CAPÍTULO I

NUEVOS ENFOQUES DE LA INGENIERÍA DE SOFTWARE.

1.1. INTRODUCCIÓN.

La ingeniería de software esta inmiscuida en cada proyecto que lleva implícita la calidad total de un sistema, si bien es posible la creación de aplicaciones sin tener en cuenta técnicas de ingeniería de software, a largo plazo, se ve la necesidad al momento de hacer cambios ya bien por actualización o nuevas necesidades del usuario, los desarrolladores entran en problemas al no poder ajustar el software a las necesidades del cliente.

En la actualidad existen muchas formas en las cuales se orienta la ingeniería de software de acuerdo a múltiples percepciones y campos.

Los programas tienen muchas propiedades matemáticas. Como ejemplo la corrección y la complejidad de muchos algoritmos son conceptos matemáticos que pueden ser probados. El uso de matemáticas en la IS[4] es llamado métodos formales. Edsger Dijkstra ha dicho que la IS es una rama de las matemáticas.

También el software tiene muchas propiedades científicas que se pueden medir. Por ejemplo, el desempeño y la escalabilidad de programas bajo diferentes cargas de trabajo puede ser medida. La efectividad de los cachés, procesadores más grandes, redes más rápidas, nuevas tecnologías de base de datos tienen que ver con la ciencia. Se pueden deducir ecuaciones matemáticas de las medidas.

La Ingeniería de Software es considerada por muchos como una disciplina de ingeniería porque tiene los puntos de vistas de expertos y las características esperadas de los ingenieros. Análisis, documentación, y código comentado son signos de un ingeniero. David Parnas ha argumentado que es una ingeniería.

Los programas son construidos en una secuencia de pasos. El hecho de definir propiamente y llevar a cabo estos pasos, como en una línea de ensamblaje, es necesario para mejorar la productividad de los desarrolladores y la calidad final

[4] IS, (Ingeniería de Software). Ingeniería de construcción de sistemas.

de los programas. Este punto de vista inspira los diferentes procesos y metodologías que se encuentra en la IS.

El software comercial requiere manejo de proyectos. Hay presupuestos y calendarios establecidos. Gente para liderar. Recursos espacio de oficina, computadoras por adquirir. Todo esto encaja apropiadamente con la visión del Manejo de Proyectos.

Los programas contienen muchos elementos artísticos. Las interfaces de usuario, la codificación. Incluso la decisión para un nombre de una variable o una clase. Donald Knuth es famoso porque ha argumentado que la programación es un arte.

1.2. METODOLOGÍAS DE CICLOS DE VIDA MÁS USADOS POR LAS HERRAMIENTAS ALM.

Las herramientas ALM hacen uso de modelos de ciclo de vida que anteriormente en la ingeniería de software se los usaba de un modo más rústico, entre los cuales se puede nombrar algunos como: ciclos de vida en cascada, espiral, prototipado, en v, a continuación detallamos dos modelos de ciclo de vida más usados actualmente.

1.2.1. MICROSOFT SOLUTIONS FRAMEWORK (MSF)

En realidad no es un modelo ágil, sigue siguiendo prescriptivo[5]. Este framework es considerado metamodelo ya que abarca varios modelos. MSF incorpora directamente prácticas para controlar requisitos de calidad de servicio como, equipo y proceso. Visual Studio Team System con Team Foundation Server es la herramienta para administrar el proceso de desarrollo de software, para ser desarrollado en una plataforma.

[5] Prescriptivo. Fija criterios, guías técnicas y soluciones determinadas por lo que no promueve la innovación y puede suponer barreras técnicas.

MSF tomó las mejores prácticas de los expertos e integró en distintos modelos, principios y guías.
MSF consiste en siete modelos, que pueden ser usados individualmente o combinados. Estos modelos son:

- Team Model (Modelo de Equipo)
- Process Model (Modelo de Procesos)
- Application Model (Modelo de Aplicación)
- Solution Design Model (Modelo de Diseño de Soluciones)
- Enterprise Architecture Model (Modelo de Arquitectura Empresarial)
- Infraestructure Model (Modelo de Infraestructura)
- Total Cost Ownership Model (Modelo de Costo Total de Propiedad)

El Modelo de Diseño de Soluciones ayuda al equipo del proyecto a anticiparse a las necesidades del usuario incluyéndolo en el problema.

En el Modelo de Diseño de Soluciones, los usuarios se ven involucrados en el proceso de diseño. Obteniendo de ellos información sobre ciertos detalles, como de funcionalidad y otros requerimientos, el equipo puede determinar cómo se va a usar la aplicación e incrementar su productividad.

Más allá de involucrar a los usuarios en el diseño, el Modelo de Diseño de Soluciones provee una estrategia para diseñar soluciones orientadas a negocios que deben ser creadas para satisfacer necesidades especificas. Este modelo une el Modelo de Equipo, el Modelo de Aplicación y el Modelo de Procesos, de tal manera que los recursos pueden ser enfocados en las áreas donde tengan mayor rendimiento.

ASPECTOS DEL MODELO DE DISEÑO DE SOLUCIONES

Las perspectivas son usadas para identificar los requerimientos técnicos y de negocios para la aplicación. El resultado de utilizar este modelo es una mejor distribución de los recursos del proyecto.

El Modelo de Diseño de Soluciones está compuesto por diferentes aspectos:

a) Diseño Conceptual

Es donde se origina el concepto inicial de la solución. Es en este diseño donde el equipo de desarrollo trata de entender las necesidades de los usuarios de la solución. Escenarios y modelos son usados para suavizar este entendimiento de manera que cada una de las entidades involucradas: equipos de desarrollo, clientes y usuarios, sepan que es lo que se necesita de la solución.

El proceso de Diseño Conceptual está compuesto de las siguientes tareas para determinar y substanciar los requerimientos de la aplicación:

- Identificación de usuarios y sus roles.
- Conseguir información de los usuarios.
- Validación del diseño.

Una vez que se ha llegado al final del proceso del Diseño Conceptual, se está generalmente listo para aplicar los documentos obtenidos al diseño lógico.

b) Diseño Lógico

Este diseño toma la información brindada por el Diseño Conceptual y la aplica al conocimiento técnico, es en éste diseño que la estructura y comunicación de los elementos de la solución son establecidas. Los objetos y servicios, la interfaz de usuario y la base de datos lógica son el conjunto de elementos identificados y diseñados en esta perspectiva.

El diseño lógico está comprendido de dos partes importantes:

1) <u>Organización de las Estructuras Lógicas.</u>

Una vez que se han identificado los objetos, es necesario organizarlos según los servicios que proveen, y las relaciones que tienen unos con otros.

2) Del Diseño Conceptual al Diseño Lógico.

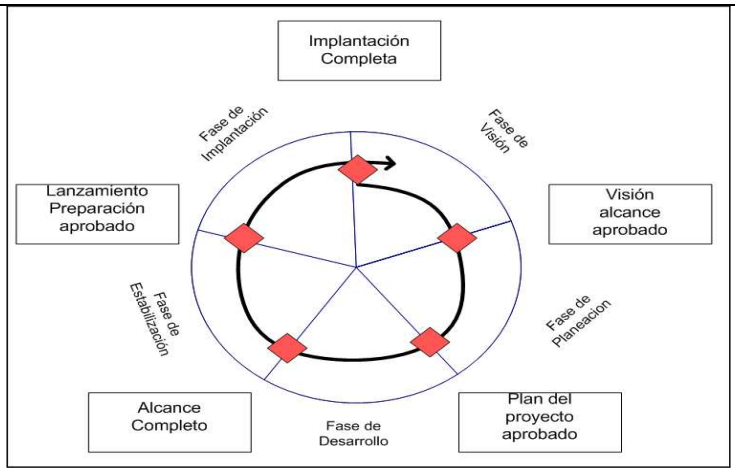
Cuando se diseñan los objetos es importante que estos se centren en una sola cosa en lo posible, en otras palabras los objetos deberían solamente proveer servicios relacionados con un único propósito.

c) Diseño Físico

Es donde los requerimientos del diseño conceptual y lógico son puestos en una forma tangible. Es en este diseño que las restricciones de la tecnología son aplicadas al Diseño Lógico de la solución. El Diseño Físico define cómo los componentes de la solución, así como la interfaz de usuario y la base de datos física trabajan juntos. Desempeño, implementación, ancho de banda, escalabilidad, adaptabilidad y mantenibilidad son todos resueltos e implementados a través del Diseño Físico.

CICLO DE VIDA DEL MSF

El marco de trabajo MSF de Microsoft Visual Studio Team System, en su modelo de ciclo de vida vemos el espiral debidamente delimitado en fases, las iteraciones también son planeadas para que haya predictibilidad en el proyecto. Cada rombo representa un hito en el proyecto.[6]



Fuente [6]

Figura 1.2 Modelo de proceso de desarrollo de aplicaciones MSF.

[6] www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node10.html

1.2.2. RUP (Proceso Racional Unificado)

Es un proceso de desarrollo de software y junto con UML[6], constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP[7] es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización

Los principales principios de desarrollo son:

Adaptar el proceso

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico.

Balancear prioridades

Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

Colaboración entre equipos

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.

---

[6] UML. Lenguaje Unificado de Modelado.
[7] RUP. Proceso Racional Unificado, es una metodología para desarrollo de software.

Demostrar valor iterativamente

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto.

Elevar el nivel de abstracción

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Esto previene a los ingenieros de software ir directamente de los requisitos a la codificación de software a la medida del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

Enfocarse en la calidad

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

Ciclo de vida del RUP.

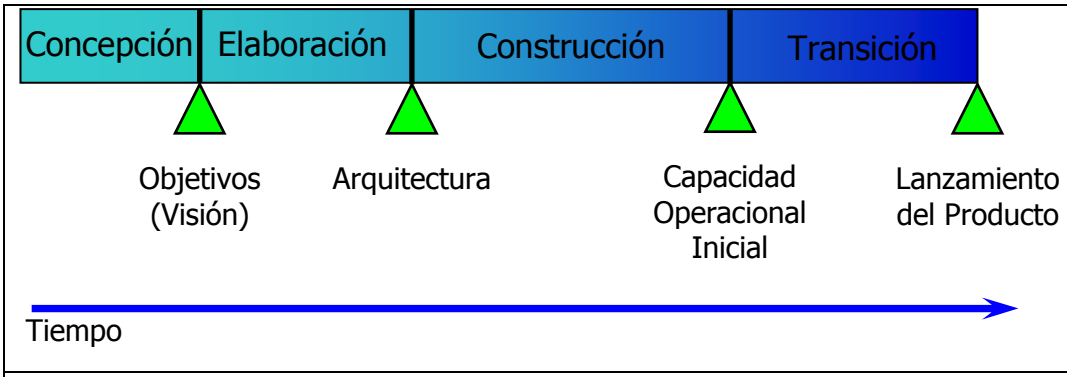
El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- **Concepción**: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.

- **Elaboración**: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- **Construcción**: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- **Implementación**: se instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

La duración y esfuerzo dedicado en cada fase es variable dependiendo de las características del proyecto.[7]



Fuente [7]

Figura 1.3. Concepción del Ciclo de Vida con metodología RUP.

El Proceso Unificado se enfoca en la arquitectura como el centro del desarrollo para asegurar que el desarrollo basado en componentes sea clave para un alto nivel de rehusó.

Se considera que hay cuatro perspectivas de arquitectura que cumplen los requerimientos de una empresa:

- **Arquitectura de Negocios** - Describe como opera un negocio. Desarrolla una imagen clara de los procesos de flujo de trabajo de la organización y de cómo son apoyados por una infraestructura tecnológica basada en servicios.

[7] http://babotejada.wordpress.com

- **Arquitectura de Aplicación**.- Adopta un modelo de aplicación de toda la empresa para diseñar y desarrollar sistemas de negocios que puedan compartir un conjunto de componentes back-end[8] de alto valor.

- **Arquitectura de Información.-** Define qué información es necesaria para apoyar el proceso de negocios y como poner esa información eficientemente en manos de quienes que la necesitan sin crear islas de datos inaccesibles ni sistemas redundantes.

- **Arquitectura Tecnológica** – Define los estándares y guías para la adquisición y despliegue de herramientas, bloques de construcción de aplicaciones, servicios de infraestructura, componentes de conectividad de red y plataformas cliente servidor.[8]

El Proceso Unificado es un proceso porque "define quién está haciendo qué, cuándo lo hacer y cómo alcanzar cierto objetivo, en este caso el desarrollo de software".

Fase e iteraciones	¿Cuándo se hace?
Flujos de trabajo de procesos (actividades y pasos)	¿Qué se está haciendo?
Artefactos (modelos, reportes, documentos)	¿Qué se produjo?
Trabajador: un arquitecto	¿Quién lo hace?
Fuente[8]	
Tabla 1.1 Conceptos clave del Proceso Unificado.	

La metodología RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no sea posible cubrir los costos de dedicación del equipo de profesionales necesario.

[8] Back-End. Se refieren al final de un proceso.
[8] http://babotejada.wordpress.com/2007/06/16/proceso-unificado-de-rational

1.3. CONSIDERACIONES DE LA GESTIÓN DEL CICLO DE VIDA EN LA INGENIERÍA DE SOFTWARE.

Cada fase del ciclo de vida aporta beneficios individualmente pero su valor aumenta cuando se integran y coordinan entre ellos de modo eficaz. Sin embargo, estas fases y herramientas deben asociarse a procesos ALM que tengan correspondencia con la cultura de la organización informática así como del negocio. El uso de mejores prácticas que faciliten el rigor y la adopción de herramientas ALM y que impacten en la cultura corporativa e informática existente es fundamental para tener éxito. Estas prácticas, junto con una sólida gestión de proyectos para programar tareas y distribuir recursos, crean una base sólida para un eficaz desarrollo de software, ya que proporcionan un marco de trabajo para el uso apropiado y una plantilla para el cambio de comportamiento.

Incluso con la mejor tecnología del mundo, las organizaciones no tendrán éxito si no utilizan las herramientas de forma óptima, del modo adecuado, en el momento adecuado.

Los procesos eficaces también ayudan a conseguir una mejor comunicación entre los grupos. Por ejemplo, los importantes requisitos de negocio actuales conducen al éxito de los proyectos. Estudios de mercado indican que el 70-80% de los fracasos en los proyectos son resultado directo de una pobre recopilación, gestión y análisis de requisitos. Una vez más, el proceso desempeña un papel fundamental. Una pobre comunicación entre los usuarios de negocio y el personal técnico conduce a requisitos obsoletos incluso antes de iniciar el proyecto. Permitir que los usuarios puedan comunicarse con los desarrolladores utilizando herramientas de negocio con las que están familiarizados como, Microsoft Word o Excel, y facilitar los flujos de comunicación entre el negocio y la informática son elementos clave para mejorar la calidad y garantizar la vigencia de los requisitos.

Estos requisitos pueden ayudar a construir el modelo. Para necesidades de requisitos más rigurosos, herramientas de alta gama adicionales proporcionan funciones de seguimiento y una sólida gestión.

Un modelado adecuado permite a las organizaciones crear aplicaciones que reflejen las necesidades del negocio y las demandas arquitectónicas y de sistema.

Los planteamientos de procesos iterativos permiten una interacción continua entre los usuarios de negocio, los diseñadores y los responsables de la implementación de código. De esta forma, la probabilidad de que el contenido del código refleje las demandas actuales del negocio es mucho mayor. Los propios diseñadores desempeñan diversos roles. Normalmente, existe una desconexión entre los arquitectos y las operaciones que impacta negativamente y conlleva un incremento del coste de implantación del software. Estos temas pueden abordarse con mayor rigor con una mayor visibilidad de los retos de implementación e infraestructura mediante la comunicación entre arquitectos, diseñadores y personal de operaciones. Habilitar herramientas de diseño completas que faciliten la visibilidad entre grupos y en todo el ciclo de vida resulta también esencial.

Para los desarrolladores, facilitar el proceso de desarrollo mediante una estrecha coordinación entre el IDE elegido y la gestión del ciclo de vida también es importante, incluyendo pruebas, gestión de cambios y configuraciones de software, y control de versiones.

Encontrar errores en el inicio del ciclo de vida de una aplicación reduce los costes.

Diversos estudios del mercado indican que encontrar problemas de software tarde en el ciclo de vida de una aplicación incrementa entre 10 y 100 veces el coste de encontrar y solucionar el defecto durante las pruebas unitarias iníciales. Los costes de una reputación debilitada además de la pérdida de ingresos de los sistemas transaccionales y de cara al cliente son incalculables. Activar pruebas iníciales y frecuentes mediante el soporte de pruebas unitarias es el primer paso.

Esto significa que la calidad del código producido para fases posteriores del sistema mejorará con las pruebas de integración del sistema y la posterior

aceptación de los usuarios y su implantación. Además, es importante saber qué código ha sido probado, así como abordar pruebas de rendimiento y de carga, pruebas manuales, y la gestión del proceso de pruebas. Esto permite a los probadores estructurar y gestionar mejor sus planteamientos de realización de pruebas en coordinación con el desarrollo. La mejora de la calidad reduce los costes para el negocio, aumenta la capacidad de respuesta informática, y facilita la transición a las operaciones y al mantenimiento.

La gestión de cambios y configuraciones de software y el control de versiones son elementos clave para los desarrolladores en la gestión eficaz de la creación de código. Un pobre cuidado en relación a la gestión de cambios de código así como una insuficiente coordinación con la realización de pruebas una vez implementados los cambios abre la puerta a fallos y caos en las revisiones de diversas formas. Los cambios en sí mismos pueden presentar problemas de calidad y necesitan ser gestionados para permitir la colaboración de equipo (en lugar de confusión de código) y versiones de software coherentes que estén en línea con las principales necesidades del negocio. La priorización de las peticiones de cambios es también esencial en este contexto. Los informes sobre la respuesta del departamento de informática a las peticiones de cambios proporcionan otra métrica cuantitativa para alimentar el análisis y el entendimiento ejecutivo sobre la eficacia de los recursos informáticos.

En términos generales, está claro que cada una de estas fases debe estar bien establecida individualmente. Sin embargo, la coordinación de estas fases con herramientas automatizadas eficaces que estén estrechamente integradas y que se hayan implantado utilizando una gestión adecuada del proceso y del proyecto producirá importantes beneficios al negocio. Estos beneficios incluyen respuestas adaptables y flexibles a las presiones competitivas y menores costes para la creación, calidad y gestión del código en la fase de implantación.

Como el reto de las barreras humanas a la evolución ALM tiende a ser mayor, facilitar la adopción mediante productos accesibles e intuitivos y la gestión de procesos y proyectos son elementos esenciales para el éxito.[9]

1.4. BENEFICIOS QUE PRESTAN LAS HERRAMIENTAS ALM.

Los beneficios de las herramientas de Gestión del Ciclo de Vida de las Aplicaciones son innumerables, en un futuro estas podrán expandirse a mayor escala, con mayores ventajas, que afectan a la economía y las sociedades de muchas maneras entre las cuales podemos detallar:

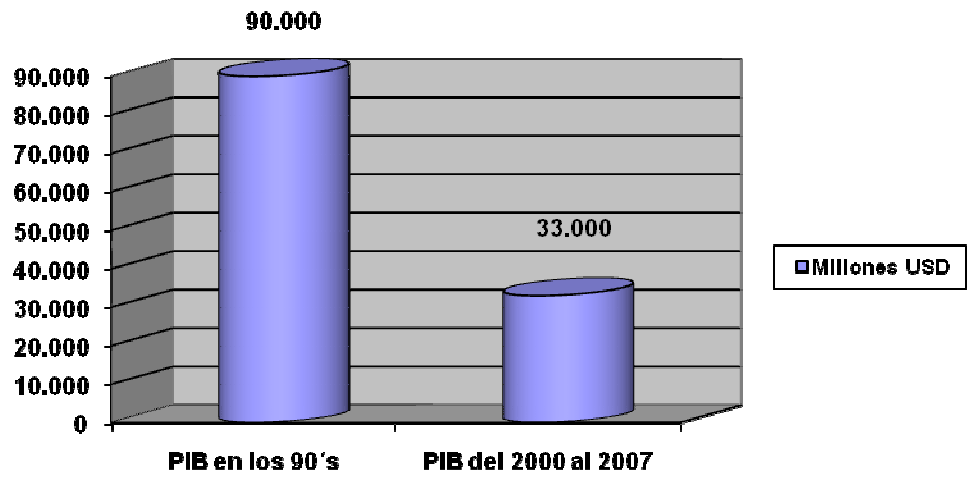
- Económicamente.
- Socialmente.
- Metodológicamente.

1.4.1. ECONÓMICAMENTE

En los EEUU, el software contribuyó a 1/4 de todo el incremento del PIB[9] durante los 90's, alrededor de 90,000 millones de dólares por año, y 1/6 de todo el crecimiento de productividad durante los últimos años de la década, alrededor de 33,000 millones de dólares por año. La ingeniería de software contribuyó a $1 billón de crecimiento económico y productividad en esa década. Alrededor del globo, el software contribuye al crecimiento económico en formas similares, aunque es difícil de encontrar estadísticas fiables.[10]

---

[9] http://www.danysoft.info/free/vs05tsA.pdf
9 PIB. Producto interno bruto.
[10] http://es.wikipedia.org/wiki/Desarrollo_de_software



Fuente [10]

Figura 1.4 PIB EN EEUU EN LA INDUSTRIA DE SOFTWARE.

1.4.2. SOCIALMENTE

La ingeniería de software cambia la cultura del mundo debido al extendido uso de la computadora. El correo electrónico, la WWW[10] y la mensajería instantánea permiten a la gente interactuar en nuevas formas. El software baja el costo y mejora la calidad de los servicios de salud, los departamentos de bomberos, las dependencias gubernamentales y otros servicios sociales. Los proyectos exitosos donde se han usado métodos de ingeniería de software incluyen a Linux[11], el software del transbordador espacial, los cajeros automáticos y muchos otros.

1.4.3. METODOLÓGICAMENTE

Las herramientas ALM se encargan de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y

[10] WWW **World Wide Web** (o la "**Web**") es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet.
[11] Linux. Es un sistema operativo tipo Unix (también conocido como GNU/Linux) que se distribuye bajo la Licencia Pública General de GNU o GPL, es decir que es software libre.

la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Normalmente durante su tiempo de vida los sistemas tienen muchas versiones, pudiendo durar incluso más de 10 años. Existen herramientas CASE[12] para la gestión de la configuración y otras denominadas de ingeniería inversa para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.

Para mejorar el proceso es básico disponer de datos numéricos que evidencian la efectividad de la aplicación del proceso con respecto a cualquier producto software resultante del proceso. Para disponer de estos datos, la metodología debe contener un conjunto de mediciones de proceso para identificar la calidad y costo asociado a cada etapa del proceso.

1.5. CONSIDERACIONES

La ingeniería de software a demostrado estar vinculada estrechamente con la matemática, ciencia, ingeniería, manufactura, manejo de proyectos y el arte, lo que ha logrado que se fusione con estos campos de la humanidad al mismo tiempo que ayuda a potenciarlos, ayuda a los humanos a entender de una mejor manera los procesos, además que permite retroalimentar los sistemas generales haciendo cada vez más rápido la creación de sistemas funcionales.

Es importante la ingeniería del software, porque permite desarrollar aplicaciones de una manera más sencilla permitiendo disminuir costos mejorar los tiempos de desarrollo con lo que administra de mejor manera los proyectos, dando paso al nacimiento de la Revolución Industria del Software, y a la creación de las nuevas herramientas de gestión del ciclo de vida del software, con las cuales se ha logrado mejorar en la calidad, así como también se logra una mejor coordinación

[12] CASE. Ingeniería Asistida por Computadora.

en los equipos y una integración implícita de todas las fases de un ciclo de vida del software como son:

- Análisis de requisitos.
- Especificación.
- Diseño y arquitectura.
- Programación.
- Prueba.
- Documentación.
- Mantenimiento.

Para obtener mayor información revise la revista Software Gurú el siguiente enlace http://www.dotnetmania.com/Articulos/008/editorial.html.