**CAPITULO 1**

**SOFTWARE E INGENIERIA DEL SOFTWARE**

La invención de la tecnología puede tener efectos profundos e inesperados en otras tecnologías con las que en apariencia no tiene ninguna relación. A este fenómeno de lo denomina “la ley de las consecuencias imprevistas”.

Por ejemplo nadie podía haber predicho que el software se convertiría en una tecnología indispensable en los negocios, la ciencia y la tecnología; tampoco que permitiría la creación de nuevas tecnologías.

**El papel evolutivo del software**

El software tiene el papel de un producto y un vehículo mediante el cual se entrega un producto.

Sin importar el lugar en que resida el software, este es un transformador de información; realiza la producción, el manejo, la adquisición, la modificación, el despliegue o la transmisión de la información.

En su papel de vehículo para la entrega de un producto, el software actúa como la base para el control de la computadora (sistemas operativos), la comunicación de información (redes), y la creación y el control de otros programas (utilerías de software y ambientes).

El software entrega el producto más importante de nuestro tiempo: información.

**Software**

El software es el elemento lógico de un sistema. Tiene características muy diferentes a las del hardware, como ser:

* Se desarrolla o construye, no se manufactura.

Existen similitudes entre el desarrollo del software y la manufactura del hardware, pero son diferentes. En ambas, la alta calidad se alcanza por medio del buen diseño, pero la fase de manufactura del hardware puede incluir problemas de calidad inexistentes en el software.

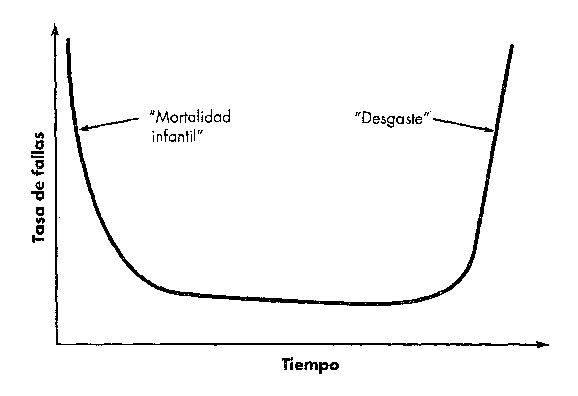
Ambas actividades requieren la construcción de un “producto”, pero los enfoques son diferentes.

Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no se pueden manejar como si fueran proyectos de manufactura.

* No se desgasta pero se deteriora.

Curva de la bañera para el hardware:

Para el hardware la tasa de fallas se muestra en función del tiempo, al inicio de su vida tiene un alto número de fallas, los defectos se corrigen y la tasa de fallas baja hasta un nivel estable. Conforme pasa el tiempo la tasa de fallas se elevan de nuevo debido al desgaste.

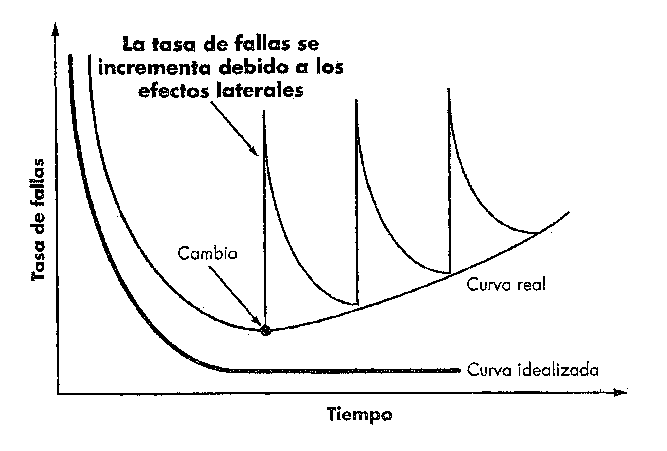


La curva de la tasa de fallas para el software debería tener la forma de Curva idealizada. Donde los defectos sin descubrir causan tasas de falla altas en las primeras etapas de vida de un programa. Los errores se corrigen y la curva se aplana. Esta curva es una simplificación burda del modelo de fallas real para el software. La implicación es clara: el software no se desgasta, pero si se deteriora.

Curva real:

El software durante su vida experimenta cambios, conforme éstos ocurren se presenta la posibilidad de tener errores, lo que ocasiona que la curva tenga un pico.

Antes de que la curva pueda regresar a su estado original con una tasa de fallas estable, se requiere otro cambio, lo que ocasiona que la curva tenga otro pico, el nivel de fallas mínimo se comienza a elevar, el software se deteriora debido a los cambios. Cualquier falla del software implica un error en el diseño.



* La mayoría aún se construye a medida. (el hardware se sustituye, el software se rediseña)

En el mundo el hardware, la reutilización de componentes ya una parte natural. En el ámbito del software, dicha actividad apenas se ha comenzado a extender.

Un componente de software se debe diseñar e implementar de forma que pueda utilizarse en muchos programas diferentes.

Los componentes reutilizables modernos encapsulan tanto los datos como el proceso que se aplica a estos, lo que permite crear aplicaciones nuevas a partir de partes reutilizables.

Ej.: Las interfaces actuales con el usuario se construyen con componentes reutilizables que permiten la creación de ventanas graficas, menús desplegables y mecanismos de interacción.

**La naturaleza Cambiante del Software**

En la actualidad existen siete grandes categorías de software:

**1- Software de sistemas:** Colección de programas escritos para servir a otros programas. Algunos programas procesan estructuras de información (compiladores, editores) otras procesan datos indeterminados (sistema operativo). Se caracteriza por la interacción muy intensa con el hardware; utilización por múltiples usuarios.

**2- Software de aplicación**: Programas independientes que resuelven la necesidad de negocios específica. Facilitan las operaciones de negocios o toma de decisiones técnicas o de gestión. También, se utiliza para controlar las funciones de negocios en tiempo real (procesamientos de transacciones de venta).

**3- Software científico y de ingeniería:** Abarca desde la astronomía hasta la vulcanología, desde la biología molecular hasta la manufactura automatizada. Se utiliza para el diseño asistido por computadora, la simulación de sistemas y otras aplicaciones interactivas, aunque éstas han comenzado a tomar características de software en tiempo real e incluso de software de sistemas.

**4- Software empotrado:** Reside dentro de la memoria de solo lectura del sistema y con él se implementan y controlan características y funciones para el usuario final y el sistema mismo.

Ej.: Puede desempeñar funciones como el control del teclado de un horno de microondas hasta funciones digitales de un automóvil, como el control de combustible.

**5- Software de línea de productos:** Diseñado para proporcionar una capacidad especifica y utilización de muchos clientes diferentes. Se puede enfocar en un nicho de mercado limitado, como productos para el control de inventarios, o dirigirse hacia mercados masivos como procesadores de texto, hojas de cálculo, manejo de bases de datos, etc.

**6- Aplicaciones basadas en Web:** En su forma más simple son apenas un poco más que un conjunto de archivos de hipertexto ligados que presentan información mediante texto y algunas gráficas.

Proporcionan características, funciones de cómputo y contenidos independientes al usuario final, así como también integración con bases de datos corporativas y aplicaciones de negocios.

**7- Software de inteligencia artificial:** Incluyen robótica, los sistemas expertos, el reconocimiento de patrones (imagen y voz), las redes neuronales artificiales, la comprobación de teoremas y los juegos de computadora.

Se espera que la generación actual facilite la tarea de los ingenieros de software del futuro, Sin embargo, han aparecido retos nuevos:

**Computación ubicua:** El reto es desarrollar software de sistema y de aplicación que permita que dispositivos pequeños, computadoras personales y sistemas de empresa se comuniquen a través de redes.

**Alimentación de la red:** El reto es crear aplicaciones que beneficien a mercados de usuarios finales específicos alrededor del mundo.

**Fuente abierta:** El reto es construir un código fuente que sea descriptivo en sí mismo, y desarrollar técnicas que permitan tanto a los clientes como a los diseñadores conocer los cambios realizados y la forma en que se manifiestan dentro del software.

**La “nueva economía”:** El reto es construir aplicaciones que faciliten la comunicación y la distribución de productos.

**Software Heredado**

Fueron desarrollados hace décadas y modificados en forma continua para cumplir los requerimientos de los cambios en los negocios y en las plataformas de cómputo.

* Se perciben como costosos en su mantenimiento y riesgosos en su evolución.
* Persisten como el soporte de las funciones centrales de negocios y son indispensables para las empresas.
* Lo caracterizan su longevidad y el ser crítico para los negocios.
* Son de la poca calidad.
* Tienen diseños imposibles de extender, código complicado, documentación escasa o inexistente, casos de prueba y resultados que nunca fueron archivados, un historial de cambio manejado con pobreza, etc.

Conforme pasa el tiempo, los sistemas heredados evolucionan por una o más de las razones siguientes:

1. El software debe adaptarse para satisfacer las necesidades de los nuevos ambientes o las nuevas tecnologías de cómputo.
2. El software debe mejorarse para implementar los nuevos requerimientos de los negocios.
3. El software debe extenderse para hacerlo operable con sistemas y bases de datos más modernos.
4. El software debe rediseñarse para hacerlo viable dentro de un ambiente de red.

Es por eso que se debe someter a una reingeniería de modo que conserve su viabilidad en el futuro.

**Evolución del Software**

El software evoluciona a través del tiempo. El cambio conduce este proceso, y se presenta:

* cuando se corrigen errores.
* cuando el software se adapta a un nuevo ambiente.
* cuando el cliente solicita características o funciones nuevas.
* cuando la aplicación experimenta una reingeniería para proporcionar beneficios en un contexto moderno.

Mitos

**Mitos de la administración**

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

*SI se está atrasado en el itinerario es posible contratar más programadores para así terminar a tiempo.*

*Si decido subcontratar el proyecto de software a un tercero, puedo relajarme y dejar que esa compañía la construya.*

**Mitos del Cliente**

*Un enunciado general de los objetivos es suficiente para comenzar a escribir programas, los detalles se pueden afinar después.*

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

**Mitos del Desarrollador**

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

*Mientras el programa no se esté ejecutando, no existe forma de evaluar su calidad*

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

*La ingeniera del software obligara a emprender la creación de una documentación voluminosa e innecesaria y de manera invariable tornara más lento el proceso.*

**CAPITULO 2**

**EL PROCESO DEL SOFTWARE**

**¿Qué es?**

Consiste en una seria de pasos predecibles, una especie de mapa que ayuda a crear un resultado de alta calidad y a tiempo. Ese mapa se llama: “Proceso de software”

**¿Quien lo hace?**

Los ingenieros de software y sus jefes adaptan el proceso a sus necesidades, además el cliente forma parte de ese proceso en cuanto a la prueba y refinamiento.

**¿Por qué es importante?**

Ofrece estabilidad, control, y organización a una actividad.

**¿Cuáles son los pasos?**

El proceso depende directamente del software que se está construyendo, de ahí que haya varias maneras de hacerlo.

**¿Cuál es el producto obtenido?**

Los productos obtenidos son los programas, documentos y datos que se producen como consecuencia de las actividades y tareas definidas por el proceso.

**¿Cómo puedo estar seguro que lo he hecho correctamente?**

Por medio de diferentes medios o mecanismos de evaluación, de manera tal que se identifique la madurez de dicho proceso.

**Ingeniería del Software**

La ingeniería del software debe estar sustentada por el compromiso con la calidad, la cual está conformada por:

**Proceso:** Define el marco de trabajo que debe establecerse para la entrega efectiva de la tecnología de la ingeniería de software. Forma la base para el control de la gestión de los proyectos y establece el contexto en el cual se aplican los métodos técnicos.

**Métodos:** Proporcionan los “como” técnicos para construir el software. Abarcan amplias tareas que incluyen la comunicación, el análisis, el modelado de diseño, la construcción del programa y las pruebas.

**Herramientas:** Proporcionan el soporte automático para el proceso y los métodos. Cuando se integran de forma que la información que cree una de ellas pueda usarla otra, se dice que se ha establecido un sistema para el soporte del desarrollo del software, llamada Software Asistida por computadora.

**Marco de Trabajo para el Proceso**

El marco de trabajo establece la base para un proceso de software completo al identificar un número pequeño de actividades del marco de trabajo aplicables a todos los proyectos de software sin importar su tamaño o complejidad. Cada actividad tiene un conjunto de tareas relacionadas que produce un producto de trabajo.

**Actividades:**

**Comunicación:** Implica una intensa colaboración y comunicación con los clientes e investigación de requisitos y otras actividades relacionadas.

**Planeación:** Describe las tareas técnicas que deben realizarse, los riesgos y los recursos que serán requeridos.

**Modelado:** creación de modelos que permiten al desarrollador y al cliente entender mejor los requisitos del software y el diseño que conviene.

**Construcción:** Generación de código y realización de pruebas para control.

**Despliegue:** Se entrega al cliente, quien evalúa el producto recibido y proporciona información basada en su evaluación.

Actividades “Sombrillas”: Se aplican durante el proceso de software y se enfocan en la gestión, el rastreo y el control del proyecto.

**Seguimiento y control del proyecto de software:** Evalúa el proceso comparándolo con el plan del proyecto y así tomar las acciones necesarias.

**Gestión del riesgo:** evalúa los riesgos que pudieran afectar los resultados del proyecto.

**Aseguramiento de la calidad del software:** Conduce las actividades para refinar el software.

**Revisión de técnicas formales:** Evalúa los productos del trabajo de la ingeniería antes que se propaguen los errores que contengan.

**Medición:** Define y recolecta mediciones del proceso, el proyecto y el producto para poder ayudar al equipo.

**Gestión de la configuración del Soft:** Maneja los efectos del cambio que haya.

**Gestión de la Reutilización:** Establece mecanismos para la creación de componentes reutilizables.

**Preparación y producción del producto de trabajo:** Actividades requeridas para crear productos del trabajo como modelos, documentos etc.

La adaptación del proceso de software es esencial para el éxito del proyecto.

**Modelo prescriptivo:** Intenta mejorar la calidad del sistema, hacer que los proyectos sean más manejables, que las fechas de entrega y los costos sean más predecibles y guiar a los equipos de ingenieros mientras realizan el trabajo.

**Modelo Ágil:** Resalta la manejabilidad y adaptabilidad. Son apropiados para muchos tipos de proyectos y son útiles de manera particular cuando se desarrollan aplicaciones de red.

**Integración del Modelo de Capacidad de madurez ( IMCM)**

Representa un modelo de proceso en dos formas diferentes: Como un modelo continuo y como un modelo discreto. El modelo continuo se basa en:

**Niveles de Capacidad:**

Nivel 0: incompleto. La gestión de requisitos aun no se ha realizado o no alcanza todas las metas.

Nivel 1: Realizado. Todas las metas específicas del proceso están establecidas y han sido satisfactorias.

Nivel 2: Administrado. El trabajo asociado con el área de proceso se ajusta a una política organizacional definida con la cual el equipo que ejecuta el trabajo tiene acceso a todo el material de trabajo para realizar su labor y los clientes están implicados de manera activa en el área de proceso.

Nivel 3: Definido. Adaptado al conjunto de procesos estándar de la organización, de acuerdo con las políticas de adaptación de esta misma, y contribuye a la información de los productos.

Nivel 4: Administrado en forma cuantitativa. El área de proceso se controla y mejora mediante mediciones y evaluaciones.

Nivel 5: Mejorado. Se adapta y mejora mediante el uso de medios cuantitativos estadísticos para conocer las necesidades cambiantes del cliente y mejorar continuamente dicha área de proceso.

**Metas especificas:** Establecen las características que deben existir para que las actividades implicadas sean efectivas.

M1: Establecer estimaciones.

Alcance del proyecto.

Estimaciones para los atributos del producto y las tareas.

Definir el ciclo de vida del proyecto.

Determinar esfuerzo y costo.

M2: Desarrollar un plan de proyecto.

Establecer el presupuesto y el programa.

Identificar los riesgos.

Planear la gestión de los datos.

Los recursos del proyecto.

Los conocimientos y habilidades que se requieren.

La participación con el cliente.

Establecer el plan de proyecto.

M3: Comprometerte con la planeación.

Revisar los planes que afectan el proyecto.

Conciliar el trabajo y los niveles de recursos.

Comprometerse con la planeación.

**Metas genéricas:**

M1: Alcanzar las metas especificas.

M2: Institucionalizar un proceso de gestión.

Establecer una política organizacional.

Planear el proceso.

Proporcionar recursos.

Asignar responsabilidades.

Capacitar gente.

Identificar y hacer participar a los clientes.

Monitorear y controlar el proceso.

M3: Institucionalizar un proceso definido.

Establecer un proceso definido

Recolectar información de la mejoría.

M4: Institucionalizar un proceso manejado en forma cuantitativa.

Establecer objetivos cuantitativos para el proceso.

Estabilizar el desempleo del subproceso.

M5 Institucionalizar un proceso de mejoramiento.

Asegurar la mejora continua del proceso.

Corregir las causas de los problemas desde la raiz

El modelo discreto define las mismas áreas, metas y prácticas pero con la diferencia que establece 5 niveles de madurez, en vez de 5 niveles de capacidad. (Optimización, gestionado de modo cuantitativo, definido, gestionado, ejecutado

Patrones del Proceso

Ofrece una plantilla, un método consistente para describir una característica importante del proceso. Mediante la combinación de patrones un equipo de software puede construir un proceso que satisfaga lo mejor posible las necesidades del proyecto.

Patrón[ambler]

**Nombre del patrón:** Nombre significativo.

**Propósito:** Se describe brevemente el objetivo del patrón.

**Tipo:** de tarea /de escenario / de fase

**Contexto inicial:** Se describen las condiciones en las cuales se aplica el patrón.

**Problema:** Se describe el problema que debe resolverse.

**Solución:** es la implementación del patrón.

**Contexto resultante:** Condiciones que habrá una vez que el patrón haya sido implementado.

**Patrones relacionados:** Lista de patrones relacionados con este.

**Evaluación del Proceso**

El proceso mismo debe evaluarse para asegurarse de que se cumpla una serie de criterios básicos del proceso que han demostrado ser esenciales para una ingeniería de software exitosa. Se han propuesto enfoques para la evaluación del proceso:

* Método de evaluación de la IMCM estándar.
* Apreciación basada en el CMM para el mejoramiento del proceso interno.
* El estándar SPICE.
* El ISO 9001:2000 para software.

**Proceso de Software Personal (PSP):** Resalta la medida personal del producto de trabajo que se produce y la calidad resultante del producto de trabajo. EL modelo PSP define 5 actividades:

* Planeación: Selecciona requisitos y con estos desarrolla el tamaño y estimación de los recursos.
* Diseño de alto nivel:
* Revisión del diseño de alto Nivel ( a través de métodos formales de verificación)
* Desarrollo: Se genera, revisa y prueba el código.
* Análisis de resultados: Se determina la efectividad del proceso.

EL PSP destaca la necesidad de registrar y analizar los tipos de errores que se comenten para desarrollar estrategias encaminadas a eliminarlos.

**Proceso de software en equipo (PSE):** Su meta es construir un equipo de proyecto “autodirigido” que se organice para producir un software de alta calidad. Sus objetivos son:

* Motivar a sus equipos y ayudarlos a sostener un alto desempeño
* Acelerar el mejoramiento del proceso de software
* Construir equipos autodirigidos que planeen y tengan un seguimiento de su trabajo, establezcan metas y posean sus procesos y planes.

Los escritos del PSE definen elementos del proceso del equipo y actividades que ocurren dentro del proceso.

**CAPITULO 3**

**Modelos Prescriptivos de Proceso**

**¿Qué es?**

Los modelos prescriptivos de proceso definen un conjunto distinto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad.

**¿Quién lo hace?**

Los ingenieros de software y sus gerentes adaptan un modelo prescriptivo de proceso a sus necesidades y después lo siguen. Además, la gente que ha solicitado el software tiene un papel por desempeñar conforme se ejecuta el modelo de software.

**¿Por qué es importante?**

Porque proporciona estabilidad, control, y organización a una actividad que si no se controla puede volverse caótica.

**¿Cuáles son los pasos?**

El proceso conduce a un equipo de software a través de un conjunto de actividades del marco de trabajo que se organizan en un flujo de proceso, el cual puede ser lineal, incremental o evolutivo. La terminología y los detalles de cada modelo varían entre sí.

**¿Cuál es el producto obtenido?**

Desde el punto de vista de un ingeniero de software, los productos de trabajo son los programas, documentos y datos que se producen como consecuencia de las actividades y tareas que define el proceso.

**¿Cómo puedo estar seguro que lo hice correctamente?**

Existen mecanismos para la evaluación del proceso de software que permite determinar la “madurez” de sus respectivos procesos. Sin embargo, los mejores indicadores de la eficacia del proceso que se utiliza son la calidad, el tiempo de entrega y la viabilidad a largo plazo del producto que se construye.

**Un modelo descriptivo del proceso llena el marco de trabajo con una serie de tareas conjuntas explicitas para las acciones de la ingeniería del software**

**Modelo en Cascada**

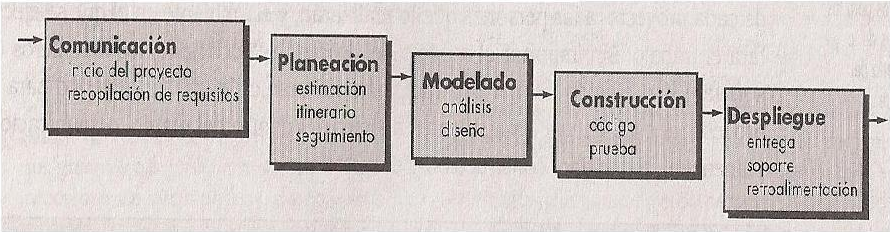
El modelo en cascada es el paradigma más antiguo para la ingeniería de software.

Es aplicable en ocasiones en que los requisitos de un problema se entienden de una manera razonable.

Sugiere un enfoque sistemático secuencial hacia el desarrollo de software.

Se utiliza en situaciones de adaptación o mejoras bien definidas a un sistema existente

**Etapas**

****

***Desventajas:***

* Es muy raro que los proyectos reales sigan el flujo secuencial que propone el modelo.
* Es difícil para el cliente establecer todos los requisitos de manera explícita.
* El cliente debe tener paciencia. Una versión que funcione de los programas estará disponible cuando el proyecto este muy avanzado.
* Conduce a “estados de Bloqueo” en donde miembros del equipo deben esperara a otros para terminar tareas pendientes

**Modelos de Proceso Incrementales**

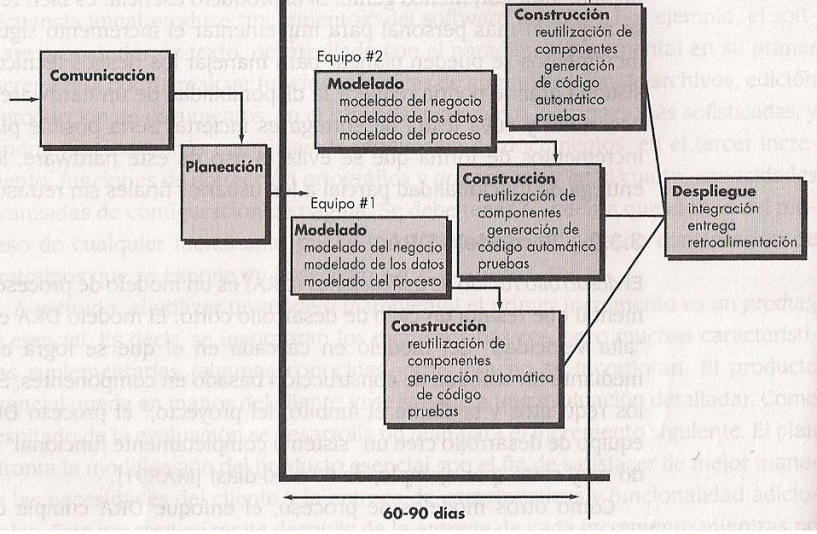
Un modelo incremental se puede aplicar cuando haya una necesidad imperiosa de proporcionar de manera rápida un conjunto limitado de funcionalidad para el usuario. Después se refinará y expandirá en entregas posteriores del software.

**Modelo Incremental**

* Combina elementos del modelo en cascada aplicado en forma iterativa.
* Aplica secuencias lineales en forma escalonada conforme avanza el tiempo en el calendario.
* Cada secuencia lineal produce incrementos de software.
* El primer incremento es un *producto esencial.*
* Una vez evaluado el primer incremento se elabora un plan para el incremento siguiente.
* El segundo incremento tendrá características y funcionalidad adicionales y así sucesivamente hasta entregar el producto final.
* Se enfoca en la entrega de un producto operacional en cada incremento.
* Los primeros incrementos son versiones “incompletas” del producto final pero proporcionan al usuario la funcionalidad que necesita y una plataforma para evaluarlo.
* Este modelo es útil cuando el personal necesario para una implementación completa no está disponible.
* Si el producto esencial es bien recibido, se agrega (si se requiere) más personal para implementar el incremento siguiente.

**Modelo DRA (Desarrollo rápido de aplicaciones)**

* El Desarrollo Rápido de Aplicaciones es un modelo de proceso de software incremental que resalta un ciclo de desarrollo corto.
* 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.
* El modelo DRA permite que un equipo de desarrollo cree un “*sistema completamente funcional*” dentro de un periodo muy corto (por ejemplo de 30 a 90 días).
* Cada gran función se puede abordar mediante un equipo de DRA por separado para después integrarlas y formar un todo.



**Comunicación:** Entender el problema de negocios.  
**Planeación:** Varios equipos trabajan en paralelo sobre dif. Funciones.  
**Modelado:** Incluye, de datos, de negocios y del proceso.  
**Construcción:** Se emplean componentes existentes y la generación automática de código.  
**Despliegue:** Establece una base para las iteraciones subsecuentes.

**Inconvenientes:**

* Para proyectos grandes el DRA necesita suficientes recursos humanos para crear el número correcto de equipos.
* Si desarrolladores no se comprometen a terminar en tiempo y forma sus actividades, el proyecto DRA fallara.
* Si un sistema no se puede modular en forma apropiada, la construcción de los componentes necesarios para el DRA será problemática.
* El DRA sería inapropiado cuando los riesgos técnicos son altos.

**Modelos De Proceso Evolutivos**

El software evoluciona con el tiempo así como también los requisitos de los negocios y productos a menudo cambian conforme se realiza el desarrollo.

Los modelos evolutivos son iterativos; los caracteriza la forma en que permiten que los ingenieros de software desarrollen versiones cada vez más completas del software. Además, producen una versión completa en forma incremental con cada iteración**.**

**Construcción de Prototipos**

A menudo un cliente define un conjunto de objetivos generales pero no identifica los requisitos detallados de entrada, procesamiento o salida. En estas, y en muchas otras situaciones un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.

Ayuda al Ing. En Sistemas y al cliente a entender de mejor manera cual será el resultado de la construcción cuando los requisitos estén satisfechos

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 mas definición. Entonces se plantea con rapidez una iteración de construcción de prototipos y se presenta el modelado en forma de un diseño rápido.

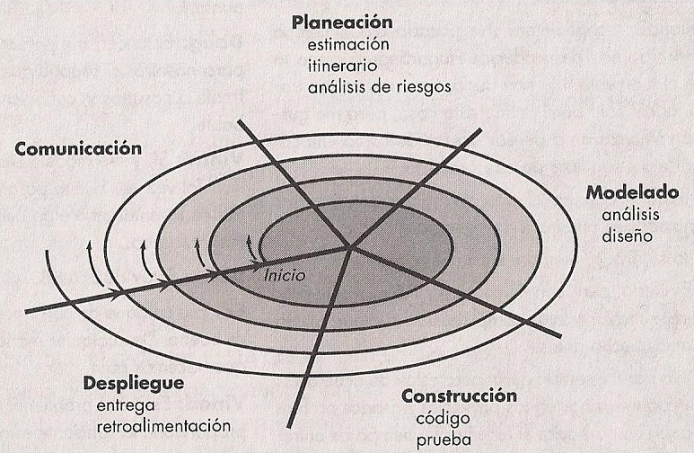
El cliente evalúa el prototipo y con la retroalimentación se refinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente.

Inconvenientes:

* El cliente ve al prototipo como una versión en funcionamiento del software sin saber que es de baja calidad e incompleto (el cliente quiere amoldar el prototipo).
* A veces los desarrolladores utilizan sistemas operativos o lenguajes inapropiados para desarrollar el prototipo rápidamente.

**Modelo en Espiral**

* Se puede adoptar y aplicarlo a través del ciclo de vida completo de una aplicación, desde el desarrollo del concepto hasta el mantenimiento
* El modelo en espiral es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada.
* Los costos y el itinerario se ajustan con base en la retroalimentación derivada de la relación con el cliente después de la entrega.
* Es adaptable y puede aplicarse a lo largo de la vida del software.
* El riesgo es un factor considerado en cada vuelta a lo largo del espiral. Pero si se aplica correctamente, debe reducirlos antes de que se vuelvan problemáticos.
* El software se desarrolla en una serie de entregas evolutivas.
* Durante las primeras iteraciones la entrega tal vez sea un documento o un prototipo. Durante las últimas se producen versiones cada vez más completas del sistema desarrollado.

****

**Modelo de desarrollo concurrente**

* Representa en forma esquemática una serie de actividades del marco de trabajo, acciones y tareas de la ingeniería del software y sus estados asociados.
* Define una serie de eventos que dispararan transiciones de estado a estado para cada una de las actividades, acciones o tareas.

**Modelos Especializados de Proceso**

Los modelos especializados de proceso toman muchas de las características de los procesos convencionales, pero tienden a utilizarse en enfoques de ingeniería de software bien definidos.

**Desarrollo basado en componentes**

Se basa en la utilización de componentes de software, los cuales proporcionan funcionalidad dirigida con interfaces bien definidas que permiten que el componente se integre en el software.

Incorpora muchas de las características del modelo en espiral, es evolutivo por naturaleza y exige un enfoque iterativo para la creación de software. Sin embargo, el modelo configura aplicaciones a partir de componentes de software empaquetados en forma previa.

Se comienza con la identificación de los componentes candidatos. Estos componentes se pueden diseñar como módulos de software convencionales o como clases o paquetes de clases orientados a objetos. Mediante la implementación de un enfoque evolutivo, el modelo basado en componentes incorpora los siguientes pasos:

* Los productos basados en componentes disponibles se investigan y evalúan para el dominio de aplicación en cuestión.
* Se consideran los aspectos de integración de componentes.
* Se diseña una arquitectura de software para adaptar los componentes.
* Los componentes se integran en la arquitectura.
* Se realizan pruebas detalladas para asegurar una funcionalidad apropiada.

El modelo de desarrollo basado en componentes conduce a la reutilización del software, la cual proporciona beneficios a los ingenieros de software.

**Modelos de métodos formales**

Permite que un ingeniero de software especifique, desarrolle y verifique un sistema de computadora al aplicar una notación matemática rigurosa.

Proporcionan un mecanismo para eliminar problemas difíciles de superar con otros paradigmas de la ingeniería del software.

La ambigüedad, el estado incompleto y la inconsistencia se descubren y se corrigen con mayor facilidad a través de la aplicación de un análisis matemático.

A pesar de la utilidad de los métodos formales, se pueden destacar ciertos percances como:

* El desarrollo de métodos formales es muy caro y consume mucho tiempo.
* Se requiere una capacitación detallada, debido a que pocos responsables del desarrollo de software cuentan con los antecedentes necesarios para aplicar métodos formales.
* Es difícil la utilización de estos modelos como mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos.

**Desarrollo del software orientado a aspectos**

Cuando los intereses se relacionan con múltiples funciones y características se denominan intereses generales. Los requerimientos de aspectos definen estos intereses generales que ejercen un impacto a través de la arquitectura del software.

El desarrollo de software orientado a aspectos es un paradigma de la ingeniería del software que proporciona un proceso y enfoque metodológico para definir, especificar, diseñar y construir aspectos.

El DSOA define aspectos que expresan los intereses del cliente, los cuales afectan muchas funciones, características e información del sistema.

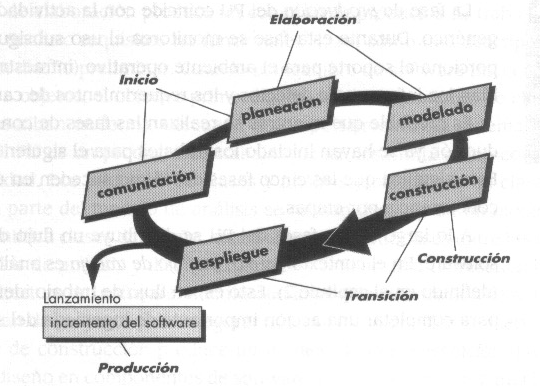
**Proceso Unificado**

El proceso unificado es un intento encaminado a reunir los mejores rasgos y características de modelos de procesos de software, pero los caracteriza de manera que implementa mucho de los mejores principios del desarrollo ágil del software.

El PU enfatiza el importante papel de la arquitectura de software, y ayuda al arquitecto a enfocarse en las metas correctas, como el entendimiento, el ajuste a los cambios futuros y la reutilización.

También sugiere un flujo de proceso iterativo e incremental y que proporciona el sentido evolutivo esencial en el desarrollo del software moderno.

En los años siguientes, Jacobson, Rumbaugh y Booch desarrollaron el *proceso unificado*, un marco de trabajo para la ingeniería del software orientada a objetos, mediante la utilización del UML.



La fase de ***inicio*** del PU abarca la comunicación con el cliente y las actividades de planeación. Al colaborar con los clientes y usuarios finales se identifican los requisitos de negocios para el software, se propone una arquitectura aproximada para el sistema y se desarrolla un plan para la naturaleza iterativa e incremental del sistema subsiguiente.

La fase de ***elaboración*** abarca la comunicación con el cliente y las actividades de modelado del modelo genérico del proceso. La elaboración refina y expande los casos de uso preliminares que se desarrollaron como parte de la fase de inicio. Además, expande la representación arquitectónica para incluir 5 visiones diferentes del software: el modelo de caso de uso, el modelo de análisis, el modelo de diseño, el modelo de implementación y el modelo de despliegue.

La fase de ***construcción*** del PU desarrolla o adquiere los componentes del software que harán que cada caso de uso sea operativo para los usuarios finales. Lograr esto requiere que los modelos de análisis y diseño iniciados durante la fase de elaboración se completen para reflejar la versión final del incremento del software. Todas las características y funciones necesarias del incremento del software (por ej. la entrega) se implementan en código fuente.

La fase de ***transición*** del PU abarca las últimas etapas de la actividad genérica de construcción y la primera parte de la actividad genérica de despliegue. El software se entrega a los usuarios finales para realizar pruebas beta, y la retroalimentación del usuario reporta tanto defectos como cambios necesarios. Al final de la fase de transición, el incremento de software se convierte en un lanzamiento de software utilizable.

La fase de ***producción*** del PU coincide con la actividad de despliegue del proceso genérico. Durante esta fase se monitorea el uso subsiguiente del software, se proporciona el soporte para el ambiente operativo (infraestructura), y se reciben y evalúan los informes de defectos y los requerimientos de cambios.

A lo largo de las fases del PU se distribuye un flujo de trabajo de ingeniería del software. En el contexto del PU, un *flujo de trabajo* es análogo a un conjunto de tareas. Esto es, un flujo de trabajo identifica las tareas necesarias para completar una acción importante de ingeniería del software y los productos de trabajo que se producen como consecuencia de la realización exitosa de tareas. Se debe destacar que no todas las tareas identificadas para un flujo de trabajo del PU se realizan para cualquier proyecto de software.

**Producto de trabajo del proceso unificado**

Desde el punto de vista del ingeniero de software, el producto de trabajo más importante generado durante la etapa de inicio es el modelo de casos de uso.

La fase de elaboración produce un conjunto de productos de trabajo que elabora requisitos, así como una descripción arquitectónica y un diseño preliminar.

La fase de construcción produce un modelo de implementación que traduce las clases de diseño en componentes de software que se construirán para ejecutar el sistema. Un modelo de prueba describe las pruebas empleadas para asegurar que los casos de uso se reflejen de manera apropiada en el software que se ha construido.

La fase de transición entrega el incremento del software y evalúa los productos de trabajo elaborados durante la etapa en que los usuarios finales trabajan con el software. Se produce la retroalimentación y los requerimientos cualitativos de cambio.

**CAPITULO 25**

GESTION DEL RIESGO

Estrategias de riesgo Reactivas y Proactivas

**Estrategia Reactiva:** EL equipo de software no hace nada acerca de los riesgos hasta que algo sale mal. Entonces el equipo se precipita en la acción con la finalidad de corregir el problema. Esto se llama el modo bombero. Cuando esto falla, la gestión de crisis asume el control y el proyecto esta en peligro.

**Estrategia Proactiva:** Comienza mucho antes de que se inicie el trabajo técnico. Se identifican los riesgos potenciales, se valoran su probabilidad e impacto, y se les clasifica según su importancia. Luego, el equipo establece un plan para gestionar el riesgo.

El riesgo siempre involucra dos características: Incertidumbre y pérdida. Es importante cuantificar el grado de estas características, esto se logra a través de diferentes categorías de riesgos.

**Riesgos del Proyecto:** amenazan el proyecto, si se vuelven reales es probable que la calendarización se altere y que los costos aumenten. Identifican potenciales problemas en presupuesto, tiempo, personal, recursos.

**Riesgos Técnicos:** Amenazan la calidad y actualidad del software que se producirá, si se vuelven reales es probable que la implementación se torna difícil o imposible. Identifican potenciales problemas en diseño, implementación, interfaz.

**Riesgos de Negocios:** Amenazan la viabilidad del software que se construirá. Candidatos para los 5 mayores riesgos de negocios son:

1. **Riesgo de Mercado**
2. **Riesgo de Estrategia**
3. **Riesgo de Ventas**
4. **Riesgo Administrativo**
5. **Riesgo Presupuestal**

**Riesgos Conocidos:** Son aquellos susceptibles de descubrirse después de una evaluación del plan del proyecto.

**Riesgos Predecibles:** se extrapolan de la experiencia.

**Riesgos Impredecibles:** Son los difíciles de identificar con antelación.

**Identificación de Riesgos**

**Riesgos genéricos:** Son una amenaza potencial para todo el proyecto de software

**Riesgos Específicos del producto:** Pueden identificarse solo aquellas personas con conocimiento de la tecnología, personal y el entorno especifico que se construirá.

Un método para identificar riesgos consiste en crear una lista de verificación de riesgos.

* Tamaño del producto
* Impacto en el negocio (restricciones del mercado, gerencia)
* Características del cliente
* Definición del proceso
* Entorno de desarrollo (disponibilidad y calidad de las herramientas)
* Tecnología que construir (complejidad del sistema)
* Tamaño y experiencia de la plantilla de personal

**Componentes y Controladores del riesgo**

**Riesgo de Desempeño:** Incertidumbre de que el producto satisfaga los requisitos y se juste al uso pretendido.

**Riesgo de Costo:** Incertidumbre de que se mantenga el presupuesto del proyecto.

**Riesgo de Soporte:** Incertidumbre de que el software resultante será fácil de corregir, adaptar y mejorar.

**Riesgo de Calendarización:** Incertidumbre de que se mantenga la calendarización del proyecto.

**Proyección del Riesgo**

El planificador del proyecto, junto con otros gestores y personal técnico, realizan cuatro pasos en la proyección del riesgo.

1. Establecimiento de una escala que refleje la posibilidad percibida de un riesgo
2. Delineado del riesgo
3. Estimación del impacto del riesgo en el proyecto y el producto
4. Tomar nota de la precisión global de la proyección del riesgo de modo que no hay malas interpretaciones

**Evaluación del Impacto del riesgo**

Tres factores afectan las consecuencias que son probables si un riesgo ocurre:

**Naturaleza:** Indica los problemas que son probables si ocurre.

**Ambito:** Combina la severidad con su distribución global.

**Tiempo:** El tiempo de un riesgo considera cuando y durante que periodo se sentirá el impacto. En la mayoría de los casos un gestor de proyecto tal vez quiera que ocurran las malas noticias tan pronto como sea posible.

Pasos para determinar las consecuencias globales de un Riesgo:

1. Terminar el valor promedio de la probabilidad de que ocurra para cada componente de riesgo.
2. Empleando la figura de la (Diapositiva N°9), determinar el impacto para cada componente, con base en los criterios mostrados.
3. Completar la Tabla de Riesgos y analizar los resultados como se describe en las secciones precedentes

La exposición al riesgo global, ER, se determina mediante la siguiente relación => ER= P x C

**Reducción, Supervisión y Gestión del Riesgo**

Una estrategia eficaz debe considerar tres temas:

* Evitar del Riesgo
* Supervisar el Riesgo
* Gestionar el Riesgo y los Planes de Contingencia

EL riesgo se reduce si el gestor del proyecto desarrolla una estrategia para reducir la movilidad. Algunos pasos serian: determinar las causas de la movilidad, reducirlas y controlarlas antes de que comience el proyecto. Definir estándares de documentación. Llevar a cabo revisiones.

**El Plan RSGR (reducción, supervisión y gestión de riesgo)**

Este plan documenta todo el trabajo realizado como parte del análisis del riesgo y el gestor lo emplea como parte del plan global del proyecto.

Algunos equipos no elaboran todo un documento sino que van documentando cada riesgo mediante una hoja de información del riesgo (HIR)

Una vez documentado el pan RSGR y que el proyecto ha comenzado se inician los pasos de:

1. Reducción: Encaminada a evitar el problema.
2. Supervisión: Es una actividad de seguimiento del proyecto con 3 objetivos:
   * 1. Valorar si los riesgos predichos de hecho ocurren
     2. Asegurar que los pasos para evitar el riesgo definidos para éste se están aplicando con propiedad
     3. Recopilar información que pueda usarse en futuros análisis de riesgo