1.1.- CONCEPTOS DE INGENIERÍA DE SOFTWARE

Definición: La IEEE en su publicación Standard Collection: SoftwareEngineering define a la Ingeniería del software como:

"La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software."

Ian Sommerville la define en su libro Ingeniería del Software como:

"...una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después que se utiliza..."

La ingeniería de software es un intento para la solución de un problema, es decir una disciplina cuyo objetivo es la producción de software libre de fallas, enviado a tiempo y dentro del presupuesto, que satisfaga las necesidades del cliente. Además, el software debe modificarse con facilidad cuando cambian las necesidades del usuario.

1.1.1.- CARACTERISTICAS DE LA INGENIERÍA DE SOFTWARE

La ingeniería de software a lo largo de los años se ha caracterizado por abarcar un conjunto de elementos claves que son:

- Métodos
- Herramientas, y
- Procedimientos

Estos elementos facilitan al gestor controlar el proceso de desarrollo de software y suministrar las bases para construir software de alta calidad de una forma productiva.

Los **métodos** indican "cómo" construir técnicamente el software. Estos métodos abarcan tareas que incluyen: planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento. Los métodos de la ingeniería de software introducen frecuentemente una notación especial orientada a un lenguaje o gráfica y un conjunto de criterios para la calidad del software

Las **herramientas** suministran un soporte automático o semiautomático para los métodos. Cuando se integran las herramientas de forma que la información creada por una herramienta pueda ser usada por otra; se establece un sistema de soporte llamada Ingeniería de Software Asistida por Computadora (CASE).

Los **procedimientos** son el pegamento que junta los métodos y las herramientas, y facilita un desarrollo racional y oportuno del software de computador. Definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requieren, los controles para asegurar la calidad y coordinar los cambios, y las directrices que ayudan a los gestores de software a evaluar el progreso.

1.2.- CLASIFICACIÓN DE SOFTWARE

***1.2.1.-* Software de sistemas.** Está formado por todos aquellos programas cuya finalidad es servir al desarrollo o al funcionamiento de otros programas. Estos programas son muy variados: editores, compiladores, sistemas operativos, entornos gráficos, programas de telecomunicaciones, entre otros, pero se caracterizan por estar muy próximos al hardware, por ser utilizados concurrentemente por numerosos usuarios y por tratarse de programas de

amplia difusión, no estando diseñados normalmente a medida. Esto permite un mayor esfuerzo en su diseño y optimización, pero también les obliga a ser muy fiables, cumpliendo estrictamente las especificaciones para las que fueron creados. Un ejemplo de este tipo de software son los sistemas operativos, como Windows, Unix y Linux.

1.2.2.-**Software de tiempo real.** El software que coordina, analiza y controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo, y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse la repuesta en tiempo real (típicamente en el rango de un milisegundo a un segundo). Un ejemplo de estos Software son los sistemas de climático, petrolero y sísmico.

1.2.3.-**Software de gestión.** El proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los «sistemas» discretos (por ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión (SIG) que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas).

1.2.4.-**Software de ingeniería y científico.** Es el que se encarga de realizar cálculos complejos sobre datos numéricos de todo tipo. En este caso la

corrección y exactitud de las operaciones que realizan es uno de los requisitos básicos que deben de cumplir.

El campo del software científico y de ingeniería se ha visto ampliado últimamente con el desarrollo de los sistemas de diseño, ingeniería y fabricación asistida por ordenador (Diseño asistido por computador CAD), los simuladores gráficos.

1.2.5.-**Software empotrado.** Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura, se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa, con capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado, etc.).

1.2.6.-**Software de computadoras personales.** El mercado del software de computadoras personales se ha generalizado en las pasadas dos décadas. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, aplicaciones financieras de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

1.2.7.-**Software de inteligencia artificial.** El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también llamados sistemas basados en el conocimiento, reconocimiento de patrones (imágenes y voz), redes neuronales artificiales, prueba de teoremas, y los juegos son representativos de las aplicaciones de esta categoría.

1.3.- MITOS DE LA INGENIERÍA DE SOFTWARE.

Los mitos del software son creencias acerca del software y de los procesos empleados para construirlo.

En la actualidad, la mayoría de los profesionales reconocidos en la ingeniería del software identifican los mitos en su real dimensión: actitudes equivocadas que han causado problemas serios a los administradores y al personal técnico por igual. Sin embargo, las antiguas actitudes y viejos hábitos son difíciles de modificar, por lo que aún subsisten creencias falsas sobre el software. Es por ello qué, los mitos se pueden dividir en:

***1.3.1.*-Mitos de la Administración.** Los administradores con responsabilidades sobre el software, a menudo están bajo presión por mantener los presupuestos, evitar que los itinerarios se extiendan y mejorar la calidad. Con frecuencia el administrador del software se aferra a un mito si siente que esta creencia reducirá la presión (aun en forma temporal).

a. **Mito:** *Ya se tiene un libro lleno de estándares y procedimientos para la construcción de software. ¿Esto proporcionará a mi gente todo el conocimiento necesario?*

Realidad: Tal vez sea verdad que el libro de estándares existe, pero ¿se usa? ¿Los encargados de la construcción del software saben de su existencia? ¿El libro refleja la práctica moderna de la ingeniería del software? ¿Está completo? ¿Es adaptable? ¿Está dirigido al mejoramiento del tiempo de entrega sin dejar de enfocarse en la calidad? En muchos casos la respuesta a todas estas preguntas es no, debido a que todo desarrollo de aplicaciones son diferentes sus requisitos, ambiente y grupos de trabajos.

b. **Mito:** *Si se está atrasado en el itinerario es posible contratar más programadores para así terminar a tiempo.*

Realidad: El desarrollo de software no es un proceso mecánico como la manufactura. En palabras de Brooks [BR075]: "Agregar gente a un proyecto de software atrasado lo atrasa más". De inicio, este enunciado podría parecer contrario a la intuición. Sin embargo, cuando se agregan nuevos integrantes a un equipo, la gente que ya estaba trabajando debe invertir tiempo en la enseñanza a los recién llegados, lo cual reduce el tiempo dedicado al esfuerzo para el desarrollo productivo. Se puede agregar gente, pero sólo de una manera planeada y bien coordinada.

c. **Mito:** *Si decido subcontratar el proyecto de software a un tercero, puedo relejarme y dejar que esa compañía lo construya.*

Realidad: Si una organización no entiende cómo administrar y controlar internamente los proyectos de software, de manera invariable entrará en conflicto al subcontratar este tipo de proyectos.

***1.3.2.-*Mitos del Cliente.** El cliente que solicita un software de computadora puede ser la persona del escritorio, un grupo técnico, el departamento de ventas o de mercadotecnia, o una compañía externa que ha solicitado el software bajo contrato. En muchos casos, el cliente cree en los mitos que existen sobre el software, debido a que los gestores y desarrolladores de software hacen muy poco para corregir la mala información. Los mitos conducen a que el cliente se cree una falsa expectativa y, finalmente, quede insatisfecho con el desarrollador del software.

a. **Mito:** *Un enunciado general de los objetivos es suficiente para comenzar a escribir o desarrollar el programa; los detalles se pueden afinar después.*

Realidad: A pesar de que no siempre es factible que el enunciado de los requerimientos sea comprensible y estable, un enunciado ambiguo de los objetivos es la receta perfecta para el desastre. Los requerimientos precisos se desarrollan sólo mediante la comunicación continua y efectiva entre el cliente y el desarrollador.

b. **Mito:** *Los requerimientos del proyecto cambian de manera continua, pero el cambio puede ajustarse con facilidad porque el software es flexible.*

Realidad: Es verdad que los requerimientos del software cambian, pero el impacto del cambio varía de acuerdo con el momento en que éste se introduce. Cuando los cambios en los requerimientos se solicitan en etapas tempranas (antes de iniciar con el diseño o el código), el impacto en el costo es relativamente pequeño. Sin embargo, conforme pasa el tiempo, el impacto en el costo crece con rapidez en los cuales se han distribuido los recursos, estableciendo un marco general para el diseño y el cambio puede provocar una convulsión que requiera recursos adicionales y una modificación significativa en el diseño.

***1.3.3.*-Mitos del Desarrollador.** Los mitos en los que aun creen muchos desarrolladores se han ido fomentando durante 50 años de cultura informática. Durante los primeros días del desarrollo del software, la programación se veía como un arte. Las viejas formas y actitudes tardan en morir.

a. **Mito:** *Una vez que el programa ha sido escrito y puesto a funcionar, el trabajo está terminado.*

Realidad: Alguien dijo alguna vez que entre más rápido se comience a escribir código, más tiempo pasará para que el programa esté terminado. Los datos de la industria indican que entre 60 y 80 por

ciento de todo el esfuerzo aplicado en el software se realizará después de que el sistema haya sido entregado al cliente por primera vez.

b. Mito: *Mientras el programa no se esté ejecutando, no existe forma de evaluar su calidad.*

Realidad: Uno de los mecanismos más efectivos para el aseguramiento de la calidad del software se puede aplicar desde el inicio de un proyecto:

- la revisión técnica formal.
- Las revisiones al software.

Son un "filtro de calidad" que han probado ser más efectivas que las pruebas para encontrar ciertas clases de errores en el software.

c. Mito: *El único producto del trabajo que puede entregarse para tener un proyecto exitoso es el programa en funcionamiento.*

Realidad: Un programa en funcionamiento es sólo una parte de la configuración del software que incluye muchos elementos. La documentación proporciona un fundamento para la ingeniería exitosa y, aún más importante, representa una guía para el mantenimiento del software.

1.4.- Etapas de la ingeniería de software.

La etapa de la ingeniería de software es un enfoque sistemático y secuencial del desarrollo de software, que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba, documentación y mantenimiento del software. Entra las etapas podemos tener:

1.4.1.- Etapa de la Ingeniería y Análisis del Sistema:

La ingeniería y análisis del sistema abarcan los requerimientos globales a un nivel de sistema con una pequeña cantidad de análisis y diseño a nivel superior. Además de un análisis costo beneficio del sistema, es decir si toda la inversión que se hará para el sistema conviene a los beneficios que traerá el mismo.

1.4.2.- Etapa del Análisis de requisitos

El proceso de recoger los requerimientos se centra y se intensifica especialmente en esta etapa. Para comprender la naturaleza de los programas que hay que construir. El ingeniero de software debe comprender el dominio de la información del software, así como la función, rendimiento e interfaces requeridas. En esta etapa los requerimientos del sistema se documentan y se analizan con el cliente.

1.4.3.- Etapa del Diseño

El diseño del software es realmente un proceso multipaso que se enfoca sobre 4 atributos del programa, los cuales son:

a. **Estructura de datos:** Especifica las estructuras de datos necesarias para implementar el sistema.
b. **Arquitectura de software:** Define las relaciones entre los elementos estructurales (módulos) del programa.
c. **Detalle procedimental:** Transforma los elementos estructurales de la arquitectura del programa en una descripción procedimental.
d. **Características de la interfaz:** Describe como se comunica el software consigo mismo, con los sistemas que operan con él y con los operadores que lo emplean.

El diseño traduce los requerimientos en una representación del software la cual pueda ser establecida de forma que, obtenga la calidad requerida antes del comienzo de la codificación. En esta etapa los requerimientos y el diseño se documentan, para que forme parte de la configuración del software.

1.4.4.- Etapa de la Codificación

El diseño debe traducirse en una forma legible para la máquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

1.4.5.- Etapa de la Prueba

Consiste en comprobar que el software realice correctamente las tareas indicadas. Una técnica es probar por separado cada módulo del software, y luego probarlo de forma integral, para así llegar al objetivo. Se considera una buena práctica el que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó, idealmente un área de pruebas. En general hay dos grandes formas de organizar un área de pruebas:

a. La primera es que esté compuesta por personal inexperto y desconozca el tema de pruebas; de esta forma, se evalúa que la documentación entregada sea de calidad, los procesos descritos sean tan claros que cualquiera puede entenderlos y el software hace las cosas tal y como están descritas.

b. El segundo enfoque es tener un área de pruebas conformada por programadores con experiencia, personas que saben sin mayores indicaciones en cuáles condiciones puede fallar una aplicación y pueden poner atención en detalles que personal inexperto no consideraría.

1.4.6.- Etapa de la Documentación

Todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones, diagramas, pruebas, manuales de usuario, manuales técnicos; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

1.4.7.- Etapa del Mantenimiento

Se centra en el *cambio* que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios:

a. ***Mantenimiento Corrección.*** El *mantenimiento correctivo* cambia el software para corregir los defectos.
b. ***Mantenimiento Adaptación.*** El *mantenimiento adaptativo* produce modificación en el software para acomodarlo a los cambios de su entorno externo.
c. ***Mantenimiento Mejora.*** El *mantenimiento de mejora* lleva al software más allá de sus requisitos funcionales originales.
d. ***Mantenimiento Prevención.*** El mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente.

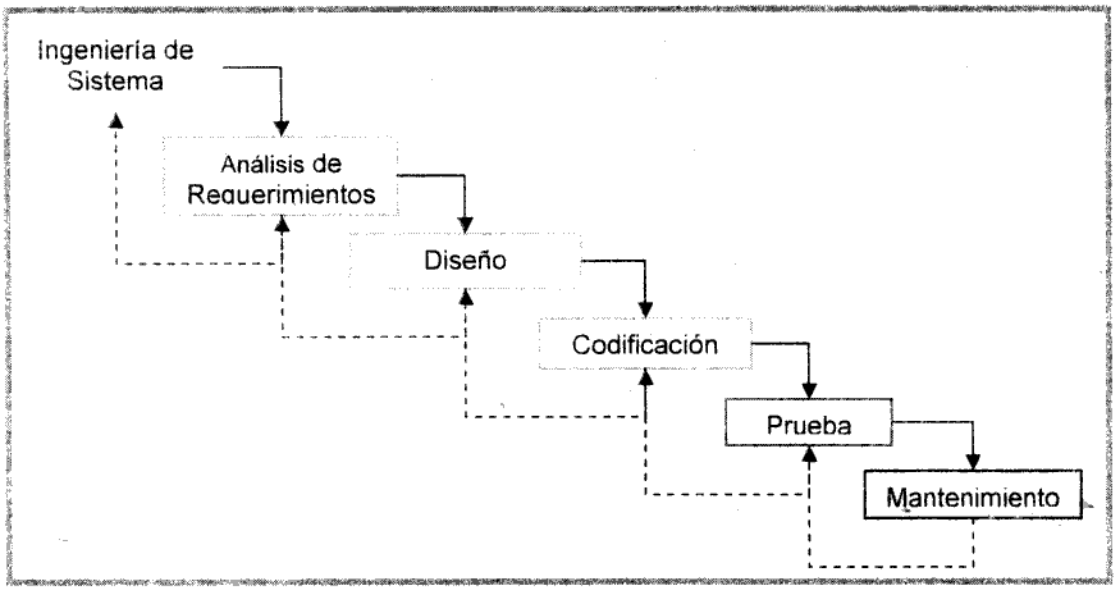
1.5.- Arquitectura del Software.

Existen varias arquitecturas de desarrollo de software definido y diseñado, que se utilizan o emplean durante el proceso de desarrollo de software, estos también son conocidos como "Modelos de Proceso de desarrollo de software". Cada modelo de proceso sigue un ciclo de vida particular, con el fin de asegurar el éxito en el proceso de desarrollo de software. Entre ellos tenemos:

1.5.1.- Modelo de cascada

Este modelo fue propuesto por Winston Royce en los años 70. También llamado ciclo de vida básico o modelo lineal-secuencial, este modelo divide el proceso de desarrollo en un conjunto de fases secuenciales e independiente, una fase no puede empezar hasta que no haya terminado la anterior, al final de cada una, el personal de desarrollo y los usuarios revisan el progreso del proyecto y se genera todo un conjunto de documentos.

Las fases de este modelo son:



a. Ingeniería de Sistema

- Es donde se establece los requerimientos del sistema y una parte de estos son asignado al software.
- Estos requerimientos son necesario para establecer la relación entre el software y el mundo exterior (hardware, usuario, base de datos u otros).

b. Análisis de Requerimientos

- Visión profunda del problema desde el punto de vista de los desarrolladores y usuarios.
- Especifica la información sobre la cual el software se va a desarrollar.
- La representación grafica del análisis de los requerimientos motiva su uso para evitar la ambigüedad de los requerimientos.

c. Diseño

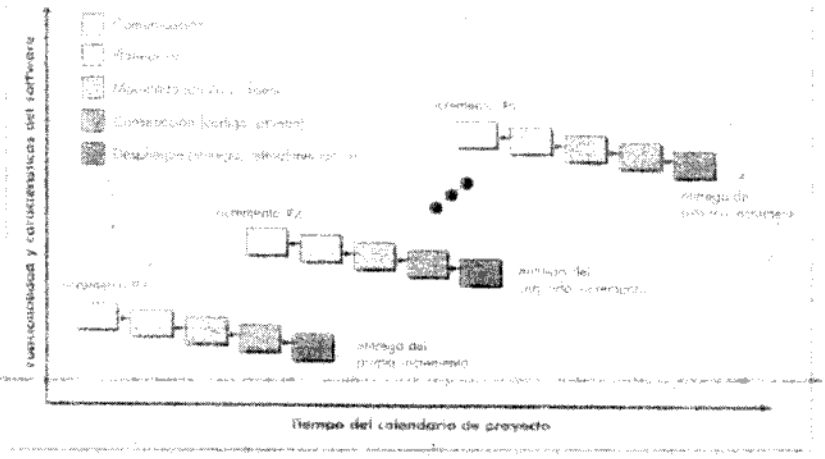
- Permite describir cómo el software va a satisfacer los requerimientos del usuario.

2.5.1.-Modelo de Procesos Incrementales

2.5.1.1.- Modelo Incremental

El **modelo incremental** combina elementos del modelo en cascada aplicado en forma iterativa. Este modelo aplica secuencias lineales de manera escalonada conforme avanza el tiempo en el calendario. Cada secuencia lineal produce "incrementos" del software. Se debe tener en cuenta que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos de ser necesario.

Al utilizar un modelo incremental el primer incremento es un **producto esencial.** Es decir, se incorporan los requisitos básicos. El producto esencial queda en manos del cliente para ser evaluado. Como resultado de la evaluación se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto esencial con el fin de satisfacer de mejor manera las necesidades del cliente y la entrega de características y funcionalidades adicionales. Este proceso se repite después de la entrega de cada incremento mientras no se haya elaborado el producto completo.



2.5.1.2.- Ventajas y Desventajas del Modelo Incremental

VENTAJA

- Reduce costos, pues si algo sale mal solo se regresa a la antigua versión y se comienza de nuevo.

- Es que al usuario se le entrega parte del producto, es decir una versión con la cual él puede trabajar.

Desventaja

- Que no todo proyecto puede tener versiones entonces es por eso que el desarrollador debe hacer un análisis previo.

2.5.1.3.- Modelo DRA

El desarrollo rápido de aplicaciones (DRA) es un modelo de proceso de software incremental que resalta un ciclo de desarrollo corto. El modelo DRA es una adaptación a "alta velocidad" del modelo en cascada en el que se logra el desarrollo rápido mediante un enfoque de construcción basado en componentes. Si se entienden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite que un equipo de desarrollo cree un "sistema completamente funcional" dentro de un periodo muy corto (por ejemplo, de 60 a 90 días).

Las actividades del modelo DRA:

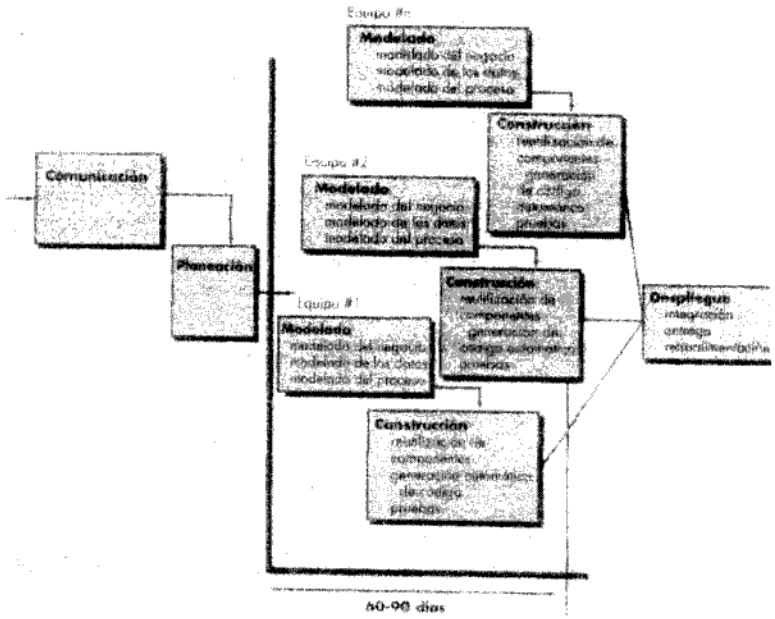
La **comunicación** trabaja para entender el problema de negocios y las características de información que debe incluir el software.

La **planeación** es esencial porque varios equipos de software trabajan en paralelo sobre diferentes funciones del sistema.

El **modelado** incluye tres grandes fases: modelado de negocios, de datos y del proceso y establece representaciones del diseño que sirven como base para la actividad de construcción del modelo DRA.

La **construcción** resalta el empleo de componentes de software existente y la aplicación de la generación automática de código.

Por último, el **despliegue** establece una base para las iteraciones subsecuentes, si éstas son necesarias.



VENTAJAS Y DESVENTAJAS

Ventajas

- Si una aplicación de negocios se puede modular de forma que cada gran función pueda completarse en menos de tres meses.
- Cada gran función se puede abordar mediante un equipo de DRA por separado, para después integrarlas y formar un todo como todos los modelos de proceso.

Desventaja

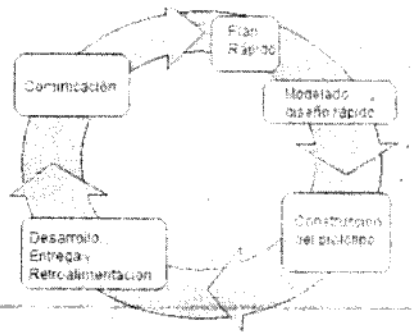
- Para proyectos grandes, pero escalables, el DRA necesita suficientes recursos humanos para crear el número correcto de equipos DRA.
- Si los desarrolladores y clientes no se comprometen con las actividades rápidas necesarias para completar el sistema en un marco de tiempo muy breve, los proyectos de DRA fallarán.
- El DRA sería inapropiado cuando los riesgos técnicos son altos (por ejemplo, cuando una aplicación nueva aplica muchas nuevas tecnologías).

2.5.2.- Modelo de Proceso Evolutivo

2.5.2.1.- Construcción de Prototipo

En el Modelo de Construcción de Prototipo el cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. El responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humana – máquina.

El modelo contiene un conjunto de pasos, en donde inicia con la **comunicación**, el ingeniero de software y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es necesaria más definición. Para luego desarrollar un **plan de rapidez,** una iteración de construcción de prototipos y se presenta el **modelado** (en la forma de un diseño rápido). El **diseño rápido** se centra en una representación de aquellos aspectos del software que serán visibles para el usuario final. Este conduce a la **construcción de un prototipo**. Después, el prototipo **desarrollado** es **entregado** para ser evaluado por el cliente/usuario y con la **retroalimentación** se redefinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo se desarrollador entienda mejor lo que se debe hacer el sistema.



MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

VENTAJAS Y DESVENTAJAS

VENTAJAS:

- No modifica el flujo del ciclo de vida.
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.

- Reduce costos y aumenta la probabilidad de éxito.
- Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.

DESVENTAJAS

- El cliente ve funcionando lo que para él es la primera versión del prototipo que ha sido construido con "chicle y cable para embalaje", y puede decepcionarse al indicarle que el sistema aun no ha sido construido.
- El desarrollador puede caer en la tentación de aumentar el prototipo para construir el sistema final sin tener en cuenta las obligaciones de calidad y de mantenimiento que tiene con el cliente.

2.5.2.2.- El Modelo en Espiral

Este es un modelo de proceso de software evolutivo, el cual enlaza la naturaleza iterativa de la construcción de prototipos, pero conservado aquellas propiedades del modelo en cascada.

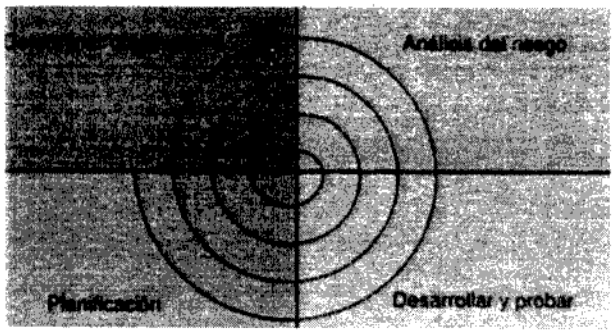
El modelo en espiral fue desarrollado por Boehm, quien lo describe así:

El modelo de desarrollo en espiral es un generador de modelo de proceso guiado por el riesgo que se emplea para conducir sistemas intensivos de ingeniería de software concurrente y a la vez con muchos usuarios.

Se caracteriza principalmente por:

- Un enfoque cíclico para el crecimiento incremental del grado de definición e implementación de un sistema, mientras que disminuye su grado de riesgo.
- Conjunto de puntos de fijación para asegurar el compromiso del usuario con soluciones de sistema que sean factibles y mutuamente satisfactorias.

Funcionamiento del modelo Espiral



En cada vuelta tomamos en cuenta:

- Determinar Objetivos: Que necesidad debe envolver el programa.
- Análisis del Riesgo: Los distinto métodos de alcanzar los objetivos de manera exitosa, a través de diferentes puntos como son:
    - Características: experiencia del personal, exigencias a efectuar.
    - Formas de gestión del programa.
    - Riesgo tomado con cada alternativa.
- Desarrollar y Probar: Programar y probar el programa.
- Planificarán los siguientes pasos y se volverá a empezar la espiral. La espiral tiene una forma de caracola y se dice que mantiene dos dimensiones la radial y la angular:
    - Angular=Avance del proyecto Software, dentro de un ciclo.
    - Radial=Aumento del coste del proyecto, ya que con cada nueva iteración se pasa más tiempo desarrollando.

Este sistema es muy utilizado en proyectos largos como pueden ser la creación de un Sistema Operativo y que necesitan constantes cambios.

Al ser un modelo de Ciclo de Vida orientado al riesgo, se dice que; uno de los aspectos fundamentales de su éxito, radica en que el equipo que lo aplique sea capaz de detectar y catalogar correctamente dicho riesgo.