**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**

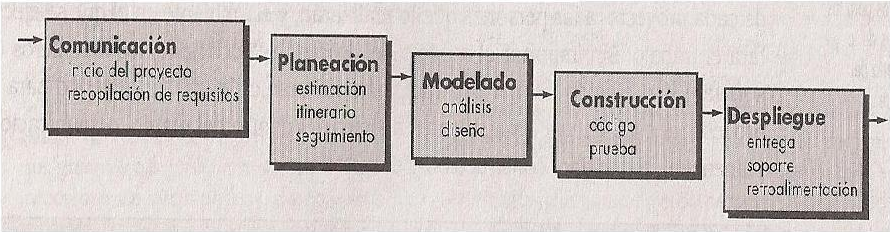
El modelo en cascada es el paradigma más antiguo para la ingeniería de software.

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**

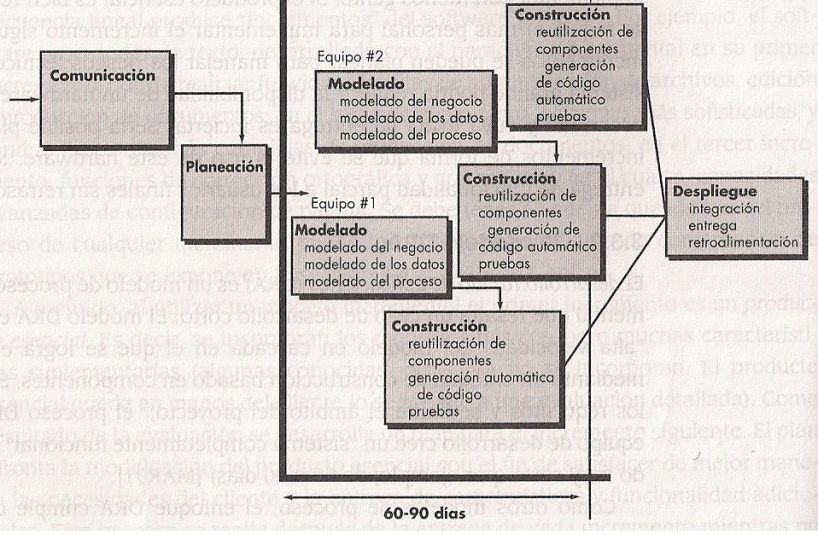
Un modelo incremental se puede aplicar cuando haya una necesidad imperiosa de proporcionar de manera rápida un conjunto limitado de funcionalidad para el usuario. Después se refinará y expandirá en entregas posteriores del software.

**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.

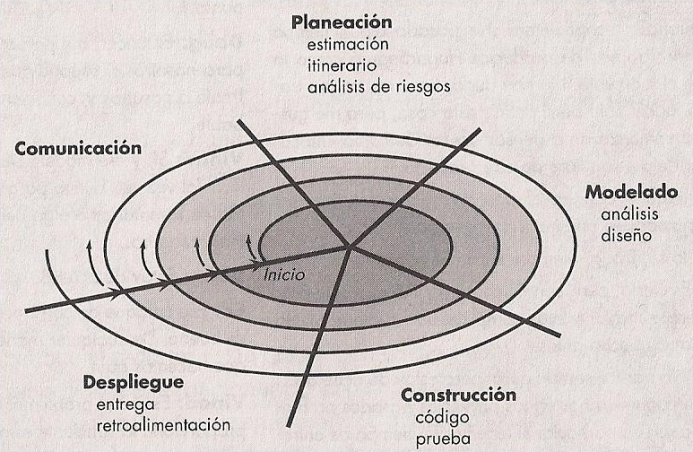
El cliente evalúa el prototipo y con la retroalimentación se refinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente.

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.

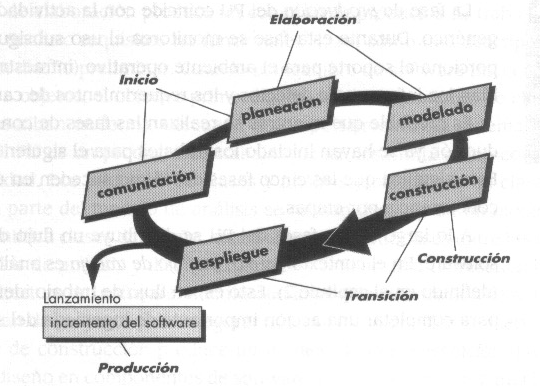
**Proceso Unificado**

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.