunto de tareas de liderazgo, organización y dirección técnica del proyecto, necesarias para su correcto desarrollo” [PEREÑA, 1996]

Como bien se plantea los conceptos anteriores la Gestión de Proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema. Los autores quedan complacidos con esta afirmación, que corrobora la definición inicial dada por ellos. La planificación y el seguimiento son dos actividades principales del Proceso de Gestión cabria preguntarse: ¿Qué actividad le da soporte a ellas? La respuesta a esta interrogante se encontró en el trabajo de Ana M. Moreno donde identifica una tercera actividad dentro del proceso de gestión, como otras de las fundamentales, la estimación.

Una estimación o predicción no es un objetivo, sino una valoración probabilística. En [DeMarco,1982] al definirla plantea: “una estimación es una predicción del valor de una variable, que es igualmente probable que esté por encima o por debajo del resultado real”. Del modo que para indicar su calidad se ha de definir un rango, representado como una tripleta: el valor más probable (que es, la mediana de la distribución), mas los límites superior e inferior del valor.

En términos estadísticos a estos límites inferior y superior se les denomina intervalos de confianza. La calidad de la estimación está dado por la efectividad de la predicción o sea que el valor real de la variable debe de estar dentro del rango de confianza de la predicción.

### 1.2.1 La estimación en la Gestión de Proyectos

La estimación es la primera actividad dentro del proceso de gestión proyectos. En este marco puede definirse la estimación como “el proceso que proporciona un valor a un conjunto de variables para la realización de un trabajo, dentro de un rango aceptable de tolerancia.” [MORENO, 2000]

Las variables a estimar dentro del marco de un proyecto de desarrollo de software pueden ser muchas. Estas van desde la estimación de la duración del proyecto, el esfuerzo y el costo para realizar un proyecto etc., hasta la estimación de índices de calidad tales como cantidad de defectos por unidad de medida, casos de prueba por caso de uso, etc. [CAO, 2006]

A continuación los autores citaran algunas consideraciones que han tenidos varios autores sobre el tema:

“la estimación del proyecto de software puede dejar de ser un oscuro arte para convertirse en una serie de pasos sistemáticos que proporcionen estimaciones con un grado de riesgo aceptable” [PRESSMAN, 2002]

“Estimar es echar un vistazo al futuro y aceptamos resignados cierto grado de incertidumbre. Aunque la estimación, es más un arte que una Ciencia, es una actividad importante que no debe llevarse a cabo de forma descuidada”. [CONCEPCIÓN, 2005]

“La estimación es una actividad que se debe realizar desde las primeras etapas en que se plantea la idea del proyecto informático de construcción de software”. [BASURTO, DÁVILA, MELÉNDEZ, 2005]

Sin lugar a dudas es la estimación una actividad de gran importancia, en el proceso de gestión, cuando se necesita saber cuánto costara en esfuerzo, materiales o tiempo llevar a cabo un proyecto informático.

### 1.2.2 Estimación en la Planificación

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos costos y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto. Además las estimaciones deberían definir los escenarios del mejor caso, y peor caso, de modo que los resultados del proyecto pueden limitarse.

Según [SOMMERVILLE, 2002] “La estimación y calendarización (planificación) del proyecto se llevan a cabo de forma conjunta”. Los autores viendo esta relación han llegado a considerar a la planificación como la actividad organizacional que le da uso a los valores de las variables estimadas.

Con estos planteamientos se ha de concluir que la planificación y la estimación son actividades relacionadas, pero difieren en su alcance y propósito. La estimación normalmente está orientada al proyecto en su conjunto, mientras que la planificación está dirigida a los individuos.

La estimación, de forma general, se realizará el principio del proyecto, precisamente para saber cuánta gente se necesita o cuanto durara una etapa en el proceso y la planificación es la etapa donde se asigna exactamente quién hace qué y en cuanto tiempo. La etapas de estimación y planificación se harán de forma continua a todo lo largo del proceso de desarrollo de software al inicio de cada etapa.

### 1.2.3 El Seguimiento

El seguimiento es la recolección de datos y su almacenamiento, sobre tiempos, recursos, costes, e hitos asociados con un proyecto, para el análisis y estudio de la evolución real de dicho proyecto, comparando el progreso real con el planificado. En esencia el seguimiento es la actividad que comprueba

la aplicación de lo planificado y la calidad de las estimaciones, haciendo comparaciones entre los valores estimados y los reales. [SOMMERVILLE, 2002]

Esta actividad está estrechamente relacionada con las otras antes mencionadas (la estimación y la planificación). Debido a esto los autores consideran que se puede decir que una carece de sentido sin la otra, en cuanto a que: cuando se tiene una estimación sobre el proyecto que se va a desarrollar, se podrá definir una planificación, siempre dentro del marco de esta estimación, es decir, la salida del proceso de estimación son las entradas del proceso de planificación.

Una vez realizada la planificación comenzara el seguimiento del proyecto. Por lo tanto, las entradas del proceso de seguimiento serán la estimación y planificación del proyecto, además de los datos reales recogidos mientras evoluciona el proyecto. Estos datos recogidos dentro del proceso de seguimiento será la base de próximos procesos de estimación. [MORENO, 2000]

### 1.3 Detalles para dar comienzo a un Proceso de Estimación de Proyectos de Software

La estimación es un proceso continuo, a medida que el proyecto avanza mas se conoce de él, y por lo tanto más parámetros están disponibles para introducir en un modelo de estimación.

Todo proceso de estimación esta divido en dos etapas, una primera etapa es donde se realizan los estudios de la viabilidad del proyecto o sea se establece el ámbito del software, en cual se toma en cuenta elementos como los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad. Es donde se requiere solo de los datos descriptivos necesarios para determinar el tiempo y el esfuerzo que llevara realizar un proyecto. Es donde se define si es viable para la organización continuar adelante en el proyecto. El ámbito se define como un pre -requisito para la estimación.

El segundo momento de la estimación es mucho más específico, va orientado a determinar etapas dentro del proceso de desarrollo y no solo genera datos para el estudio de viabilidad sino los suficientes como para establecer una planificación detallada de las etapas del proceso de desarrollo del proyecto. Es donde se establecen los valores necesarios de los recursos que se van a utilizan en el proceso.

Los recursos simulan una pirámide donde las Herramientas (Hardware y Software), son la base que proporcionaran la infraestructura de soporte al esfuerzo de desarrollo, en segundo nivel de la pirámide

se encuentran los Componentes reutilizables son, como su nombre lo indica, todo aquello que pueda volverse a utilizar; en la parte más alta de la pirámide se encuentra como el recurso primario, las personas (RR-HH).

Debido a la diferencia que hay entre los procesos de producción industrial y los procesos de producción de software en cuanto a que los últimos generan productos intangibles y que para ello se requiere de comunicación y coordinación intensivas entre el los implicados en el proceso, hace que los recursos humanos ocupen la cúspide de la pirámide de recursos.

El entorno es donde se apoya el proyecto de Software, llamado a menudo entorno de Ingeniería de Software, incorpora el Hardware y el Software a utilizar dentro del proceso. Es lo que proporcionara la plataforma que se requiere para producir el producto, que son el resultado de la buena práctica de la Ingeniería del Software.

### 1.4 ¡Las Métricas, Medidas, Medición o Estimación de Software!

Por algo más de cincuentas años el área del conocimientos que explora la medición del software a generado definiciones que han traído confusión en la utilización de los términos métricas de software y mediciones de software, sin embargo, en muchas literaturas los términos métrica, medida o medición son usados como vocablos análogos. [ZUSE, 1995].

Según la Norma ISO Medida es valor numérico para un atributo cuya magnitud se desea valorar en función de una escala concreta. [PFLEEGER, 2001]

Medición. Establece la terminología relacionada con el acto de medir software. Una medición (que es una acción) es un conjunto de operaciones cuyo objetivo es determinar el valor de un atributo dado de una entidad, utilizando una forma de medir. Los resultados de la medición son las medidas. [PFLEEGER, 2001]

Métrica: es un criterio para determinar la distancia entre dos entidades. [ZUSE, 1998]

Toda métrica es una medida, pero una medida no tiene por qué ser una métrica. El término adecuado es “medida” a pesar de que ambos se usan indistintamente como se había ante expuesto. [ZUSE, 1998]

La medición del software se refiere a derivar a un valor numérico para algún atributo de un producto de software o un proceso de software. Comparando estos valores entre ellos y con los estándares aplicados en la organización, es posible sacar conclusiones de la calidad del software o de los procesos del software. [SOMMERVILLE, 2002]

Otra definición de medición más acabada y con la cual los autores están totalmente de acuerdo es la siguiente:

“el proceso mediante el que se asignan números o símbolos a los atributos de las entidades del mundo real de manera que dichos atributos sean descritos de acuerdo a unas reglas previamente definidas.” [FENT, 1996].

El proceso de medición del software está dirigido por las necesidades de información en la organización que lo desarrollan. Por cada necesidad de información este proceso produce un producto de información que intenta satisfacerlo, conocido como modelo de información. El mismo describe cómo los atributos relevantes se cuantifican y se convierten en indicadores que proporcionaran la base para la toma de decisiones.

Partiendo de del planteamiento anterior y estableciendo un balance entre los conceptos abordados con anterioridad de medición y estimación, se puede percatar que ambos proceso están encaminados a satisfacer las necesidades de información sobre una entidad (procesos, productos o recursos) determinada. Según el momento en que requiera de esta información y la disponibilidad de datos el proceso que la proporcionara será el proceso de estimación o el de medición. El proceso de medición está más relacionado con el Seguimiento y la estimación más bien está relacionada con la Planificación.

# Capítulo 2 Estudios Preliminares

## 2.1 Antecedentes del Problema

A lo largo del proceso de madures de la industria del Software cubano, se han desarrollado una multitud de investigaciones, que ayudan a identificar problemas en diferentes etapas de la misma. Resultado de una de estas investigaciones fue la realizada por [FEBLES, 2000], [HERNÁNDEZ, 2001] entre los años 2000 y 2001 en la cual se identifican problemas tales como:

- Ausencia de la definición de los procesos de la empresa, siguiendo los principios de la ingeniería y la gestión de software
- No existe un procedimiento para llevar a cabo las actividades relacionadas con el desarrollo del software.
- No hay eventos predefinidos para hacer pedidos de cambios.
- No existe planificación del trabajo.
- No se puede dar seguimiento al progreso del proyecto (no hay definición de tareas, relaciones entre estas, cronogramas, puntos de chequeo, etc.)
- No se lleva el registro de quien hizo cada actividad relacionada con el proyecto (no se puede exigir responsabilidades, no se puede medir productividad, cantidad de errores cometidos a nivel individual).
- Al no planificar, no hay seguimiento, por lo que es imposible ver el estado del progreso del proyecto.
- No se puede comparar con proyectos anteriores y aprender de estos.
- No se pueden ver los problemas de calidad (cantidad de errores) en distintas áreas de trabajo.
- Existe dificultad para evaluar el trabajo de los especialistas al no poseer los mecanismos para detectar la magnitud del trabajo desarrollado.
- No existen procedimientos para obtener métricas que permitan planificar y controlar los procesos.
- Hay mala calidad en mucho del software que se produce en el país.

- Los proyectos están excesivamente tarde y los beneficios que pudieran obtenerse al utilizar los mejores métodos e instrumentos en las distintas etapas no se detectan en este medio indisciplinado y caótico de desarrollo

- Existe falta de comunicación efectiva entre los involucrados (usuarios, desarrolladores, administradores, clientes e investigadores)

Muchos de estos problemas fueron siendo erradicados a medida que la industria fue creciendo, cabe destacar que en la etapa en que se desenvolvió esta investigación se comenzaba a ver la industria como una posible fuente de ingreso de la economía cubana , en las etapas posteriores a esta investigación la industria sufrió un gran cambio al constituirse el Ministerio de la Informática y las Comunicaciones organismo encargado de establecer, regular y controlar las normas técnicas y operacionales de todas las actividades informáticas del país.

Posteriormente ya para el año 2004 surgieron otros estudios orientados a diagnosticar a la industria nacional. En los que se pueden identificar problemas más específicos dentro del Proceso de Desarrollo de Software entre estos se pueden destacarse [FEBLES, 2004]:

Organizativas

- Pocas empresas tienen definidos sus procesos a partir de los principios de la ingeniería y la gestión de software.

- Es pobre la comunicación efectiva entre los involucrados en la producción.

- No se brinda eficientemente el Soporte de Software.

Planificación y Seguimiento

- No existe una adecuada planificación del trabajo.

- No se puede dar seguimiento al progreso del proyecto (no hay definición de tareas, relaciones entre estas, cronogramas, puntos de chequeo, etc.).

- Falta de objetividad en la apreciación de la evolución del trabajo de los especialistas al no poseer los mecanismos para detectar la magnitud del trabajo desarrollado.

Calidad

- No se pueden detectar con antelación los problemas en el software afectando esto finalmente la calidad en los productos.
- No se lleva un control de los defectos encontrados en el proceso de desarrollo del software.

Mediciones

- No se miden los procesos, ni los productos con métricas rigurosas lo que implica que no se pueda controlar ni mejorar el proceso a partir de los resultados en su ejecución.

## 2.2 Análisis de la Situación de la Facultad 5

A partir de la aplicación entrevistas como técnicas de investigación, con el objetivo de conocer cómo se comporta el Proceso de Desarrollo de Software en la Faculta 5 de la Universidad de las Ciencias Informáticas, en la que se trazo temáticas:

- Conocer cuáles son las metodologías usadas en la facultad 5 para el desarrollo de software de realidad virtual.

- Tener una idea del cumplimiento de las actividades de planificación y seguimiento del proceso independientemente de la metodología de desarrollo que utilizan.

- Saber si se realizan mediciones y estimaciones en el proceso.

- Conocer si aplican métricas en el proceso de desarrollo.

Estas entrevistas se realizaron a ocho proyectos de un dominio de nueve que se dedican al Desarrollo de Aplicaciones de Realidad Virtual en la facultad, se obtuvieron resultados que ayudan a identificar

de forma concluyente los siguientes problemas en el proceso de desarrollo de proyectos de realidad virtual:

- Pocos proyectos tienen definidos sus marcos de desarrollo a partir de un proceso bien definido
- No se puede comparar con proyectos anteriores y aprender de estos.
- No existe una adecuada estimación del trabajo.
- No se miden los procesos, ni los productos con métricas que ayuden al proceso de estimación.
- No se realiza un seguimiento adecuado del proyecto. Lo que repercute en la calidad de las próximas estimaciones.
- Los registros de actividad relacionada con el proyecto carecen de especificaciones importantes con relación a la productividad de los involucrados en el Proceso.

De la identificación de estas carencias en el proceso de desarrollo de software, se puede concluir que con el establecimiento de buenas prácticas de estimación, planificación y seguimiento del marco de desarrollo de software se podrían mitigar estas carencias. De esta forma surgen las siguientes interrogantes: ¿Qué se puede medir en el Software? o ¿Qué Técnica se pueden utilizar para medir?.

## Capítulo 3 Análisis de las técnicas de Estimación

### 3.1 Que se puede medir en un Software

Según [FENTON, 1991] existen tres tipos de entidades a medir en el Desarrollo de Software, las cuales les dan una clasificación a las medidas, estas son:

**Procesos:** conjuntos de actividades relacionadas con el software. Por ejemplos: Medidas que cuantifican atributos como tiempo y esfuerzo.

**Productos:** cualquier elemento de configuración resultante de una actividad (programas, documentación, entregables,...) Ejemplos: Medidas de tamaño, complejidad de la estructura.

**Recursos:** entidades que una actividad necesita (personal, herramientas, métodos). Ejemplos: Medidas como el nivel de experiencia del personal en un lenguaje de programación.

En [FENTON, 1991] se continua diciendo que entre las características medibles de una entidad, conocida como atributos, prevalecen dos tipos: Los atributos Internos: aquellos que se pueden medir directamente, en función del proceso, producto o recurso en sí. Y los atributos Externos: aquellos que se pueden medir en función de cómo se relaciona el proceso, producto o recurso con su entorno o sea indirectamente.

Los valores de estos atributos se pueden tomar de dos maneras a través de medidas directas aplicando una herramienta de medición (técnicas o instrumentos) y las medidas indirectas a través de otros atributos, la medición indirecta es también conocida como estimación de medidas, donde se establece una relación entre medidas internas y externas de un atributo.

En el Ámbito del desarrollo de software las mediciones más utilizadas, son las orientadas al Producto y al Proceso debido a que son los valores relacionados con el marco de desarrollo del software tales como tiempo de desarrollo total (objetivo de este trabajo), esfuerzo de desarrollo del producto son muy importantes debido a que proporcionan información de gestión significativa, que obtenidas a tiempo, pueden ser usadas para mejorar la calidad de los procesos y por consiguiente de los productos.

Las medidas del producto pueden medir la complejidad del diseño, el tamaño del producto final o el número de páginas de documentación producida. Se ha de considerar que a partir de las medidas del producto se pueden llegar a conocer las medidas del proceso. Entre las métricas que intentan cuantificar los productos de software las más utilizadas son las que cuantifican el tamaño, entre estas se encuentran las líneas de código (LCO), los Puntos de Fusión (PF), los Puntos Característicos (PC), los Puntos de Objetos (PO), los Puntos de Casos de Uso (PCU) y los Escenarios Normalizados. [PÉREZ, 2002]

Las **LCO** como métrica fueron introducidas al inicio de la década del 70, es sencilla de utilizar y suministra una excelente información relacionada con el tamaño, pero tiene varios inconvenientes entre estos se encuentran que depende del lenguaje de programación, de la tecnología o plataforma en que se va a realizar el producto y además existen alrededor de 11 variantes de definiciones de este concepto, separadas en dos grupos: a nivel de programa y a nivel de proyecto.

Los **PF** es una de las métricas que se utilizan para determinar el tamaño de un producto, fue introducida en 1979 por Allan J. Albrecht especialista de IBM, refinándolo en 1984. Los PF miden el tamaño del software cuantificando la funcionalidad provista por el usuario basándose solamente en el diseño lógico y las especificaciones funcionales. Inicialmente fue concebido para sistemas clásicos de gestión. [IFPUG, 2000].

Los que les lleva a cuantificar los números de entradas, salidas, consultas del usuario, los números de archivos (interfaces internas) e interfaces externas. Lo cual la hacen en estos momentos una mala elección para otros tipos de software a pesar de ser independientes del lenguaje y la tecnología de desarrollo. Además tiene los siguientes inconvenientes: subjetividad relativa a la extracción de las funcionalidades del software, necesidad de documentos de especificación detallados y completos. A pesar de estos fue utilizado durante mucho tiempo ya que provee un proceso de medida normalizado, lo cual permite establecer estimaciones objetivas de calidad, costes y tiempos.

Los **PC** es otras de las medidas relacionadas con la funcionalidad del producto, fue propuesto por Caper Jones [Jones, 1987] como una alternativa que permitiera obtener puntos de función en software científico y de ingeniería. Para evitar confusiones con los PF, Jones lo denominó puntos de característica. Ha sido usado con mucho éxito en software del tipo CAD (del inglés Computer Aided Design), sistemas embebidos y sistemas en tiempo real.

Para calcular los puntos de características, nuevamente se cuentan y ponderan los valores del ámbito de información, como se mismo se hace en los PF. Además, las métricas de punto de característica tienen en cuenta otra característica del software, los algoritmos. La medida del punto de característica da cabida a otros tipos de aplicaciones. Como las aplicaciones de software de tiempo real, de control de procesos y de sistemas que tienden a tener una complejidad algorítmica alta y por tanto son fácilmente tratables por puntos de características.

Los **PO** al igual las medidas antes mencionadas, está igualmente relacionada con la funcionalidad vista por el usuario, el número de puntos de objeto en un programa es una estimación de peso en función del número de pantallas generadas, informes producidos y, en general, cualquier módulo generado por sistemas automáticos de desarrollo (RAD) que haya sido integrado en la aplicación. Por ejemplo: El número de pantallas independientes que se despliegan se le asignan valores desde 1 a 3 según su complejidad y el número de informes que se producen se le asignan valores de 2, 5 y 8 teniendo en cuenta al igual que la anterior la complejidad. [IFPUG, 2000].

Tienen como ventajas: Por una parte, es útil en aquellas aplicaciones que puede desarrollarse con un alto componente de reutilización de software, es decir, como si se tratara de tomar componentes ya desarrollados, ensamblarlos y adaptarlos hasta conformar la nueva aplicación. Es lo que se ha dado en llamar Composición o Ensamblaje de Aplicaciones. Es útil también en aplicaciones desarrolladas por herramientas RAD (Rapid Application Development), las cuales generan automáticamente, con el mínimo esfuerzo, todo tipo de componentes software para el sistema que se está desarrollando. Es un factor de peso que se le da a las funcionalidades teniendo en cuenta la complejidad.

A pesar de eso, se manifiestan el siguiente inconveniente: subjetividad a la hora de evaluar la complejidad. Son muy pocos usados por la comunidad debido a que se hace difícil. Es útil en aquellas aplicaciones que puede desarrollarse con un alto componente de reutilización de software

Los **PCU** es otra de las métricas relacionadas con las funcionalidades del producto, es independiente de la tecnología, del lenguaje de programación o plataforma de hardware a utilizar; el número de casos de uso de un sistema permanecerá constante. Lo único variable es la cantidad de esfuerzo para desarrollar un conjunto de datos de CU. [RIBU, 2001]

Los Caso de Uso resultan ser muy efectivos para la estimación temprana del proceso si se sigue un proceso como el RUP, que está dirigido por ellos. Los primeros a desarrollar son los que definen la mayor parte de la arquitectura del software y los que a su vez ayudan a mitigar los riesgos más significativos. En conclusión si la unidad de trabajo es el caso de uso, es más natural contabilizar casos de uso en lugar de entradas, salidas, consultas y datos. A pesar de esto tiene como inconveniente principal, que un caso de uso no tiene un tamaño determinado que solo puede resolverse si se tiene en cuenta el número de transacciones o escenarios del mismo para determinar su peso.

Tienen como ventajas: Constituye una excelente medida cuando se trabaja en un proceso como RUP (dirigido por casos de usos.). Utilizan un proceso de medida normalizado, lo cual permite establecer estimaciones objetivas. Y desventajas: Como inconveniente principal, tiene que los caso de uso no cuentan con un tamaño determinado ya que no se tiene en cuenta el número de transacciones o escenarios del mismo, para asignar su peso.

Los **Escenarios** no son especificaciones ni requerimientos, son descripciones auxiliares para el proceso de definición de requerimientos. Los escenarios representan una fuente de conocimientos donde pueden ser identificados los requerimientos y en la que pueden basarse las especificaciones [Leite, Rossi, 1997]. Son independientes de un método o tecnología de desarrollo de software particular; el uso de escenarios evita abstracciones orientadas a una solución. [BERTOLAMI, 2007]

Los escenarios representan descripciones parciales del comportamiento del sistema en un momento específico de la aplicación, se representan en lenguaje natural. Evolucionan durante el proceso de desarrollo del software y se representan de manera estructurada [LEITE, ROSSI, 1997].

Tiene la ventaja de utilizar el lenguaje natural para las descripciones, lo que favorece la comunicación y validación con el usuario. Los escenarios permiten entender la aplicación y su funcionalidad: cada escenario describe una situación específica de la aplicación centrando la atención en su comportamiento. La ventaja decisiva del enfoque desarrollado consiste en la disponibilidad de una estimación del tamaño del producto a construir en una etapa muy temprana del proyecto de desarrollo.

Con relación al tema que se trata en este trabajo (la estimación) los escenarios normalizados podrían ser una medida de software, a considerar por los autores, de gran ayuda para la predicción del tiempo

de duración del proceso de desarrollo de software de realidad virtual ya que estos presentan una gran similitud con los guiones del mundo virtual ya antes mencionados.

Tiene como ventaja: que utilizan el lenguaje natural para las descripciones, lo que favorece la comunicación y validación con el usuario. Permiten entender la aplicación y su funcionalidad. La disponibilidad de una estimación del tamaño del producto a construir en una etapa muy temprana del proyecto de desarrollo.

### 3.2 Evolución de las técnicas de Estimación de Software

A lo largo de la historia de la Estimación del Proceso de Desarrollo de Software han surgidos multitud de métodos de estimación, estos han sido agrupados en diferentes clasificaciones, según la opinión de los especialista del tema, la primera clasificación se debe a Barry Boehm , en la que distribuye a los métodos de estimación de coste (refiriéndose no solo a valor monetario sino a costo en RRHH, tiempo o cualquier otro recurso) de software en: Modelos Algorítmicos, Juicio de Expertos, Analogía, Ley Parkinson, Precio para Ganar, de Arriba hacia Abajo y de Abajo hacia Arriba. [PÉREZ, 2002]

Recientemente, Boehm ofrece una mueva clasificación, actualización de la realizada en 1981, en la cual aparecen nuevas clasificaciones, las agrupa en: Técnicas basadas en Modelos, Técnicas basadas en Expertos, Orientados al Aprendizaje, Modelos Dinámicos, Modelos basados en Técnicas de Regresión, y Modelos Compuestos Bayesianos. Dentro de los métodos orientados al aprendizaje, Boehm menciona las redes neuronales y el estudio de casos (originados a partir de los modelos por analogía). [PÉREZ, 2002]

A lo largo de las revisiones bibliográficas, desarrollada por los autores, se han encontrado con que no existe una clasificación específica para agrupar los métodos de Estimación del Tiempo de Duración del Proceso de Desarrollo Software, debido a que fundamentalmente a que la estimación del tiempo forma parte de de un proceso mayor conocido como Proceso de Estimación de Software. Los autores atendiendo a la analogía y considerando las diferentes clasificaciones que han ido evolucionando a lo lardo del tiempo, agrupan en cuatro clases, los métodos de estimación:

- Modelos Matemáticos Paramétricos
- Modelos Matemáticos No Paramétricos.

- Modelos Basados en Experiencia.

En los Modelos Matemáticos Paramétricos los autores agrupan todos aquellos modelos que utilicen ecuaciones matemáticas para realizar las estimaciones. No se introduce la diferencia de si han sido generadas por expertos porque se supone que la intervención de expertos es imprescindible en el desarrollo de la ecuación como las que hace DeMarco.

Entre estos métodos se encuentran:

- COCOMO II
- La técnica de análisis de punto de fusión.
- Modelo de análisis de punto de características.
- Modelo de análisis de punto de objetos.
- Modelo de análisis de punto de casos de usos.
- Métodos basados en escenarios.

Estos métodos en toda su formulación utilizan algunas estimaciones iníciales que son difíciles de establecer y frecuentemente inexactas, requieren de datos históricos y heredan las complicaciones de la aplicación de las métricas que utilizan como base, agregándole a esto que la fórmula puede no justarse al proyecto, tornándose complicada para aprender. Posteriormente se procederá a analizar el Modelo COCOMO, las técnicas de análisis de la funcionalidad donde se agruparan los PF, PC y los PO por la similitud de procedimientos se analizara el Modelo de PCU y el de Escenarios por la cual se inclinan el autor.

### 3.2.1 Modelos Matemáticos Paramétricos

#### 3.2.1.1 COCOMO II

Modelo Constructivo de Costo (proviene de las siglas en ingles de Constructive Cost Model) es un modelo algorítmico de estimación creado en los años 80 por Barry Boehm, en sus inicios fue concebido para la estimación de proyectos desarrollados en cascada; desde el año 1995, después de algunas variaciones, es conocido con el nombre de como COCOMO II, este último, provee la estimación de procesos de desarrollo incremental.

Este modelo [BOHEM, 2000] puede ser usado para:

- Establecer presupuestos y calendarios de proyectos como base para la planificación y el control.
- Decidir o negociar entre los costos del software, calendario, funcionalidad, rendimiento o factores de calidad.
- Decidir qué partes del sistema software se desarrollarán, re-utilizarán, sub-contratarán o comprarán.
- Tomar decisiones sobre el software existente: qué partes se van a modificar, dejar fuera, etc.
- Fijar estrategias de inversión combinadas para mejorar la capacidad de la organización, reutilización, herramientas, madurez de procesos, subcontratación, etc.
- Decidir como implementar una estrategia de mejora del proceso.

En el libro de Ingeniería de Software de Ian Sommerville el autor señala que “eligió presentar el modelo de COCOMO en su libro porque:

- Está bien documentado, es de dominio público y lo apoyan el dominio público y las herramientas.
- Se ha utilizado y evaluado muy ampliamente.
- Tienen una gran tradición desde su primera versión en 1981, pasando por su refinamiento para el desarrollo de software en ADA en 1989, hasta su versión más reciente, publicada en 1995.”

Estas consideraciones que hace [SOMMERVILLE, 2002] se ha de tener en cuenta que aunque sus textos tienen una amplia repercusión en el mundo del desarrollo de software debido a su calidad ya probada los casos de estudio descritos en ellos corresponde a Sistemas de Gestión completamente clásicos. Por lo que esta afirmación que él hace en relación a la aplicación de COCOMO II no se comporta asi en otros dominios de aplicaciones de software.

La estimación de proyectos es una actividad que se debe realizar desde las primeras etapas en que se plantea la idea del proyecto informático de construcción de software, sin embargo, la estimación podría no ser adecuada al existir una gran incertidumbre sobre muchos aspectos del proyecto y del producto

a construir. Considerando lo anterior, COCOMO II, presenta dos grandes modelos de acuerdo al mercado al cual está dirigido:

- Composición de Aplicaciones: incluye el esfuerzo de construcción de prototipos para resolver potenciales problemas de gran riesgo como interfaces de usuario, interacción de software y sistemas, eficiencia o madurez de la tecnología. La mejor estimación se realiza mediante puntos de objeto como en el caso de modelo composición de aplicaciones.

- Diseño inicial.- La estimación se puede realizar a partir de un catálogo de requisitos y con un diseño inicial (idea inicial de alternativas de arquitectura y conceptos de operación). La estimación se basa en el cálculo de puntos de función que luego son convertidos a líneas de código fuente y un conjunto reducido de factores que representan los ajustes para el cálculo del costo.

- Post-arquitectónico.- La estimación se realiza luego de completar el diseño de la arquitectura del sistema utilizando los puntos de función que son convertidos a líneas de código fuente y un conjunto amplio de factores que ajustan los costos de acuerdo al producto, el equipamiento, el personal y el proyecto mismo.

Las ventajas que presenta COCOMO son objetivó y repetible y la consideración de múltiples factores. Pero tiene como principales inconvenientes: calibración difícil, subjetividad de determinados factores y no aplicable a todos los dominios

#### 3.2.1.2 Técnicas de análisis de Punto de Función

Esta técnica surge a la par, de la los Puntos de función, consiste en sumarle factores de pesos a las funcionalidades a implementar en la aplicación. Requiere de la existencia de una base de datos históricos para realizar las consideraciones de asignación de los factores de peso y a si determinar los costos de esfuerzo y tiempo para realizar el proyecto.

Tienen a consideración de la comunidad de desarrolladores muy poco peso en la utilización en aplicaciones fuera del rango de sistemas de gestión.

#### 3.2.1.3 Técnica de Puntos de Casos de Uso

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner, y posteriormente refinado por muchos otros autores. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.[Peralta, 2004].

Este método de estimación y cálculo de tamaño del software, está basado en las cuentas hechas sobre los casos de uso para un sistema de software. El método exige la existencia de un modelo de casos de uso, por lo que la labor deberá ser hecha cuando exista algún entendimiento del dominio del problema o cuando se esté realizando las labores de arquitectura y dimensionamiento del tamaño del sistema.

### 3.3.2 Modelos Basados en la Experiencia

Los Modelos Basados en la Experiencia se basan en la comparación de los resultados reales de diferentes proyectos, siempre entregados a cliente, y en los que se ha aplicado la experiencia de los expertos como fuente de conocimiento esencial.

Entre estos métodos se encuentran

- Método analógico.
- Juicio de Expertos
- Método de desagregación estructural.
- Técnica Delphi

#### 3.3.2.1 Juicio de Expertos

Es una técnica de estimación por la cual se usa las experiencias para guiar las estimaciones de tiempo y de costos. Cada tarea es definida en términos de los tipos de programas más que como el resultado de la tarea. Se le asignan tiempo a cada entregable, agregando los tiempos de análisis y diseño.

Las ventajas son que se basa en la experiencia para las estimaciones, ajusta las estimaciones al personal asignado y las estimaciones son hechas rápidamente y eficientemente.

Las desventajas son que las estimaciones no son mejores que las experiencias de los estimadores, pueden ser sesgadas, difíciles de racionalizar y no son objetivamente repetibles.

#### 3.3.2.2 Técnica Delphi

Las Técnicas Delphi fueron desarrolladas en la corporación Rand en el año de 1948, con el fin de obtener el consenso de un grupo de expertos sin contar con los efectos negativos de las reuniones de grupos. La técnica a tenido una amplia aceptación en varias comunidades científicas, es aplicada en muchas variantes.

#### 3.3.2.3 La estructura de desagregación del proyecto

La estructura de desagregación del proyecto EDP (traducción en español de Work Breakdown Structure) se define como un “modelo sistémico de la composición o alcance del proyecto, considerado en todo sus aspectos, incluido los de su entorno”. Según describe Heredia (1995), estos aspectos serian subsistemas del proyecto, tales como los correspondientes a “actividades” o “tareas” o “paquetes” de contratos que los componen y configuran en cada una de las fases que componen el ciclo de vida. Existen muchas variantes para llevar a cabo este modelo, sin embargo se puede identificar como idea común, la modelación del proceso para la realización del proyecto. [FLEMING, 2005]

Puede de ser de gran utilidad debido a que está pensada bajo la filosofía “divides y vencerás” su calidad la precede en la aplicación de disimiles situaciones complejas.

#### 3.3.2.4 Estimación por Analogías

Esta técnica es aplicable cuando hay otros proyectos en el mismo dominio de aplicación que se hayan completado. Se estima el tiempo de un nuevo proyecto por analogía con estos proyectos completados. Depende mucho del lo proceso de seguimiento que se le haya aplicado a proyectos anteriores.

Es similar a la aplicación de la experiencia. En una estimación por analogía, un proyecto completado recientemente es seleccionado para actuar como línea base en las estimaciones de costos de un proyecto propuesto. Los costos (tiempoo esfuerzo) son determinados en base a la igualdad o la desigualdad de tareas y programas con respecto al proyecto que sirve de línea base.

La principal virtud de la estimación por analogía es que está basada en la experiencia real de un proyecto. Esta experiencia puede ser estudiada para determinar las diferencias específicas con un proyecto nuevo y el impacto de los cambios en los costos.

Esta técnica requiere de bases de datos detallados, e implica el conocimiento de todos los factores de recursos.

### 3.3.3 Modelos Matemáticos No Paramétricos

Estos métodos se basan en el uso de datos recogidos en proyectos anteriores en los que se realizaron estimaciones y usando diferentes técnicas de extracción de conocimiento. Pretenden realizar las estimaciones que se están realizando con los modelos paramétricos con estas metodologías de manera más eficiente, eficaz y , si fuera posible, con mayor precisión.

Entre las de extracción del conocimiento e encuentran la técnicas de inteligencia artificial que se basan en el uso de datos recogidos en proyectos anteriores en los que se realizaron estimaciones y usando diferentes técnicas de extracción de conocimiento, entre las que destacan las técnicas las Redes Neuronales, el Razonamiento basado en casos, Árboles de regresión, Lógica difusa, y la Programación genética. [ROMERO, 2000]

**Redes Neuronales:** Requieren un conocimiento profundo del tema. Funcionan como cajas negras. No proporcionan información sobre cómo llegan a los resultados. Esta es la más usada como alternativa a los métodos para métricos. Su fundamento es la variación de una serie de neuronas con un valor inicial, denominada semilla previa. Esta variación se efectúa por retro propagación en base a la información de entrenamiento que se le suministra y que se valida posteriormente

**Árboles de Regresión**: Los árboles de regresión trabajan tomando un conjunto de datos conocidos y obteniendo reglas de clasificación por aprendizaje. Estas permiten predecir los valores de salida en virtud de los intervalos de dominio de las reglas de clasificación obtenidas. Con el fin de crear estos árboles los algoritmos al uso necesitan una serie de atributos que clasifiquen suficientemente los datos y construyan de forma iterativa un árbol por división dependiendo de los citados atributos.

**Programación Genética**: Generaciones en las que se selecciona el mejor adaptado. El proceso básico de la programación genética es:

1. Generación aleatoria de una familia de soluciones (“a family of chromosomes”). Creación de una nueva familia de soluciones partiendo de la previa y una vez aplicadas a los operadores para afinar los “cromosomas” y acercarlos a la solución.
2. Se repite el paso b) hasta un afinamientos que obtenga la mejor solución o cuando se hayan producido un número de iteraciones anteriormente especificado.

**Razonamiento Basado en Casos**: Es una metodología que usa la información almacenada sobre datos de características de proyectos históricos y del esfuerzo dedicado en su desarrollo. Esta información se usa cruzándola con las nuevas observaciones para obtener los valores a priori del esfuerzo y demás datos de interés de la nueva aplicación.

**Lógica difusa:** No existen muchos trabajos publicados al respecto, si bien muchos autores señalan esta técnica como posible solución al problema. Hay diferentes tipos de sistemas en lógica difusa y el desarrollo del modelo se realiza en base a las diferentes elecciones disponibles.

Se ha comprobado que estas técnicas son muy efectivas para la estimación de software siempre que se posea una base de datos con un tamaño realmente considerables debido a que son técnicas de inteligencia artificial necesitan de un extenso caudal de conocimiento ya que carecen de precepción humana, como podrían tener los expertos en un tema.

### 3.4 Presentación de Otras Soluciones de Procesos de Estimación de Software

En este apartado se presentaran algunas soluciones expuestas por varios especialistas para llevar a cabo una Estimación en el Proceso de Software que servirán de preámbulo para l desarrollo de la Propuesta de Guía Metodológica para la Estimación del Tiempo de Duración de los Proyectos de Realidad Virtual de la UCI.

#### 3.4.1 Proceso de estimación de Bailey y Basili

Bailey y Basili propusieron el “meta modelo de Bailey-Basili” como un proceso para estimar el esfuerzo. Este proceso tiene tres pasos principales que se describen a continuación: [CAPUZ, ELISEO, 1999]

- Calibrar el modelo de esfuerzo: El modelo de esfuerzo se calibra usando una regresión por mínimos cuadrados, sobre un conjunto de datos locales para calcular los coeficientes del modelo.

- Estimar los límites superior o inferior del modelo de predicción: Se trata de calcular el error estándar de estimación. Se puede utilizar para calcular los límites superior e inferior del intervalo de confianza construido para el modelo, asumiendo una distribución normal.

- Ajustar las predicciones del modelo para tener en cuenta la significancia de los atributos conductores del coste: El efecto de los conductores del coste sobre el esfuerzo se estima calculando un factor de ajuste de esfuerzo y aplicándolo a la estimación de esfuerzo inicial. El factor de ajuste para un punto de la regresión original es el número mediante el que el valor efectivo ha de ser multiplicado para obtener el valor predicho. Un modelo de ajuste de esfuerzo se calibra utilizando regresión lineal múltiple para determinar los coeficientes conductores del coste, para tres atributos conductores del coste.

### 3.4.2 Proceso de Boehm.

[Boehm, 1981] propuso un proceso de siete pasos para estimar el coste del software:

- Establecer objetivos: El primer paso es determinar nuestros objetivos en la estimación. Para ello hemos de evitar gastar tiempo en recopilar información y realizar estimaciones que no sean relevantes para las decisiones que tenemos que tomar.

- Planificar recursos y datos requeridos: En el paso siguiente, debemos planificar nuestra actividad de estimación, a fin de evitar la tentación de preparar una estimación demasiado rápida.

- Fijar los requerimientos software: En este tercer paso determinaremos si las especificaciones en las que están basadas nuestra estimación son económicamente viables. Necesitamos saber que estamos intentando construir a fin de estimarlo todo con precisión.

- Detallar tanto como sea posible: El cuarto paso nos indica que hacer una estimación con mucho detalle mejora su precisión.

- Utilizar varias técnicas y fuentes independientes: Boehm clasifica las técnicas de estimación como modelos algorítmicos, opiniones de los expertos, analogía, Parkinson, Precio para Ganar, de abajo-arriba y de arriba-abajo. Este paso nos indica que hemos de usar más de una técnica para realizar una estimación del coste. El uso de una combinación de técnicas nos permite evitar los puntos débiles de cada técnica, así como tomar lo mejor de cada una.

- Comparar e iterar estimaciones: Una vez realizada nuestra estimación utilizando dos o más técnicas, debemos comparar los resultados. Si son inconsistentes, nuestro siguiente paso es investigar las diferencias. El objetivo es converger en una estimación lo más realista posible, más que establecer un compromiso arbitrario entre las estimaciones iníciales.

- Continuación: El paso final en el proceso de estimación es continúa a la estimación realizada en paso anterior. Debemos usar los resultados de la comparación entre la estimación y el valor real para mejorar nuestra técnica de estimación y nuestros modelos. Deberíamos re-estimar cuando el alcance del proyecto cambia.

### 3.4.3 Proceso de DeMarco.

[DeMarco, 1982] propone un proceso para desarrollar y aplicar modelos de coste basados en datos recolectados localmente. El defendía el uso de modelos de coste de factores simples derivados de la regresión lineal. Un modelo de coste de factores simples predice el esfuerzo para todo o parte del proyecto basándose en una variable independiente. El esfuerzo estimado obtenido desde un modelo de coste de factores simples se ha de ajustar aplicando un factor de corrección. Los factores de corrección, se derivan en el mismo modo propuesto por Bailey y Basili.

El modelo de DeMarco soporta estimación abajo-arriba y de arriba-abajo. El sugiere dividir el proyecto en componentes de coste, tales como: esfuerzo de diseño, esfuerzo de implementación y esfuerzo de integración. Cada componente de coste es estimado mediante uno o más modelos basados en la medida, los cuales están disponibles en el momento de realizar la estimación. DeMarco también defendía el uso de modelos sensibles al tiempo, los cuales relacionan el esfuerzo total del proyecto con la duración. El sugiere que se use el modelo COCOMO, para este propósito. A continuación se indican las actividades propuestas por DeMarco para utilizar este modelo. [PERALTA, 2004]

- Seleccionar el coste del componente a modelar: El primer paso en el proceso implica seleccionar los componentes sobre los que se va a realizar una estimación de coste. El coste de los

componentes está basado en una descomposición del sistema de desarrollo en actividades como especificación, diseño, implementación y test.

- Seleccionar las medidas desde las cuales predecir los componentes y el coste total: Una vez que los componentes han sido seleccionados, el siguiente paso es seleccionar las medidas desde la cual el esfuerzo para completar esos componentes pueda ser predicho. Por ejemplo, las medidas de código pueden ser utilizadas para predecir los test por un partido común.

- Desarrollar los modelos de coste basados en un simple valor, basándose en los datos existentes: Una vez que se han seleccionado las métricas, los modelos de regresión, mediante los que se predice el esfuerzo, deben ser desarrollados.

- Estimar el coste y el error estándar de la estimación para cada componente del modelo: Cuando el valor de las métricas utilizadas en el modelo de coste basado en un factor simple puede ser estimado o medido, el modelo se utiliza para predecir el esfuerzo de los componentes. Si se utiliza una estimación del valor de las métricas de entrada, el esfuerzo debería ser re-estimado cuando el valor real de estas métricas ha sido calculado.

- Ajustar las estimaciones basadas en diferencias entre el proyecto actual y proyectos pasados: DeMarco indica que la estimación del esfuerzo debe ser ajustada mediante una evaluación subjetiva de las diferencias entre el proyecto en curso y los proyectos pasados.

- Calcular el coste total a partir de los costes de los componentes: Una vez estimado el coste de todos los componentes, se calcula el coste total como suma de estos.

- Calcular el error de estimación global: El error en la estimación total se calcula sumando el error estándar de la estimación de cada componente aislado.

- Utilizar un modelo de coste sensible al tiempo para restringir la estimación total del esfuerzo del proyecto: El paso final en el proceso de DeMarco utiliza la estimación del esfuerzo como entrada a un modelo sensible al tiempo, tal como COCOMO lo establece, para estimar la duración del proyecto.

### 3.4.4 Proceso de Heemstra.

Heemstra describe un proceso general de estimación de costes. Su proceso asume el uso de modelos de esfuerzo dependientes del tamaño, y el uso de atributos conductores del coste, unidos a estos modelos para realizar ajustes de productividad. Este proceso se divide en siete pasos: [CAPUZ, ELISEO, 1999]

- Crear una base de datos de proyectos finalizados: El paso inicial en el proceso de estimación es la recolección de datos locales para validar y calibrar un modelo.

- Validar el modelo de estimación: El entorno en el que ha sido desarrollado el modelo del coste y los proyectos finalizados en los que está basado, pueden diferir del entorno en el que el modelo es usado. Heemstra advierte, por consiguiente, que antes de utilizar un modelo precalibrado por primera vez en una organización, este debería ser validado, evaluando su precisión sobre datos de proyectos ya finalizados en la organización.

- Calibrar el modelo de estimación: Si en el paso previo se indica que el modelo no es válido para el entorno en el cual va a ser utilizado, el modelo necesita ser re-calibrado usando datos de proyectos finalizados en el entorno actual.

- Estimar el tamaño: En este paso, se calcula el tamaño del software partiendo de las características propias del software a desarrollar.

- Estimar el esfuerzo y la productividad: Una vez realizada la estimación del tamaño, el siguiente paso convierte la estimación del tamaño en estimación de esfuerzo. Los atributos conductores del coste con influencia en el esfuerzo del desarrollo del software son evaluados y utilizados para ajustar la estimación del esfuerzo.

- Distribuir el esfuerzo a lo largo de las fases de realización del proyecto: En este paso del esfuerzo total y la duración del proyecto son distribuidas a lo largo de las fases de desarrollo del software.

- Analizar la sensibilidad de la estimación y los riesgos asociados: En este paso se evalúa la sensibilidad del coste estimado para valores de los atributos conductores del software, y se determinan los riesgos asociado con la estimación de costes del proyecto

### 3.4.5 Proceso de Arifoglu.

El proceso de estimación de Arifoglu consta de cuatro pasos: [CAPUZ, ELISEO, 1999]

- Estimación del tamaño: Este paso es equivalente a la estimación de tamaño del proceso de Heemstra.

- Estimación del esfuerzo y duración: Este paso convierte la estimación del tamaño en estimación del esfuerzo y estimación de la duración. Arifoglu sugiere el uso del modelo COCOMO en este paso.

- Distribución del esfuerzo y la duración durante el ciclo de vida: Una vez que el esfuerzo total y la duración del proyecto ha sido estimado, han de ser distribuidos a lo largo del ciclo de vida.

- Reflejar el esfuerzo en el calendario: El paso final del proceso de Arifoglu es convertir el número de días de trabajo requeridos para completar el proyecto en número de días transcurridos en el calendario. Arifoglu es partidario del uso del modelo de Esterling, para realizar esta conversión. Esterling propone un modelo para estimar el porcentaje de día gastado trabajando directamente en una tarea Este paso realiza la predicción de la duración del proyecto usando el modelo COCOMO u otro similar. La duración predicha por COCOMO es ya un tiempo estimado de calendario (incluyendo fines de semana, y tiempo medio de vacaciones y bajas por enfermedad).

## 3.5 Fundamentos para la Solución que se propondrá.

Para hacer la propuesta de la solución del problema planteado en el trabajo, los autores tendrán en cuenta cinco elementos que no pueden faltar en el proceso de estimación del tiempo, a continuación se describirán los siguientes: caracterizar el proyecto, seleccionar las métricas y métodos, planificar la magnitud y recolección de datos, realizar las predicciones y evaluar su precisión, analizar las medidas y métodos y almacenar la experiencia**.**

Para que sirva de validación de la guía que se expondrá en el capitulo posterior, se dará cita a los autores que reflejan estos pasos en sus respectivos procesos de estimación.

### 3.5.1 Caracterizar el proyecto.

El primer paso del proceso de predicción en proceso de Boehm es caracterizar el proyecto y el entorno, para el que se realiza la predicción. Los otros modelos de estimación del coste descritos más arriba no incluyen este pasó de forma explícita. No obstante, esta caracterización está implícita en cualquier proceso que contenga una evaluación de la influencia de los atributos conductores del software en la estimación del esfuerzo. Una posible ventaja de acarrear el paso de la caracterización explícitamente, es que las restricciones y objetivos establecidos para el proyecto, serán claramente definidos antes que la estimación sea realizada. Esto permite que los valores de restricción sean utilizados como entradas a los pronósticos.

### 3.5.2 Seleccionar las métricas y métodos.

Los procesos de [BOEHM, 1981] y de [DEMARCO, 1982] incluyen pasos que implican la selección de métricas a predecir, cumpliendo de este modo el segundo paso del proceso de predicción. En contraste los procesos de Bailey y Basili, Hemstra y Arifoglu asumen que una métrica sencilla, tal como las líneas de código o los puntos de función, se usara para generar una predicción completa del esfuerzo. No obstante en un escenario realista de desarrollo de sistemas, son necesarias más de una única estimación del esfuerzo y la duración. La inclusión de un paso que seleccione las métricas a predecir permite que el proceso sea utilizado de manera más general, que el proceso que asume que las métricas son las mismas para todos los proyectos.

### 3.5.3 Planificar la magnitud y recolección de datos.

[BOEHM, 1981] incluye, de manera análoga al proceso de predicción, un paso para planificar los datos y recursos requeridos en la estimación. La actividad de recolección de datos para realizar la calibración del modelo de estimación, se realiza explícitamente en el proceso de Heemstra. Los proceso de Bailey y Basili y de [DEMARCO, 1992] incluyen también recolección de datos, al igual que los procesos anteriores incluyen también pasos para estima y calibrar los modelos. Una recolección de datos fiables requiere un esfuerzo considerable para el personal de soporte, por ello es importante planificar la cantidad y tipos de datos a obtener en cada paso del proceso de estimación.

### 3.5.4 Realizar las predicciones y evaluar su precisión.

El cuarto paso en el proceso de predicción implica evaluar la precisión de las predicciones, para proporcionar realimentación al proceso de desarrollo. La realimentación es necesaria para mejorar los ajustes de la estimación, así como para la planificación y el control. La realimentación es parte del pro-

ceso de [BOEHM, 1981], [DEMARCO, 1982] y [Humphrey, 1995], pero no existe en el resto de los procesos.

### 3.5.5 Almacenar la experiencia.

El paso final del proceso de predicción almacena lo que se ha aprendido, de manera que estos conocimientos puedan ser reutilizados cuando se hagan predicciones futuras. Almacenar lo que se ha aprendido con la estimación actual, no es explícitamente una parte del proceso descrito. Al incluir esta actividad en el proceso, lo que se hace es contribuir a facilitar la tarea en procesos futuros.

# Capítulo 4 Procedimientos para la estimación del Tiempo.

## 4.1 Objetivo de la Guía de Estimación de Tiempo de Duración de Proyectos de Realidad Virtual.

El propósito final de esta guía metodológica es obtener a través de una serie de pasos, un estimado del tiempo de duración de los proyectos de realidad virtual.

La guía propone avanzar a través de una serie de pasos lógicos identificas en el estudio de procedimientos de estimación propuestos por prestigiosos autores de la talla de Boehm, DeMarco, Humphrey Bailey, Basili, Hemstra y Arifoglu que han dedicado la mayor parte de su vida al estudio de procesos de medición y estimación de software. A los cuales la comunidad internacional reconocen como expertos del en tema.

## 4.2 Momentos previos del proceso de Estimación.

### 4.2.1 Caracterización del Proyecto.

La caracterización del proyecto o levantamiento del ámbito del proyecto es la primera actividad que se realiza en todo Proceso de Desarrollo. Las descripciones del Ámbito del proyecto son las bases para obtener las funcionalidades del mismo. A partir de este momento se iniciara el proceso de estimación del tiempo de duración del proyecto.

Las descripciones de los distintivos propios del proyecto, desde el punto de vista del proceso, se han de obtener de forma que se tenga bien claro el entorno de desarrollo y las particularidades del proyecto como tal. Esta parte no presupone mucho trabajo ya que el dominio de las aplicaciones está bien definido (aplicaciones de realidad virtual) y el proceso (Anexo 1) a utilizar es el mismo para todo el dominio.

En procesos de desarrollo como RUP esta actividad está bien definida por lo que el buen cumplimiento de la misma será definitorio para determinar la calidad, lo mismo del proceso de desarrollo como el de estimación.

### 4.2.2 Determinación del momento de la estimación del tiempo.

Para establecer el momento en que se va a realizar la estimación, ha de considerarse la técnica de estructura de desagregación del proyecto, debido a que se debe tener en cuenta la disponibilidad de

los datos necesarios, que será la entrada del modelo de estimación a aplicar y el alcance de la estimación que se desea.

Teniendo en cuenta lo ante plantado los autores proponen los siguientes instantes, dentro del proceso propuesto en el Anexo 1, de estimación: El primer instante que proponen los autores para realizar la estimación, en cuando se posea la caracterización del proyecto, pues se ha de hacer la estimación general para llevar una idea al cliente de cómo se comportara el proceso de desarrollo. Esta actividad está dentro de la fase de inicio definida en procesos como RUP y XP.

El segundo es al realizar el Plan del Proyecto. En este momento se cuenta con el Modelo del Negocio y con las especificaciones funcionales de alto nivel ya refinadas y aprobadas por el cliente y la dirección del proyecto. Este plan se hará como parte de la planificación global del proyecto, es cuando se asignarán las tareas a los miembros del equipo de trabajo.

Considerando que la Estimación se realizara dentro de un proceso de desarrollo, iterativo es válido tener en cuenta que por cada iteración se ha de considerar, las modificaciones introducidas por el usuario que modifique los valores de las métricas a utilizar.

### 4.2.3 Selección de la Métrica de Software según el momento de la estimación.

Este paso, está muy relacionado con el anterior, debido a que como se había anticipado, depende de la cantidad de información disponible del proyecto.

Las descripciones de los Escenarios se ajustan a la estimación temprana del proyecto debido a que son la base para la obtener los Casos de Uso del negocio. Esta métrica es idónea para ser utilizada en el primer instante de la estimación.

Los Casos de Uso pueden ser otras de las métricas a utilizar en el proceso de estimación, debido a que los casos de uso dirigen el proceso de desarrollo siendo esto último una de las características fundamentales del Proceso de Desarrollo descrito en el Anexo 1. Por lo que los autores lo proponen para el segundo y tercer momento de la estimación.

### 4.2.4 Selección de la Técnica de Estimación.

Teniendo en cuenta que el uso de una combinación de técnicas, permitirá evitar los puntos débiles y así tomar lo mejor de cada una de ellas. Los autores proponen utilizar técnicas de estimación relacio-

nadas con la experiencia. En este grupo de técnicas se encuentran la estructura de desagregación de proyectos, juicio de experto y el método comparativo por analogía.

### 4.3 Proceso de estimación del tiempo de duración.

#### 4.3.1 Primer momento.

Correspondiente al inicio del proceso de estimación se procederá a identificar las actividades, del proceso de desarrollo. El encargado de la estimación, se dispondrá a realizar un listado de las actividades, desagregándolas en niveles escalonados. Como producto final debe quedar un árbol jerárquico donde los nodos (las hojas) corresponden a las actividades del proceso. Los nivele del árbol lo define el grado de especificación del proceso.

#### 4.3.2 Segundo momento.

En este instante, se debe proceder a buscar entre los archivos históricos, las descripciones recolectadas, en procesos anteriores, de los escenarios desarrollados o de los casos de usos, según la disponibilidad de la información.

En esta momento se recurre a la cualidades del estimador (persona que hace la estimación) de poder identificar las semejanzas que existan entre el proceso ya realizado y el que está por realizarse.

#### 4.3.3 Tercer momento.

Se identifican el tiempo de duración de las actividades que se realizaron y se asocian ese valor a las actividades que se realizaran. En caso de haber más de una actividad que se asemeje a la vuestra hallar la media del valor del tiempo.

#### 4.3.4 Cuarto momento.

En éste instante se debe disponer de una lista de valores del tiempo de duración, con la cual se procederá a sumar el resultado será nuestro tiempo estimado de duración del proceso de estimación.

**4.4 Retroalimentación de la estimación.**

Este paso, por ser el último no es el menos importante debido a que es donde se sientan las bases para las futuras estimaciones. A través de la recolección de datos correspondiente con el tiempo de duración real asociadas a cada actividad del proceso se validaran y calibraran el modelo estimación.

Se tiene que tener en cuenta que estos datos formaran parte de furas estimaciones por lo que su recolección debe ser rigurosa. La utilización de una base de datos podría ser una manera de coleccionar estos datos históricos.

**Conclusiones**

A manera de conclusión el autor del trabajo de tesis harán una revisión del cumplimiento de los objetivos de trazados en la introducción.

En el Capítulo 1 se ha presento un conjunto de elementos relacionados con el entorno del conocimiento donde se enmarca el proceso de estimación, utilizando un enfoque orientado al proceso de desarrollo. Desde donde se puede intuir un grupo de consideraciones a tener en cuenta para un mejor juicio de los factores componentes de la propuesta que se dio posteriormente en este trabajo.

En el Capítulo 2 se plantea la necesidad del establecer una relación entre los fundamentos teóricos del trabajo y la problemática que se trata. La cual se evidencia con el desenlace de que el Proceso de desarrollo del software requiere de la estimación como una herramienta para controlar y administrar los recursos que se necesitan y que se utilizan antes y durante el proyecto de desarrollo.

Posteriormente en el capítulo 3 se hizo un estudio de las métricas que se utilizan en las técnicas de estimación donde se concluye que una gran parte de los métodos de medición de tamaño funcional, han sido diseñados para medir el tamaño del software sin tener en cuenta las características del mismo. Sin embargo, no siempre este hecho presupone una ventaja, puede darse el caso que la utilización de estos métodos es apropiada un determinado dominio, pero es inapropiada en el resto de los dominios.

Se analizaron procesos de estimación propuestos por reconocidos profesionales que han estudiado el entorno de conocimiento que se trata en esta tesis, de donde se extrae principios que deben tenerse en cuenta en un proceso de estimación.

Por último en el Capítulo 4 se hizo la propuesta de Proceso de estimación del Tiempo de duración de Proyectos de Realidad Virtual para hacer la propuesta se tomo en cuenta las características de los proyectos de este tipo, se propuso establecer como medidas de software los escenarios y los casos de usos así como las técnicas de estimación, basadas en el aprendizaje o sea en la experiencia adquirida durante años de trabajo la aplicación de estas técnicas como el juicio de expertos, la comparación por analogía y la desagregación estructurada de proyectos.

**Recomendaciones.**

De este trabajo se extrae la necesidad de guardar registros históricos de la aplicación de métricas de software como las descripciones de los escenarios y los puntos de casos de usos, por tal motivo se propone, como primera recomendación, a las dirección de los proyectos de realidad virtual crear una base datos donde se almacenen estos datos.

Una segunda recomendación es que atendiendo a las ventajas que se evidencia en la aplicación en la actualidad de los métodos de estimación: Matemáticos No Paramétricos se propone a partir de la re-colección de datos históricos la implementación de unos de los algoritmos de inteligencia artificial mencionados en el capítulo 3.

**Referencias Bibliográficas**

[AJENJO, 2000] Ajenjo, A D. “Dirección y gestión de proyectos: un enfoque práctico.” Edit. Ra-Ma. Madrid, 2000.

[ANTONIO, 2001] Antonio, A, Gestión de Configuración. Chile, SPIN., 2001

[BECK, 2000] Beck, Kent, Extreme Programming Explained, Addison-Wesley The XP Series, 2000.

[BECK, 2002] Beck, Kent, Test-Driven Development By Example, Addison-Wesley, November 2002.

[BOEHM, 1976] Boehm,B.W Software Engineering ,IEEE Transaction on Computers, , num. 12, 1976

[BOEHM, 1981] Boehm, B. Vol. 10 “Software Engineering Economics”, Edit. Prentice-Hall, Prentice-Hall 1981

[BOEHM, 2000] Boehm.B.W, Software Cost Estimation, Prentice Hall PTR, 2000

[BOEHM, CHULANI, 2000] Boehm, B., Chulani, S.”Software development cost estimation approaches” Annals of Software Engineering, 2000.

[BOEHM, 2002] Boehm, Barry Software Risk Management: Overview and Recent Developments USC-COCOMO/SCM Forum #17 Tutorial October 22, 2002

[BERTOLAMI, OLIVEROS, 2006] Bertolami, Mabel y Oliveros Alejandro “Un Procedimiento de Estimación de Puntos Función de Escenarios”, Argentina ,2006.

[CAPUZ, ELISEO, 1999] Capuz, S, Eliseo, G;.” El Proyecto y su Dirección y Gestión. Servicio de Publicaciones” Edit. Universidad Politécnica Valencia. Valencia, 1999.

[CONTE, DUNSMORE, SHEN 1986] Conte, S.D., Dunsmore, H. E., Shen, V.Y., Software Engineering Metrics and Models, Benjamin/ Cumming Co., Inc., Menlo Park, 1986.

[COS, 1999] Cos, M C. Teoría General del Proyecto, Volumen I y II. Edit. Síntesis, Madrid 1999

[CUADRADO, GARRE, SICILIA, 2007] Cuadrado JJ, Garre M Sicilia M.Á. "Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software", Revista Española de Innovación, Calidad e Ingeniería del Software, ISSN: 1885-4486 I, Vol.3, No. 1, 2007

[CHRISTOPHER, 1977] Christopher, A. Pattern Language, Oxford University Press, New York, 1977.

[DEMARCO, 1982] DeMarco, T." Controlling Software Projects, Edit Yourdan Press, Yourdan Press 1982

[DOORN, LEITE, HADAD, KAPLAN, 2000] Doorn, J., Leite, J., Hadad, G., Kaplan, G., A Scenario Construction Process. Edit. Requirements Engineering Journal, 2000.

[FAIRLEY, 1992] Fairley, R.E. "Recent advances in software estimation techniques". International Conference on Software Engineering, New York, 1992.

[FEBLES, 2007]Febles, A. Conferencia Inaugural del Taller: "Perspectivas para una Industria Cubana de Software de Calidad". III Taller de Calidad en las Tecnologías de la Información y las Comunicaciones. Informática 2007 Ciudad de la Habana 2007.

[FEBLES, PIÑERO, FERNÁNDEZ, NAPAL 2006] Febles, A; Piñero, P; Fernández, Y; Napal, I. Diseño de un segundo perfil de calidad de software para graduados de Ingeniería Informática, La Habana, 2006

[FENTON, 1991] Fenton, N.E., Software metrics: a rigorous approach. Edit. Chapman & Hall, Londres, 1991.

[FRANCO, 2006] Franco, J. Á., Entorno Unificado para la Gestión de Configuración de Software. Cuba, Tesis de Maestría en Informática Aplicada, 2006

[FLEMING, 2005] Fleming, Q. Earned Value Project Management, Third Edition, Project Management Institute. 2005

[GRANJA, 2007] Granja, JC. Conferencia "Controles y Métricas Técnicas del Software" UCI, La Habana, 2007.

[HEREDIA, 2003] Heredia, R, "Dirección integrada de proyectos (DIP) Project Management". Ed. E.T.S. de Ingenieros Industriales de la Universidad Politécnica de Madrid. 2003

[HADAD, KAPLAN, OLIVEROS, LEITE, 1996] Hadad, G., Kaplan, G., Oliveros, A., Leite, J., Integración de Escenarios con el Léxico Extendido del Lenguaje en la elicitación de requerimientos. Aplicación a un caso real, Universidad de Belgrano, 1996.

[HERNÁNDEZ, BANDERAS, 2001]. Hernández Y, Banderas. I "Case para la planificación y control de configuración de software". Trabajo de Diploma para optar por el título de Ingeniero Informático. Instituto Superior Politécnico "José Antonio Echeverría". Ciudad de la Habana, 2001.

[ISDALE, 2005] Isdale J. "What Is Virtual Reality?" [Citado en marzo/007], 2005. Disponible en http://vr.isdale.com/WhatIsVR.html.

[JACOBSON, BOOCH, RUMBAUGH, 2000] Jacobson I, Booch G, Rumbaugh J. El Proceso Unificado de Desarrollo de Software. ISBN: 84-7829-036-2 Editorial Pearson Educación S.A. 2000.

[LOCK, 1994] Lock D. Gestión de Proyectos. Edit. Paraninfo, Madrid 1994.

[LABDELAOU, 2005] Labdelaoui, H. Tesis Doctoral: "Métodos de Estimación de Tamaño Funcional Software Aplicación a Enfoques de desarrollo" 2005.

[MARTIN, 1991] Martin, J., Rapid Application Development, Macmillan Inc., New York, 1991.

[MORENO, 1999] Moreno A. Ma. Control y Gestión de Proyectos Software "Estimación de Proyectos Software." Carpetas de la Carrera de Postgrado en Ingeniería de Software. Edit. Imprenta del Instituto Tecnológico de Buenos Aires. Buenos Aires 1999

[PARRA, GARCÍA, SANTELICES, 2001] Parra J.C., García R., Santelices M., "Introducción Práctica a la Realidad Virtual" Edito. Universitario. Bío-Bío, Concepción, Chile, 2001.

[PASTOR, JAVIER, 2004].Pastor A, Javier J.; Problemas en la Estación Citado el :( Diciembre 2006) Disponible en la web: **http://es.theinquirer.net/2006/09/15/la_estacion_iss_en_problemas_p.html**

[PEREÑA, 1996] Pereña, J B. "Dirección y Gestión de Proyectos." Edit. Díaz de Santos. Madrid, 1996.

[PÉREZ, 2002] Pérez, I. "Métricas para el control de proyectos de software" Trabajo De Diploma para optar por el título de Ingeniero Informático. Instituto Superior Politécnico "José Antonio Echeverría". Ciudad de la Habana, 2002.

[PERALTA, 2004] Peralta M, Reportes Técnicos en Ingeniería de Software. ISSN: 1668-3137, CAPIS-EPG-ITBA,( Citado el 9 de Marzo 2007),2004. Disponible en: http://www.itba.edu.ar/capis/rtis/index.htm.

[PRESSMAN, 2002] Pressman, R. S. Ingeniería de Software "Un enfoque práctico", Edit. Félix Varela, la Habana ,2002.

[PFLEEGER, 2001] Pfleeger S. Software Engineering: Theory and Practice. Edit. Prentice Hall, Prentice Hall 2001

[RIBU, 2001] Ribu, Kirsten, Estimating Object-Oriented Software Projects with Use Cases, Master of Science Thesis, University of Oslo, Department of Informatics, November 2001.

[ROMERO, 2000] Romero, C Técnicas de Programación y control de Proyectos., Pirámide-2000

[ROYCE, 1998] Royce, Walker, Software Project Management: A Unified Framework, Addison-Wesley Object Technology Series, 1998

[RUIZ, 2007]. Ruiz, Alina, Mesa Redonda “Desarrollo y Perspectivas del desarrollo de la Industria de Software en Cuba e Iberoamérica”, Taller de Calidad en las Tecnologías de la Información y las Comunicaciones, La Habana, 2007

[SOMMERVILLE, 2002] Sommerville, I.; Ingeniería de Software, Addison Wesley, 6ta Edicion, 2002

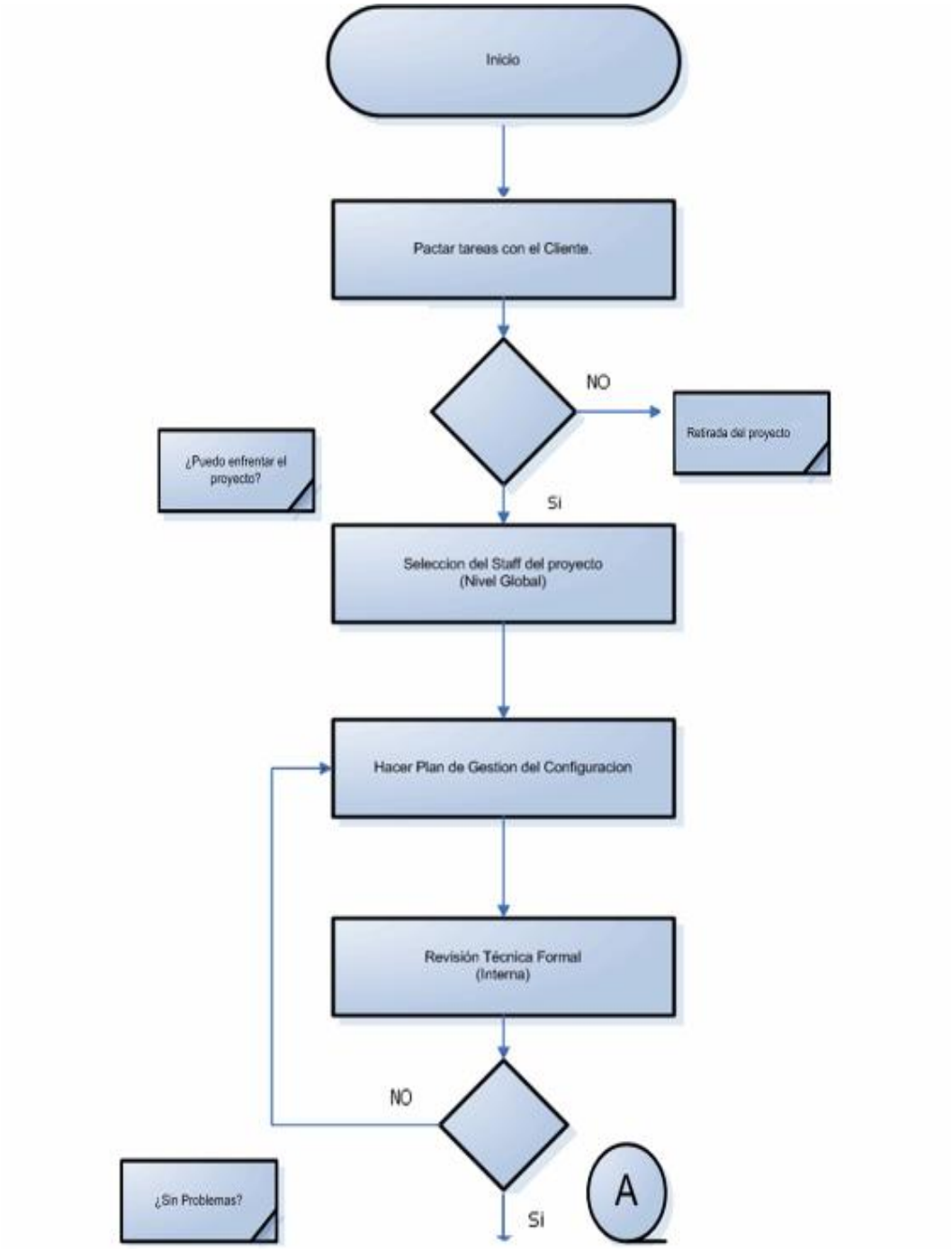
[SCHENONE, 2004] Schenone M, Tesis de Grado en Ingeniería en Informática. “Diseño de una Metodología Ágil de Desarrollo de Software”. Buenos Aires, 2004

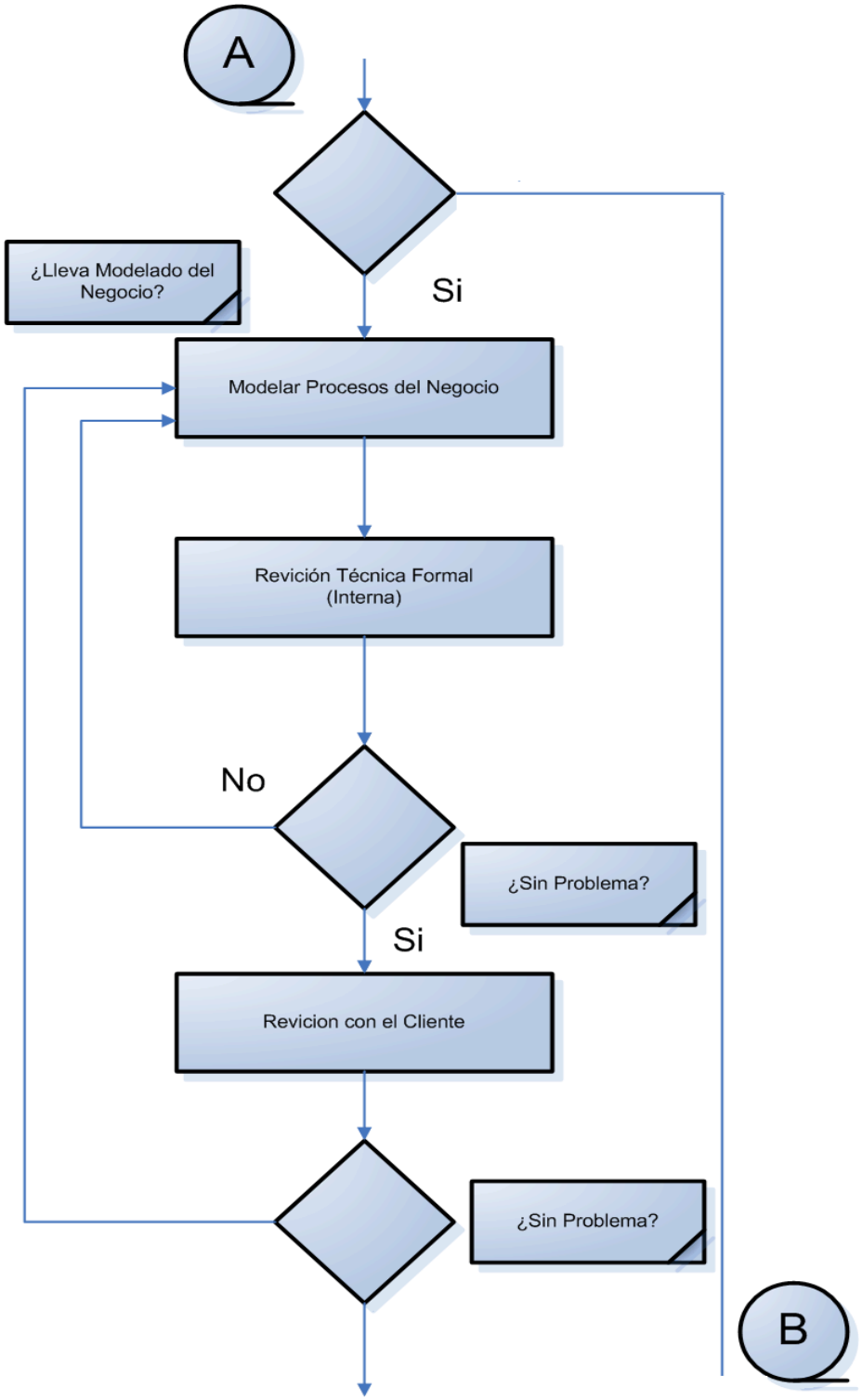
[WALKERDEN, JEFFERY, 1997] Walkerden, F. Jeffery, D., Software cost estimation: A review of models, process, and practice, Advances in Computers, 1997.

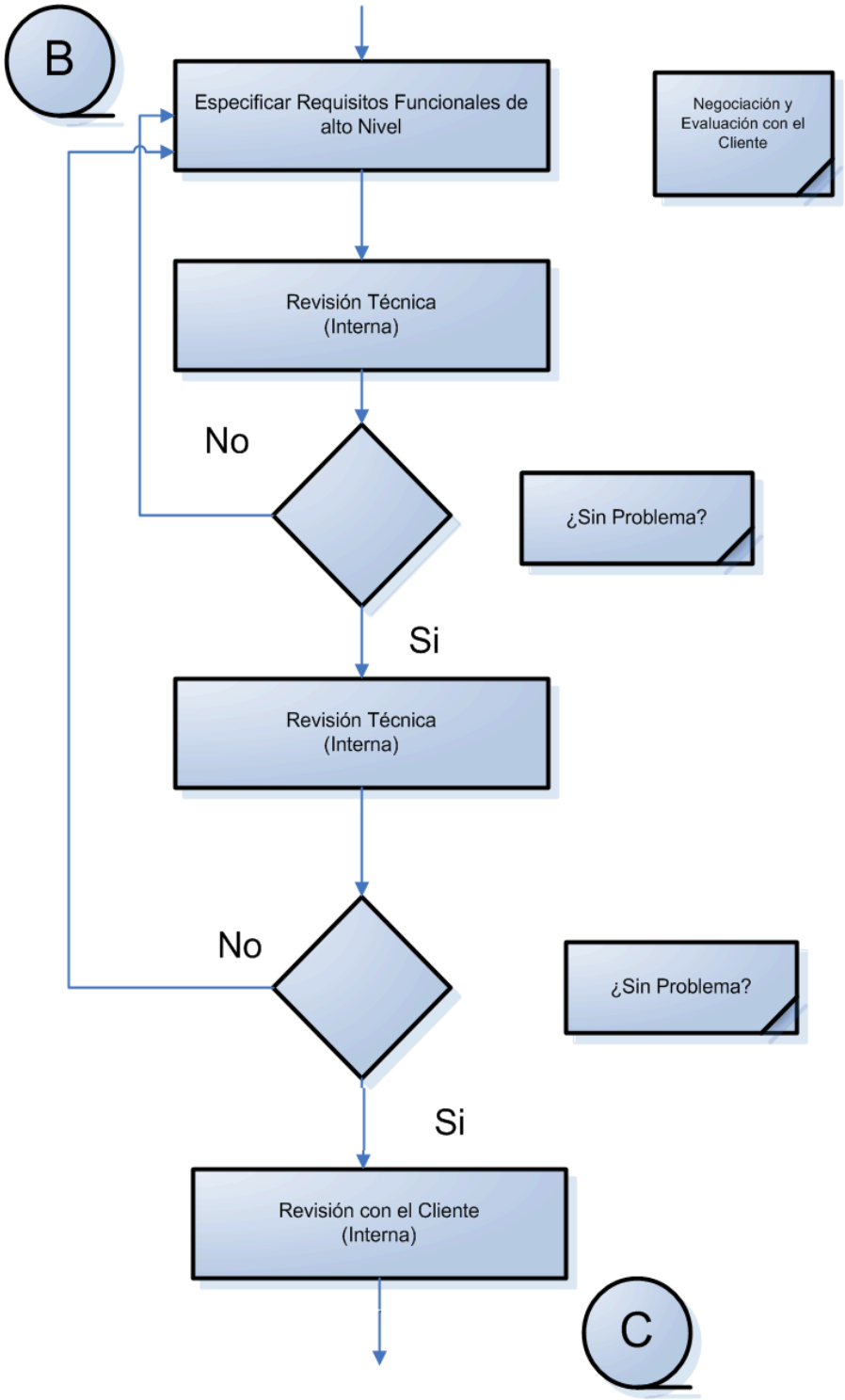
[WIECZOREK, BRIAND, 2001] Wieczorek, I. Briand, L. Resource estimation in software engineering, Technical Report. Edit. International Software Engineering Research, New York, 2001.

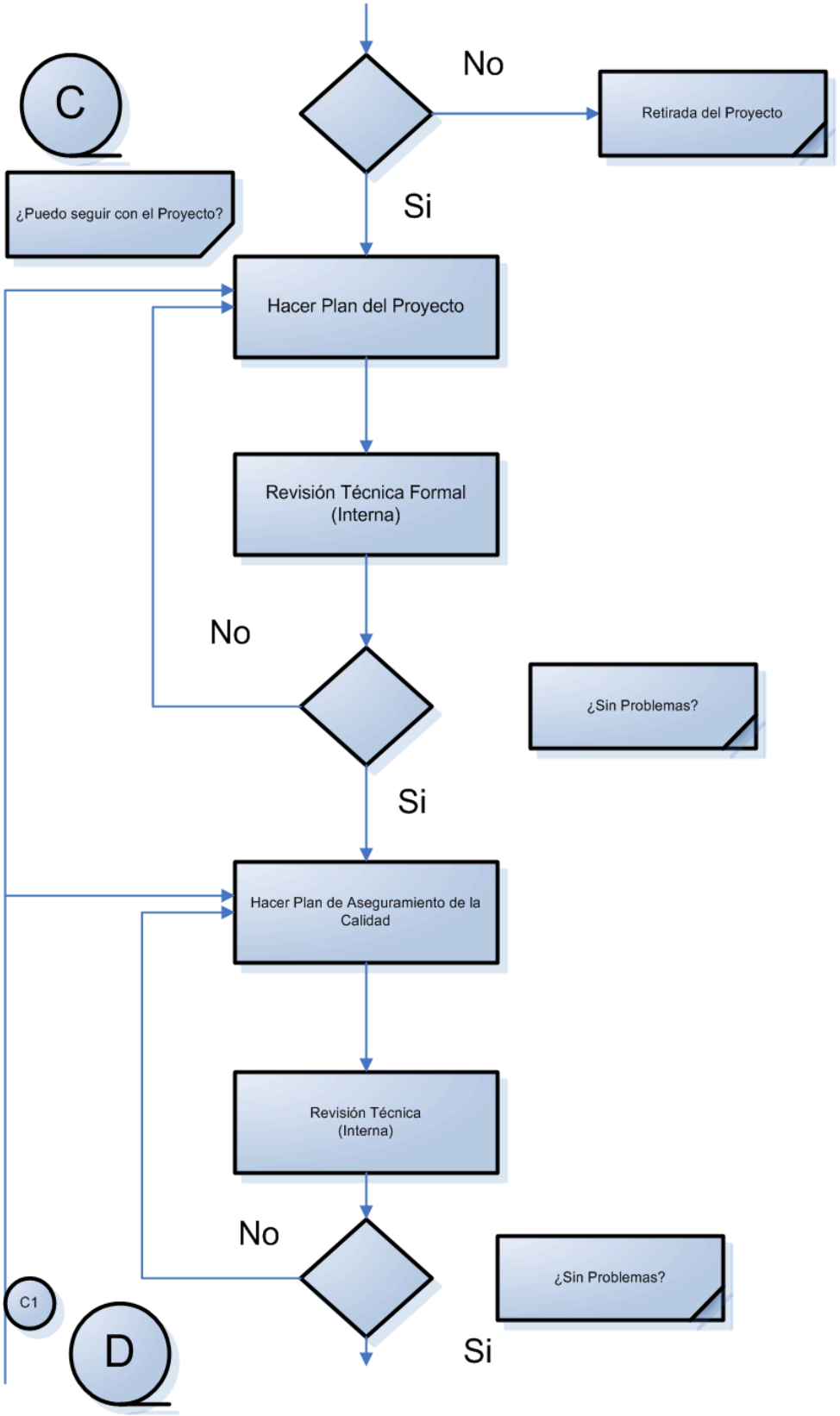
## Anexos

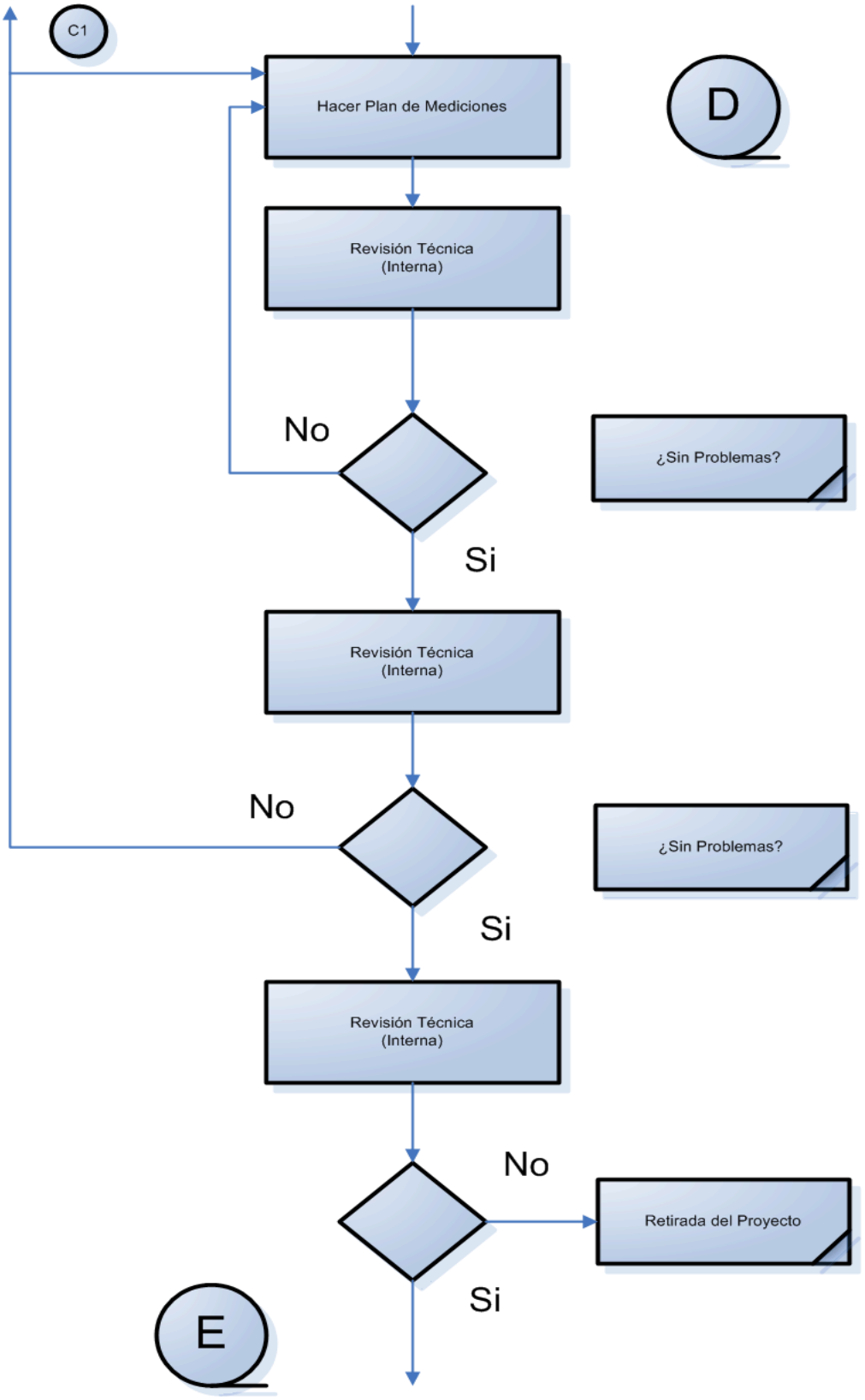
### Anexo I “Proceso ideal de Desarrollo de Software de la Facultad 5”

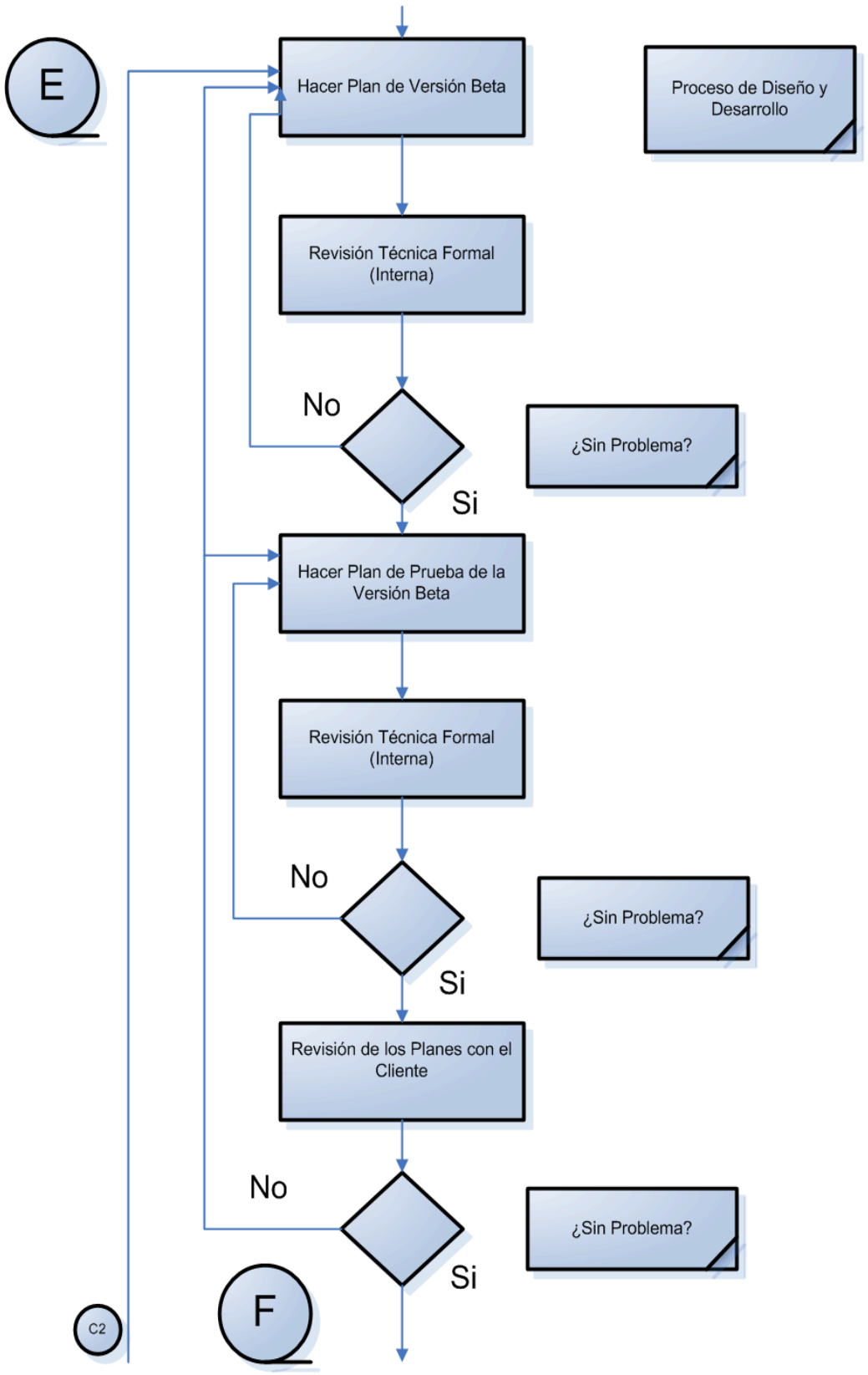


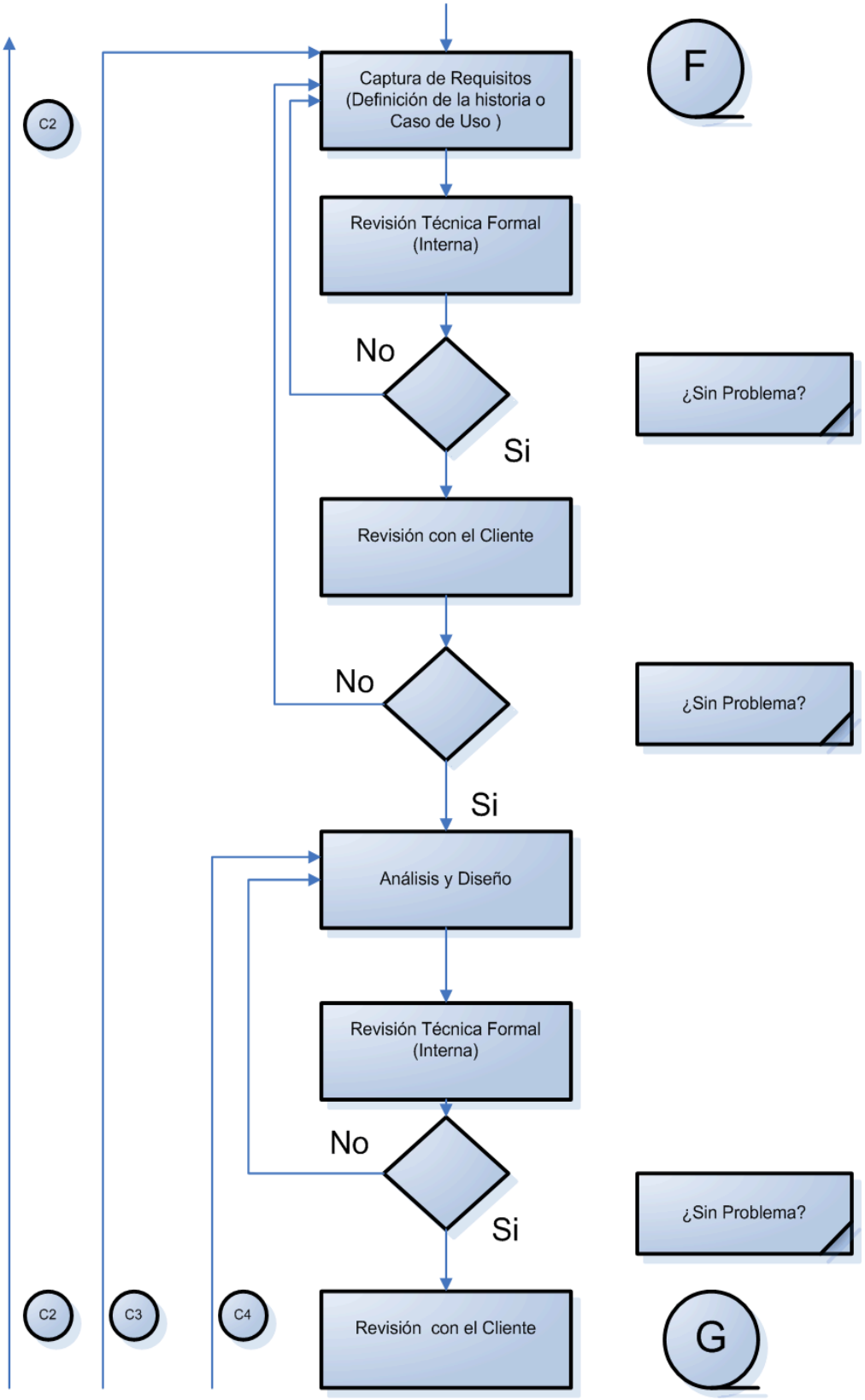

A
¿Lleva Modelado del Negocio?
Si
Modelar Procesos del Negocio
Revición Técnica Formal (Interna)
No
¿Sin Problema?
Si
Revicion con el Cliente
¿Sin Problema?
B

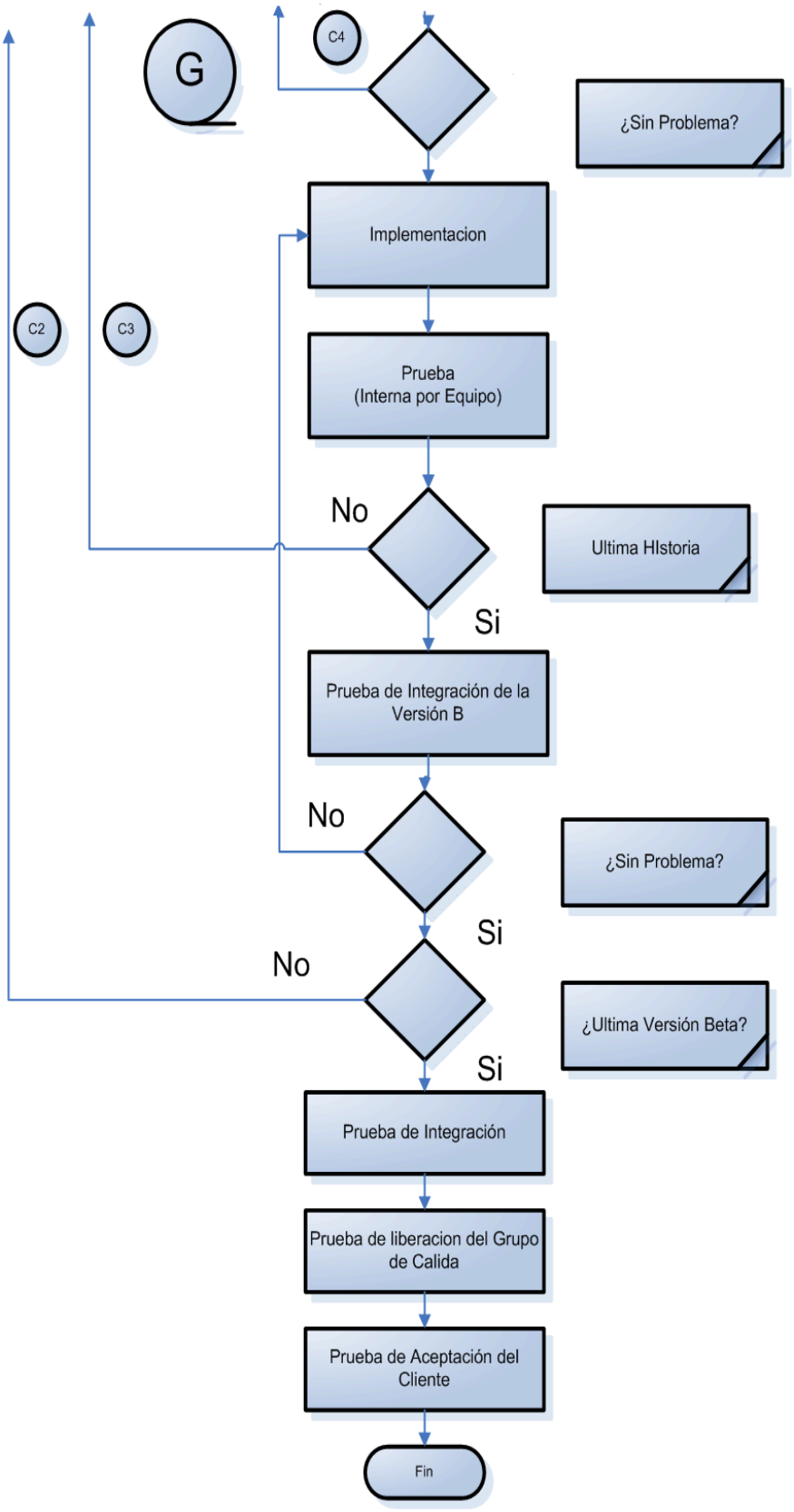

B
Especificar Requisitos Funcionales de alto Nivel
Negociación y Evaluación con el Cliente
Revisión Técnica (Interna)
No
¿Sin Problema?
Si
Revisión Técnica (Interna)
No
¿Sin Problema?
Si
Revisión con el Cliente (Interna)
C


C
¿Puedo seguir con el Proyecto?
No
Retirada del Proyecto
Si
Hacer Plan del Proyecto
Revisión Técnica Formal (Interna)
No
¿Sin Problemas?
Si
Hacer Plan de Aseguramiento de la Calidad
Revisión Técnica (Interna)
No
¿Sin Problemas?
Si
C1
D


C1
Hacer Plan de Mediciones
D
Revisión Técnica
(Interna)
No
¿Sin Problemas?
Si
Revisión Técnica
(Interna)
No
¿Sin Problemas?
Si
Revisión Técnica
(Interna)
No
Retirada del Proyecto
Si
E


E
Hacer Plan de Versión Beta
Proceso de Diseño y Desarrollo
Revisión Técnica Formal (Interna)
No
¿Sin Problema?
Si
Hacer Plan de Prueba de la Versión Beta
Revisión Técnica Formal (Interna)
No
¿Sin Problema?
Si
Revisión de los Planes con el Cliente
No
¿Sin Problema?
Si
F
C2


F
C2
Captura de Requisitos (Definición de la historia o Caso de Uso )
Revisión Técnica Formal (Interna)
No
¿Sin Problema?
Si
Revisión con el Cliente
No
¿Sin Problema?
Si
Análisis y Diseño
Revisión Técnica Formal (Interna)
No
¿Sin Problema?
Si
C2
C3
C4
Revisión con el Cliente
G


G
C4
¿Sin Problema?
Implementacion
C2
C3
Prueba
(Interna por Equipo)
No
Ultima HIstoria
Si
Prueba de Integración de la Versión B
No
¿Sin Problema?
Si
No
¿Ultima Versión Beta?
Si
Prueba de Integración
Prueba de liberacion del Grupo de Calida
Prueba de Aceptación del Cliente
Fin