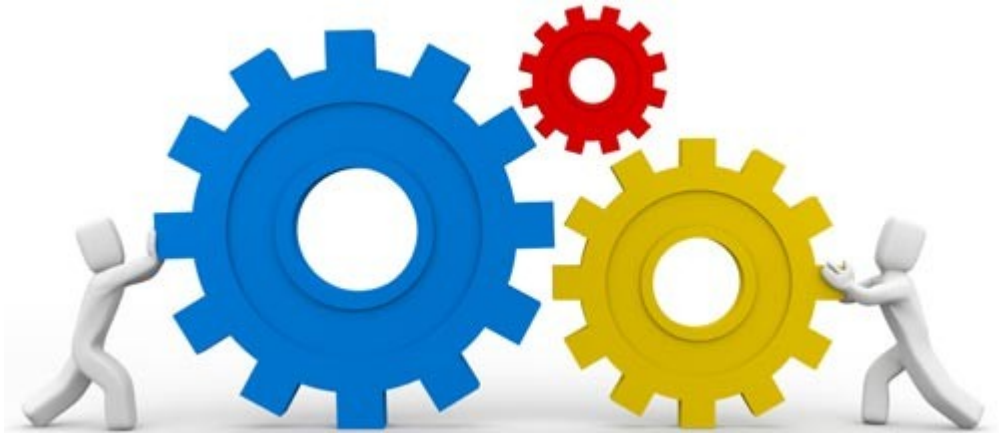
MODELOS DE PROCESO DE SOFTWARE

Proceso de Software

“Conjunto de actividades, métodos, prácticas, y transformaciones que se usan para desarrollar y mantener el software y sus productos asociados”.



Capacidad

“Describe el rango de resultados esperados que pueden ser logrados siguiendo un proceso de software”.

.

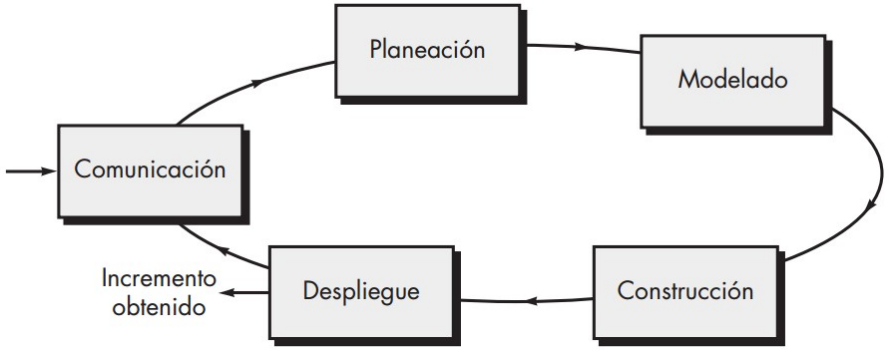
Madurez

“Ámbito o contexto en el cual un proceso específico es explícitamente definido, administrado, medido, controlado, efectivo”.

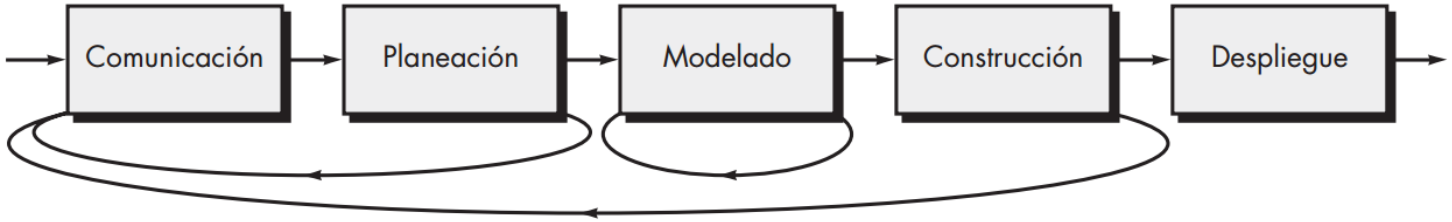
MODELOS DE PROCESO DE SOFTWARE

Una estructura general para la ingeniería de software define cinco activadas estructurales: **comunicación**, **planeación**, **modelado**, **construcción** y **despliegue**.

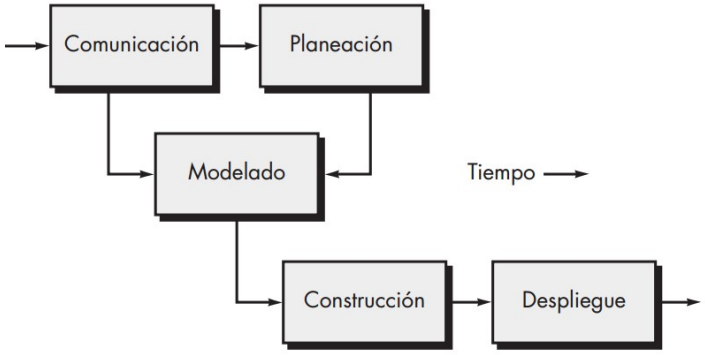
Flujo de proceso lineal



Flujo de proceso evolutivo



Flujo de proceso iterativo



Flujo de proceso paralelo

MODELOS DE PROCESO DE SOFTWARE

La existencia de un proceso del software no es garantía de que el software se entregue a tiempo, que satisfaga las necesidades de los consumidores o que tenga las características técnicas que conducirán a características de calidad de largo plazo.

En las últimas décadas se han propuesto numerosos enfoques para la evaluación y mejora de un proceso del software:

CMMI: Proporciona un modelo de cinco fases para evaluar el proceso: inicio, diagnóstico, establecimiento, actuación y aprendizaje.

CMM: proporciona una técnica de diagnóstico para evaluar la madurez relativa de una organización de software.

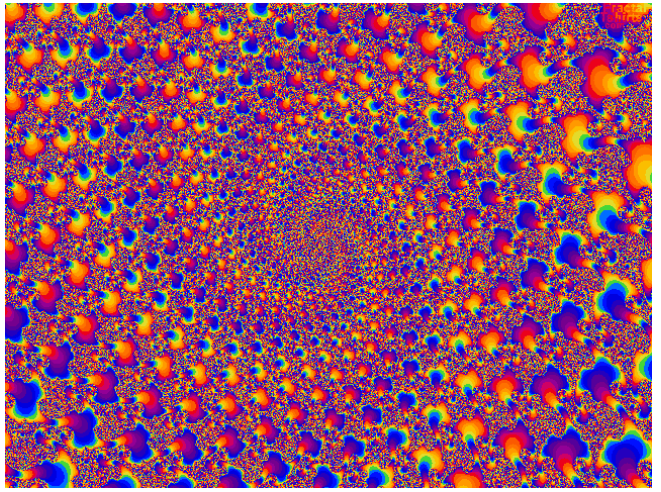
SPICE (ISO/IEC 15504): estándar que define un conjunto de requerimientos para la evaluación del proceso de software.

ISO9001:2000: estándar genérico que se aplica a cualquier organización que desee mejorar la calidad general de los productos, sistemas o servicios que proporciona.

MODELOS DE PROCESO DE SOFTWARE

Modelos de proceso prescriptivos

Los modelos de proceso prescriptivo fueron propuesto originalmente para poner en orden en el caos del desarrollo de software. Sin embargo, el trabajo de ingeniería de software y el producto que genera siguen “al borde de los caos”.



El borde los caos se define como “el estado natural entre el orden y el caos, un compromiso grande entre la estructura y la sorpresa” [Nogueira]

Modelos de proceso prescriptivos

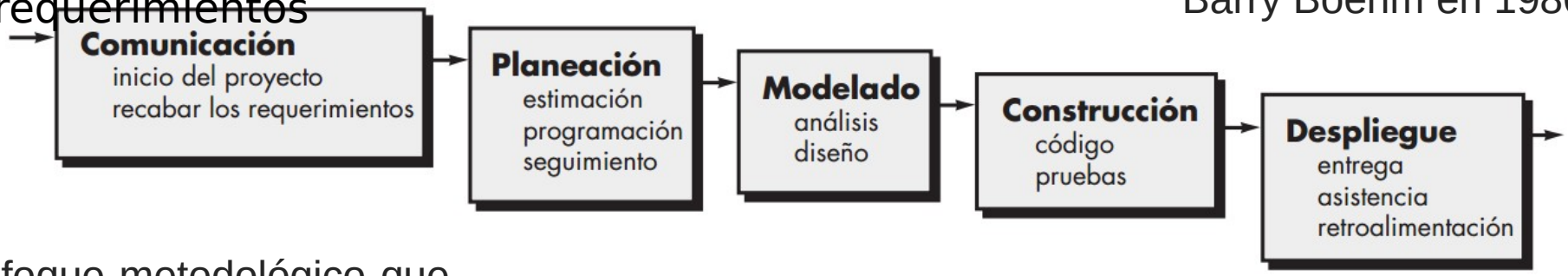
2.1. Modelo de cascada
2.2. Modelos evolutivos
2.3. Modelos incrementales
2.4. Modelos formales
2.5. Modelo basado en componentes reutilizables
2.6. Proceso unificado
2.7. Modelos ágiles
2.8. Modelos estandarizados
2.9. Otros modelos

Modelo en cascada o ciclo de vida clásico

Winston W. Royce en 1970

Barry Boehm en 1980

Comprensión correcta de los requerimientos



Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

CARACTERISTICAS	VENTAJAS	DESVENTAJAS
• Es el más utilizado. • Es una visión del proceso de desarrollo de software como una sucesión de etapas que produce productos intermedios. • Si se cambia el orden de las fases, el producto final será de inferior calidad.	• Se tiene todo bien organizado y no se mezclan las fases. • La planificación es sencilla. • La calidad del producto resultante es alta.	• Se tarda mucho tiempo en pasar por todo el ciclo. • Es difícil incorporar nuevas cosas si se quiere actualizar. • Iteraciones costosas.

Modelo en cascada o ciclo de vida

1. Análisis de los requisitos del software: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software debe comprender el ámbito de la información del software así como la función, el rendimiento y las interfaces requeridas.

2 Ingeniería y Análisis del Sistema: Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software

3. Diseño: el diseño del software se enfoca en cuatro atributos distintos del programa; la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

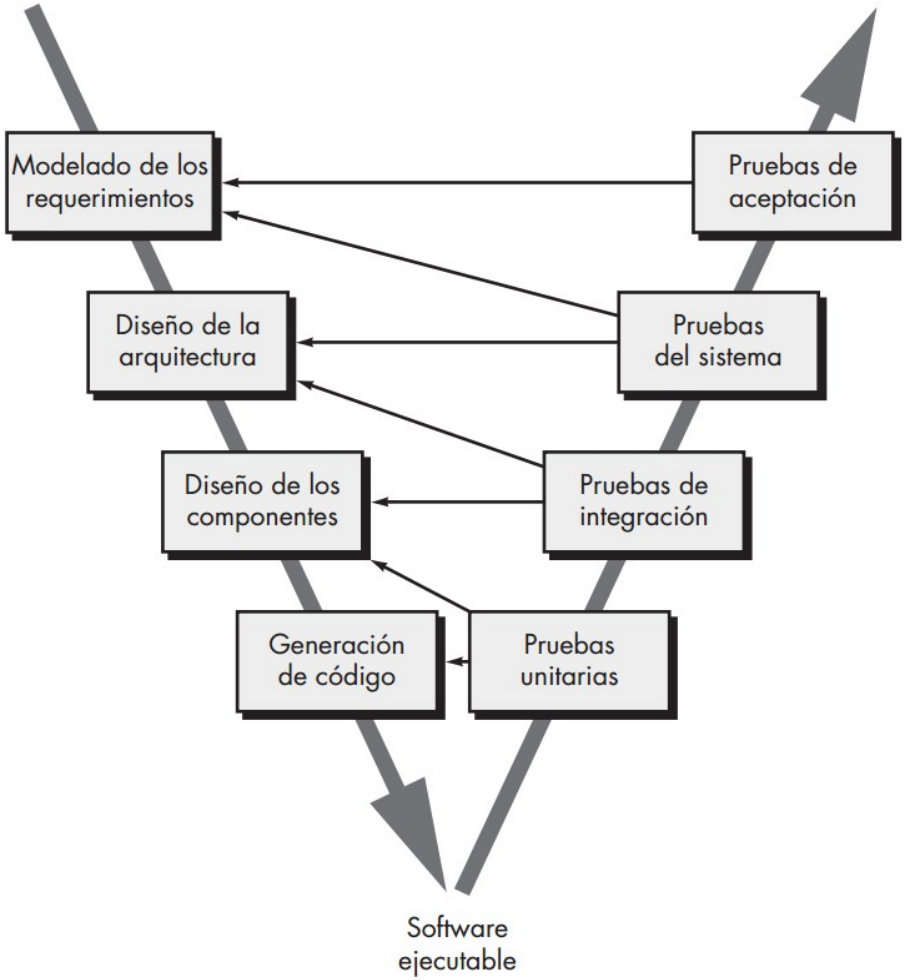
4. Codificación: el diseño debe traducirse en una forma legible para la maquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

5. Prueba: una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren

6. Mantenimiento: el software sufrirá cambios después de que se entrega al cliente.

Modelo en V

A medida que el equipo de software avanza hacia abajo desde el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema V de su solución.



Problemáticas

1. Es difícil seguir el flujo por cambios no considerados
2. No se puede conocer todos los requerimientos iniciales
3. Paciencia del cliente para ver avance

Modelo en V

VENTAJAS

•Específica bien los roles de los distintos tipos de pruebas a realizar.
•Hace explícito parte de la iteración y trabajo que hay que realizar.
•Este método involucra chequeos de cada una de las etapas del método Cascada.
•Es un método más robusto y completo que el método cascada y produce software de mayor calidad que con el modelo cascada.
•Es un modelo sencillo de y de fácil aprendizaje.
•Involucra al usuario en las pruebas.

DESVENTAJAS

•Es difícil que el cliente exponga explícitamente todos los requisitos.
•El cliente debe tener paciencia, ya que obtendrá el producto al final del ciclo de vida.
•El modelo no contempla la posibilidad de retornara etapas inmediatamente anteriores, cosa que en la realidad puede ocurrir.
•Se pierde dinero, ya que si algún proceso fue mal desarrollado, este debe ser revisado de nuevo, lo que puede traer como consecuencia un "RollBack" de todo un proceso.
•Las pruebas pueden ser caras y a veces no lo suficientemente efectivas.

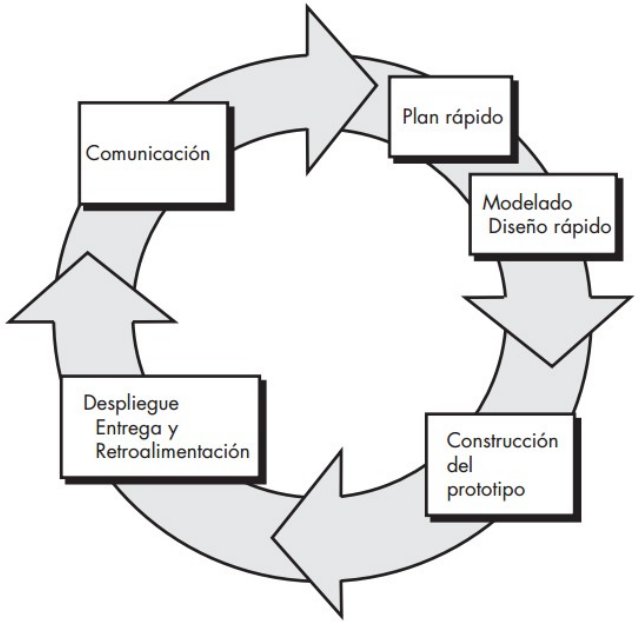
Modelos de proceso evolutivo

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada ves más completas del software.

- ✓ Plan rápido.
- ✓ Modelado, diseño rápido
- ✓ Construcción del Prototipo
- ✓ Desarrollo, entrega y retroalimentación
- ✓ Comunicación
- ✓ Entrega del desarrollo fina

Dos modelos comunes de procesos evolutivo

- Modelo de prototipos
- El modelo espiral



El ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software.

Modelos de proceso evolutivo

Dos modelos comunes de procesos evolutivo

- Modelo de prototipos
- El modelo espiral

Ventajas

•No modifica el flujo del ciclo de vida
•Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios
•Reduce costo y aumenta la probabilidad de éxito
•Exige disponer de las herramientas adecuadas
•Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
•También ofrece un mejor enfoque cuando 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 humano-máquina.

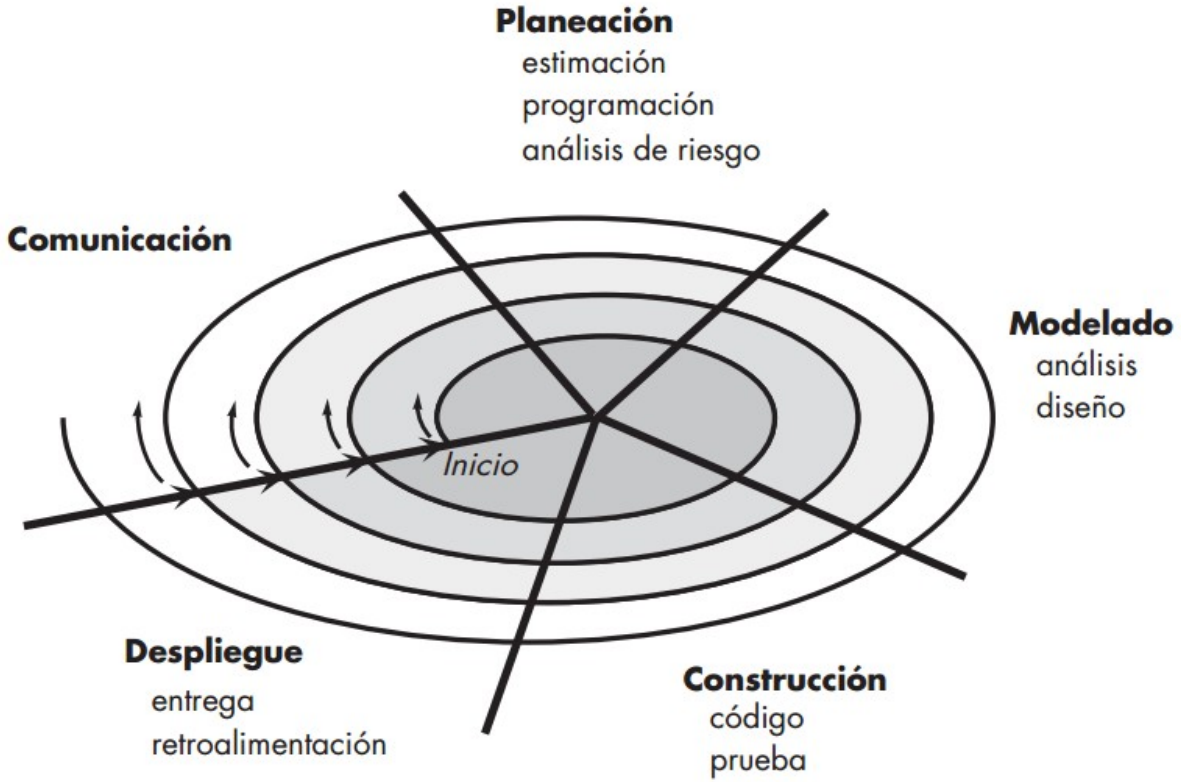
Desventajas

•Debido a que el usuario ve que el prototipo funciona piensa que este es el producto terminado y no entienden que recién se va a desarrollar el software.
•El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y mantenimiento que tiene con el cliente

Modelos de proceso evolutivo

Dos modelos comunes de procesos evolutivo

- Modelo de prototipos
- El modelo espiral



Propuesto por Barry Boehm. Es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene potencial para hacer un desarrollo rápido de versiones

TIPOS

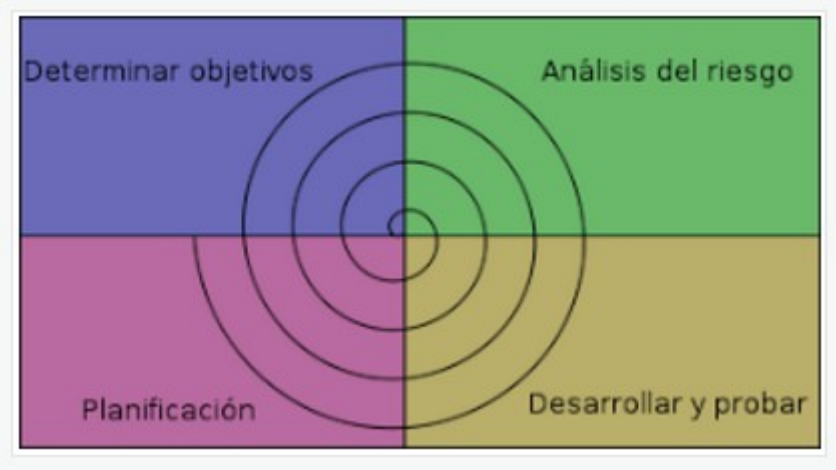
El modelo espiral tuvo varias modificaciones que son:

- Modelo Original de Boehm.
- Modelo Tipico de Seis Regiones.
- Modelo WINWIN.

Modelos de proceso evolutivo

Dos modelos comunes de procesos evolutivo

- Modelo de prototipos
- El modelo espiral



Modelos de proceso evolutivo

Dos modelos comunes de procesos evolutivo

- Modelo de prototipos
- El modelo espiral

VENTAJAS

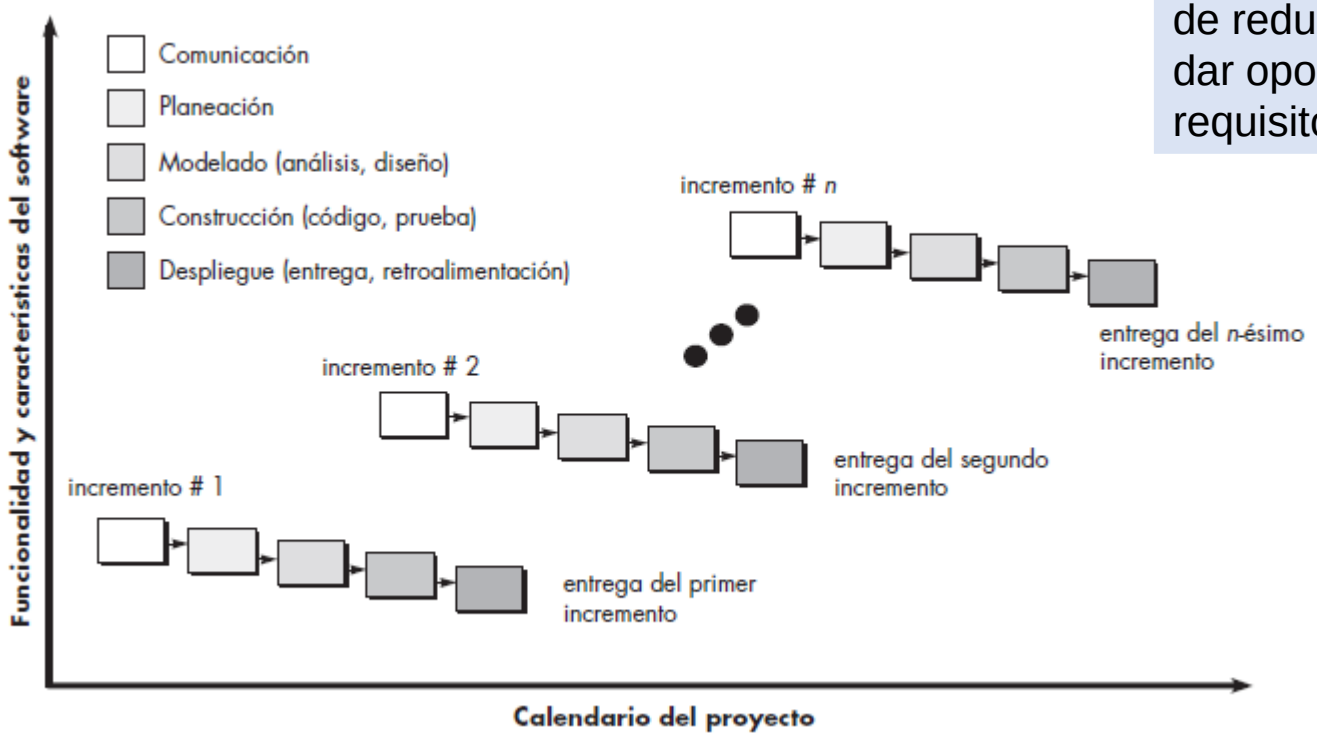
•Puede adaptarse y aplicarse a lo largo de la vida del software de computadora.
•Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los nivele evolutivos.
•El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
•El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.
•En la utilización de grandes sistemas a doblado la productividad.

DESVENTAJAS

•Resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable.
•Debido a su elevada complejidad no se aconseja utilizarlo en pequeños sistemas.
•Genera mucho tiempo en el desarrollo del sistema
•Modelo costoso
•Requiere experiencia en la identificación de riesgos

Modelos de proceso incremental

Fue propuesto por Harlan Mills en el año 1980, como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.



Este modelo se conoce también como

- ✓ Método de las comparaciones limitadas sucesivas
- ✓ Ciencia de salir del paso
- ✓ Método de atacar el problema por ramas

Modelos de proceso incremental (Secuencias lineales)

- ✓ Combina elementos del Modelo Lineal Secuencial con la filosofía interactiva de Construcción de Prototipos.
- ✓ Aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario.
- ✓ Cada secuencia lineal produce un incremento del software.
- ✓ El primer incremento generalmente es un producto esencial denominado núcleo.

Modelos de proceso incremental

Los requerimientos del usuario se priorizan y los requerimientos de prioridad más altos son incluidos en los incrementos tempranos.

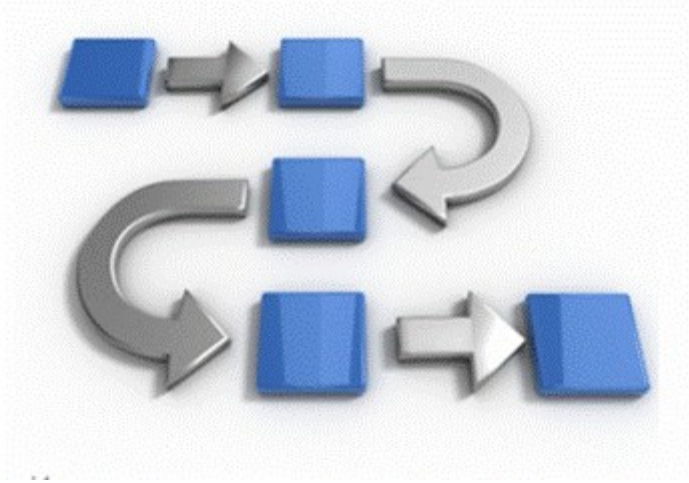
Hechos de incrementos tempranos como un prototipo, ayudan a obtener requisitos para los incrementos más tardíos.

Los usuarios no tiene que esperar.

El desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema.

Se evitan proyectos largos y se entrega “Algo de valor” a los usuarios con cierta frecuencia

El usuario se involucra más



Modelos de proceso incremental
Características

- ✓ Cada incremento agrega funcionalidad adicional o mejorada sobre el sistema
- ✓ Cada etapa debe cumplir con los requisitos de las desarrolladas
- ✓ La propuesta del modelo es diseñar sistemas que puedan entregarse por piezas.
- ✓ A partir de la evaluación se planea el siguiente incremento y así sucesivamente.
- ✓ Es interactivo por naturaleza
- ✓ Es útil cuando el personal no es suficiente para la implementación completa.
- ✓ En lugar de entrega del sistema en una sola entrega, el desarrollo y la entrega están fracturados bajo incrementos, con cada incremento que entrega parte dela funcionalidad requerida.

Modelos de proceso incremental

Ventajas

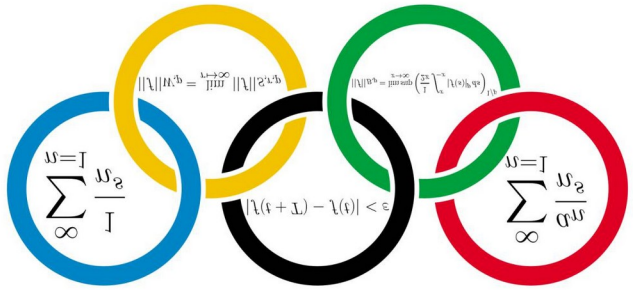
- ✓ Los clientes no tienen que esperar hasta que el sistema se entregue completamente para comenzar a hacer uso de él.
- ✓ Los clientes pueden usar los incrementos iniciales como prototipo para precisarlos requerimientos posteriores del sistema.
- ✓ Minimización del riesgo de falla en el proyecto porque los errores se van corrigiendo progresivamente.
- ✓ El resultado puede ser muy positivo.

Desventajas

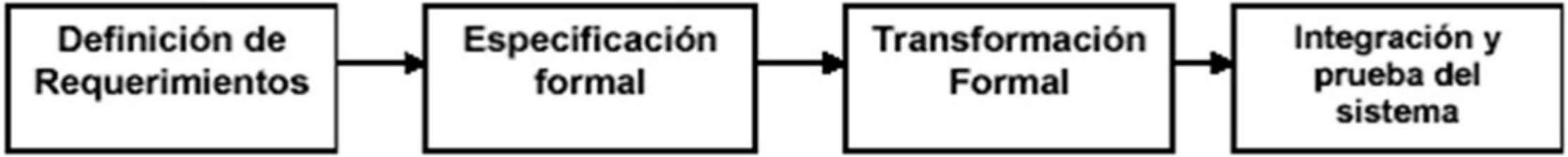
- ✓ Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo.
- ✓ Riesgos largos y complejos.
- ✓ Pueden aumentar el coste debido a las pruebas.
- ✓ Los errores en los requisitos se detectan tarde.

Modelos de métodos formal

Se usa para referirse a cualquier actividad relacionada con representaciones matemáticas del software.

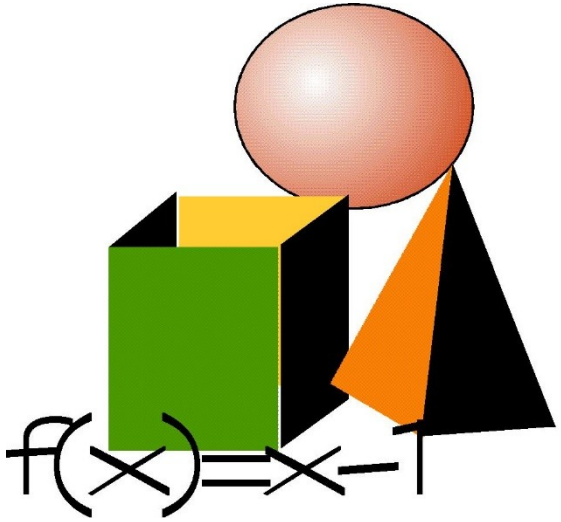


Una especificación formal del software es una especificación expresada en un lenguaje cuyo vocabulario, sintaxis y semántica están formalmente definidos.



Modelos de métodos formales

La especificaciones formales deben basarse en conceptos matemáticos cuyas propiedades se comprendan bien.



Se basa en matemática discreta, y los conceptos matemáticos provienen de la teoría de conjuntos, la lógica y el álgebra.

Modelos de métodos formales

1. Una ingeniería del software exitosa.

Claramente, esta predicción no se ha hecho realidad. Existen cuatro razones principales para esto:

Métodos formales

Otros modelos de desarrollo de software

Modelos de métodos formales

2. Cambios en el mercado

El software debe desarrollarse rápidamente, y los clientes están dispuestos a aceptar software con algunos defectos si se les entrega rápidamente. Las técnicas para el desarrollo rápido del software no funcionan de forma efectiva con las especificaciones formales.

Modelos de métodos formales

3. Ámbito limitado de los métodos formales.

Los métodos formales no son muy apropiados para la especificación de interfaces de usuario e interacciones del usuario. El componente de interfaz de usuario se ha convertido en una parte cada vez mayor de la mayoría de los sistemas, de manera que realmente solo pueden usarse métodos formales cuando se desarrollan las otras partes del sistema.

Modelos de métodos formales

4. Escalabilidad limitada de los métodos formales

Los métodos formales todavía no son muy escalables. La mayoría de los proyectos con éxito que han usado estas técnicas han estado relacionados con núcleos de sistemas críticos relativamente pequeños. A medida que los sistemas incrementan su tamaño, el tiempo y esfuerzo requerido para desarrollar una especificación formal crece de forma desproporcionada.

Modelos de métodos formales

Ventajas

- Se comprende mejor el sistema.
- La comunicación con el cliente mejora ya que se dispone de una descripción clara y no ambigua de los requisitos del usuario.
- El sistema se describe de manera más precisa.
- El sistema se asegura matemáticamente que es correcto según las especificaciones.
- Mayor calidad software respecto al cumplimiento de las especificaciones.
- Mayor productividad

Desventajas

- El desarrollo de herramientas que apoyen la aplicación de métodos formales es complicado y los programas resultantes son incómodos para los usuarios.
- Los investigadores por lo general no conocen la realidad industrial.
- Es escasa la colaboración entre la industria y el mundo académico, que en ocasiones se muestra demasiado dogmático.
- Se considera que la aplicación de métodos formales encarece los productos y ralentiza su desarrollo.

Modelos de métodos formales

El uso de métodos formales está creciendo en el área del desarrollo de sistemas críticos, en donde las propiedades emergentes del sistema tales como seguridad, fiabilidad y protección son muy importantes.

Modelos de métodos formales

El alto coste de los fallos de funcionamiento en estos sistemas implica que las compañías están dispuestas a aceptar los costes elevados iniciales de los métodos formales para asegurar que su software es tan confiable como sea posible.

Modelos de métodos formales

Los sistemas críticos en los que los métodos formales se han aplicado con éxito incluyen un sistema de información de control de tráfico aéreo, sistemas de señalización de redes ferroviarias, sistemas de naves espaciales y sistemas de control médico.

Modelos de métodos formales

La clasificación más común se realiza en base al modelo matemático subyacente en cada método, de esta manera podrían clasificarse en:

Especificaciones basadas en lógica de primer orden y teoría de conjuntos:

Especificaciones algebraicas

Especificación de comportamie

Métodos basados en Redes de Petri

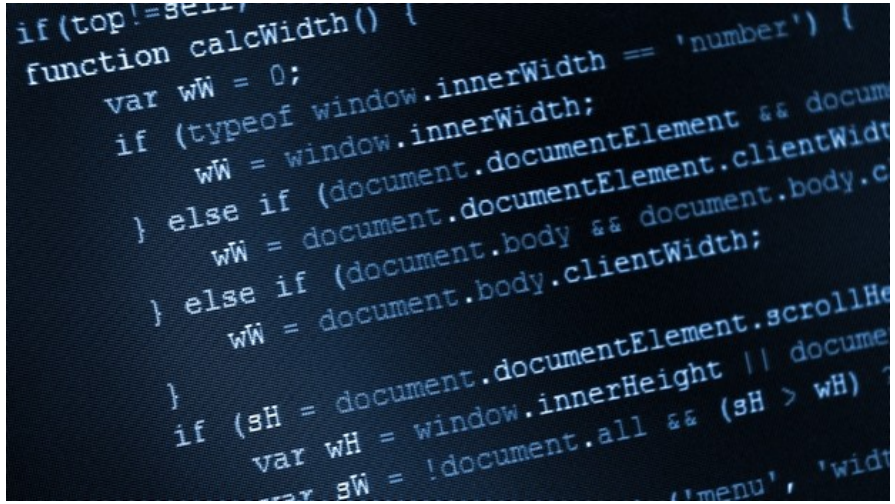
Métodos basados en lógica temporal

Modelo basado en componentes reutilizables

El desarrollo de software basado en componentes permite reutilizar piezas de ***código previamente elaborada*** *que permiten realizar diversas tareas,* ***proporcionando diversos beneficios*** *como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.*

Modelo basado en componentes

*"La **reutilización de software** es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la **especificación**, **análisis**, **diseño**, **implementación** y **pruebas** de una aplicación o sistema de software"*



*"Un **componente** es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio"*

Modelo basado en componentes reutilizables

Características de un Componente

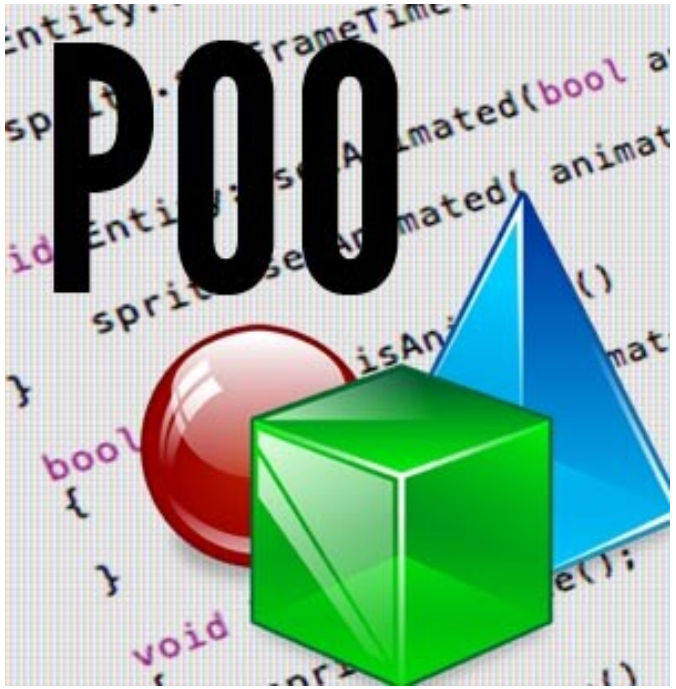
- ✓ *Identificable*
- ✓ *Auto contenido*
- ✓ *Puede ser remplazado por otro componente*
- ✓ *Con acceso solamente a través de su interfaz*
- ✓ *Sus servicios no varían*
- ✓ *Bien Documentado*
- ✓ *Reutilizado dinámicamente*
- ✓ *Independiente de la plataforma*

Modelo basado en componentes

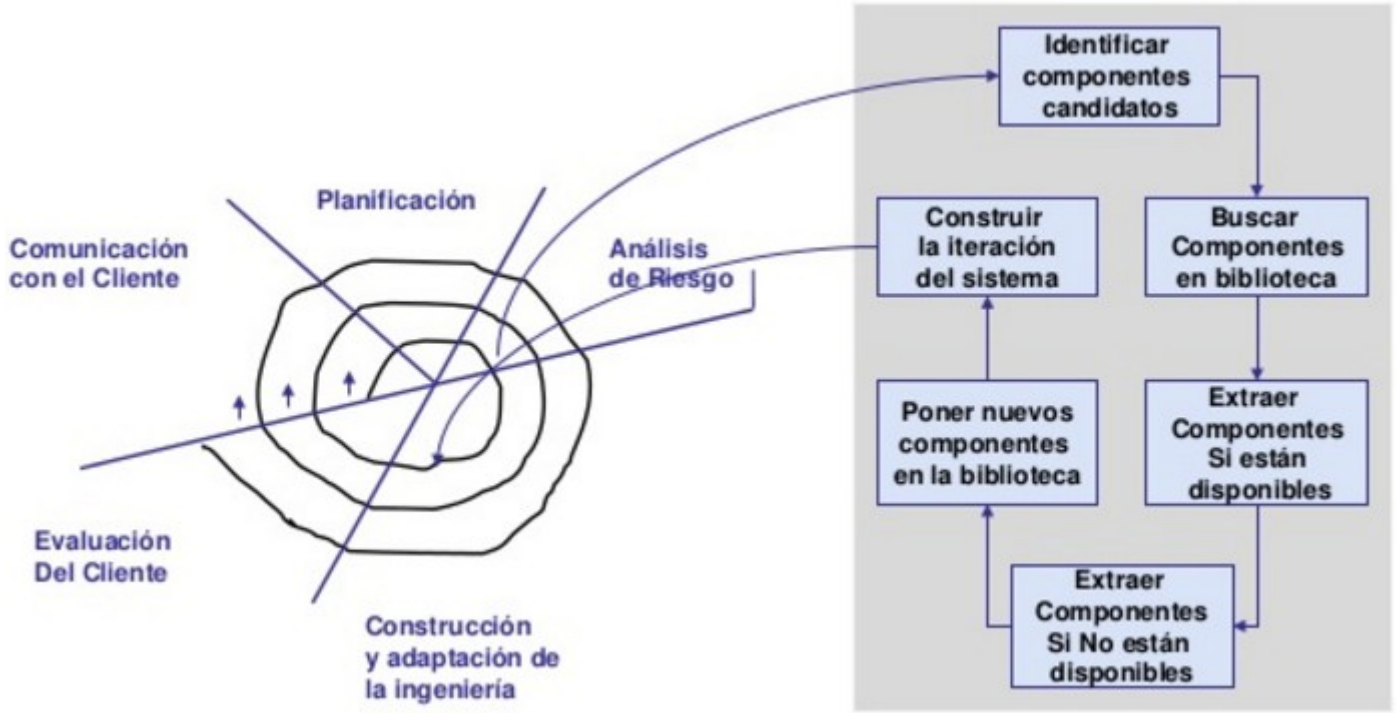
*Las **tecnologías de objetos** proporcionan el marco de trabajo técnico para un modelo de proceso basado en componentes para la ingeniería del software.*

*El paradigma orientado a objetos enfatiza la **creación de clases** que encapsulan tanto los datos como los algoritmos que se utilizan para manejar los datos.*

Si se diseñan y se implementan adecuadamente, las clases orientadas a objetos son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadora.



Modelo basado en componentes



*El modelo de **desarrollo basado en componentes** incorpora muchas de las características del modelo en espiral. Es evolutivo por naturaleza y exige un enfoque iterativo para la creación del software.*

Modelo basado en componentes reutilizables

*Según estudios de reutilización, **QSM** Associates, Inc. Informa*

- ✓ *El ensamblaje de componentes reduce el 70% de tiempo de ciclo de desarrollo del proyecto de software.*
- ✓ *Reduce el 84% de coste del proyecto de software.*
- ✓ *Y un índice de productividad del 26.2, comparado con la norma de industria del 16.9 .*

Aunque estos resultados están en función de la robustez de la biblioteca de componentes, no hay duda de que el ensamblaje de componentes proporciona ventajas significativas para los ingenieros de software.

Modelo basado en componentes

El paradigma de **ensamblar componentes** y escribir código para hacer que estos componentes funcionen se conoce como **Desarrollo de Software Basado en Componentes**.

El uso de este paradigma posee algunas **ventajas**:

1. Reutilización del software. Nos lleva a alcanzar un mayor nivel de reutilización de software.

2. Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.

Modelo basado en componentes reutilizables

1. ***Simplifica el mantenimiento del sistema*.** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.

2. ***Mayor calidad*.** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Modelo basado en componentes reutilizables

De la misma manera, el optar por **comprar componentes** de terceros en lugar de desarrollarlos, posee algunas ventajas:

1. *Ciclos de desarrollo más cortos*. La adición de una pieza dada de funcionalidad tomará días en lugar de meses ó años.

Modelo basado en componentes reutilizables

1. *Mejor ROI*. Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.

2. *Funcionalidad mejorada*. Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible.

2.8. Modelos estandarizados

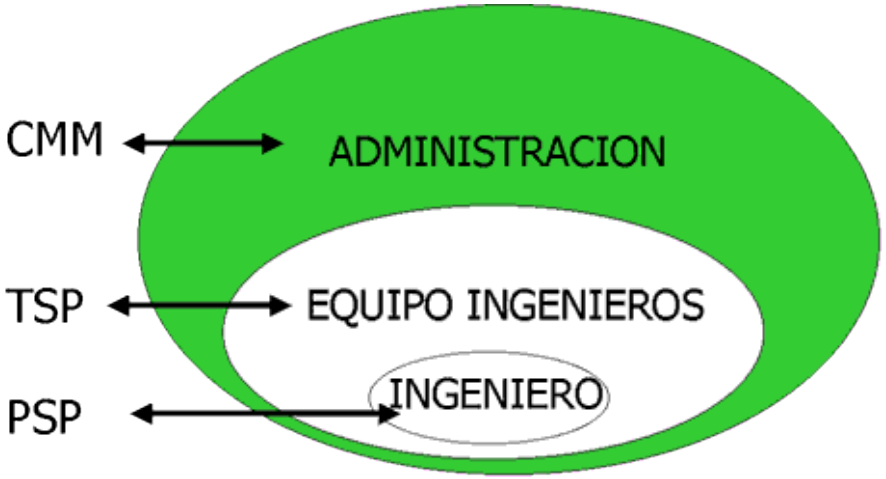
En un mundo de ***competencia global***, donde existen cambios constantes, las ***empresas de desarrollo de software*** son presionadas a alcanzar mayo eficiencia con menores costos.

Para lograr este objetivo, es necesario adoptar una forma de trabajo que permita **entender**, **controlar**, **comunicar**, **mejorar**, **predecir** y **certificar** el trabajo realizado.

Actualmente existe diversas opciones para el desarrollo de software de calidad como son **CMM**, **CMMI**, **Moprosoft**, **RUP**, **ISO**, **PSP**, **TSP**.

2.8. Modelos estandarizados

Fue definido por Watts S. Humphrey en 1995 del ***Software Engineering Institute*** (**SEI**) en la ***Carnegie Mellon University***



Cada ingeniero es diferente...

Para mejorar, se deben usar procesos personales bien definidos y cuantificados.

Para obtener productos de calidad, debemos asumir la responsabilidad personal de la calidad de sus productos.

2.8. Modelos estandarizados

Cuanto antes se **detecten** y **corrijan** los ***defectos*** menos esfuerzo será necesario.

Es mas efectivo **evitar los defectos** que detectarlos y corregirlos.

Trabajar bien es siempre la forma más rápida y económica de trabajar.

2.8. Modelos estandarizados

1) Planificar el trabajo

2) Esforzarse por cumplir la planificación

3) Esforzarse por obtener productos de la mejor calidad y esto en el contexto de un proceso de mejora continuada

2.8. Modelos estandarizados

La gestión del tiempo

Resúmenes semanales

LOC (líneas de código)

Cuaderno de trabajos

Comparar la realidad con los objetivos propuestos

2.8. Modelos estandarizados

PSP3:

PROCESO PERSONAL CICLICO

Para escalar el PSP.

Para manejo de unidades de código grandes (dividiéndolas en incrementos)

PSP2:

PROCESO DE ADMINISTRACIÓN CALIDAD PERSONAL

Manejar defectos de programación

PSP1:

PROCESO DE PLANEACION PERSONAL

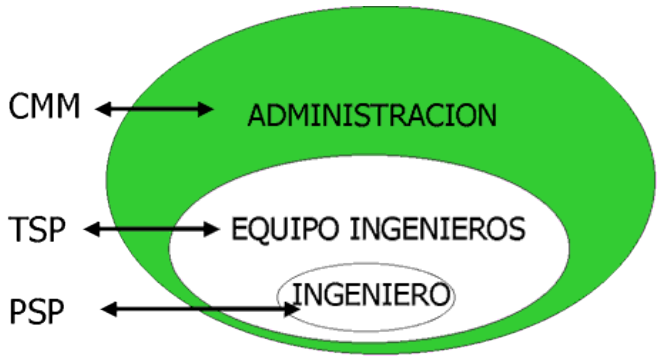
Entender relación entre tamaño de los programas y tiempo que toma desarrollarlos

PSP0:

PROCESO BASE

2.8. Modelos estandarizados

Es un ***proceso de desarrollo*** para equipos de ingenieros basado en ***CMMI***. Ayuda a conformar equipos para el desarrollo de software de calidad. TSP proporciona **directrices** para ayudar a un equipo a establecer sus **objetivos**, a **planificar** sus procesos y a revisar su trabajo con el fin de que la organización pueda establecer prácticas de ingeniería avanzadas y así obtener **productos eficientes**, **fiables** y de **calidad**.



La versión inicial del TSP fue desarrollada por ***Watts Humphrey*** en **1996**, y el primer Reporte Técnico para TSP fue publicado en el año **2000**, patrocinado por el Departamento de Defensa de los Estados Unidos.

2.8. Modelos estandarizados

TSP es una solución basada en procesos para resolver problemas de negocio, tales como: ***predicción de costo*** y ***tiempo***, ***mejora de productividad*** y ***ciclos de desarrollo*** y ***mejora de calidad de productos***.

2.8. Modelos estandarizados

Watts Humphrey desarrolló el TSP, el cual consideraba como parte importante, además de lo previsto por el PSP, los **requisitos**, las **pruebas** de integración, la **documentación** y otras actividades típicas en todo proyecto de desarrollo, de igual manera incluía actividades como los **roles** de equipo, interrelaciones dentro de la organización y la definición de un proceso de equipo para ser utilizado dentro de los procesos existentes en la organización.

2.8. Modelos estandarizados

Los Roles (responsabilidades) en los equipos en STP son:

• ***Líder del Equipo****:* Dirige al equipo, se asegura que todos reporten sus datos de los procesos y completen su trabajo tal y como se planeó. Realiza los reportes semanales del avance del equipo.
• **Gestor de desarrollo**: Guía al equipo en el diseño y desarrollo del producto.
• **Gestor de Planificación**: Apoya y guía al equipo en la planificación y seguimiento del trabajo.
• **Gestor de Calidad/Proceso**: Apoya al equipo en definir sus necesidades acerca del proceso y a establecer y administrar el plan de calidad. Genera estándares para obtener un trabajo uniforme. Modera las inspecciones y revisa cada artefacto generado.
• **Administrador de Requerimientos/Soporte**: Dirige al equipo en el desarrollo de requerimientos de software y ayuda a dar a conocer la tecnología y en las necesidades de apoyo administrativo. Administra el plan de configuración.

2.8. Modelos estandarizados

Es necesario que los ingenieros que usan TSP estén formados en PSP.

TSP está formado por dos componentes primarios que abarcan distintos aspectos del trabajo en equipo:

- Formación del equipo de trabajo
- Gestión del equipo de trabajo

2.8. Modelos estandarizados

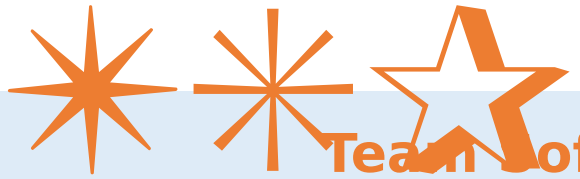
Con TSP, los equipos encuentran y reparan defectos en etapas tempranas del proceso de desarrollo, esto reduce de manera importante el tiempo de pruebas. Esto reduce de manera importante el tiempo de pruebas. Con un testing más corto, el ciclo completo se reduce.

A diferencia de otros métodos, TSP mejora el desempeño tanto de equipos como individuos, es disciplinado y ágil, provee beneficios inmediatos y medibles y acelera las iniciativas de mejora de procesos organizacionales.

2.8. Modelos estandarizados

En las fases del Ciclo TSP se planea el número de ciclos. Dentro de cada ciclo se realiza:

1. Lanzamiento
2. Estrategia
3. Plan
4. Requisitos
5. Diseño
6. Implementación
7. Pruebas
8. Postmortem



2.7. Modelos Ágiles

Surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

Es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

2.7. Modelos Ágiles

Es una metodología ágil, y como tal:

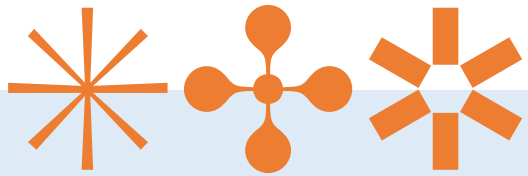
- ✓ Es un modo de desarrollo de carácter adaptable más que predictivo.
- ✓ Orientado a las personas más que a los procesos.
- ✓ Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

2.7. Modelos Ágiles

Product Backlog List
Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto.

Sprints
Un Sprint es el procedimiento de adaptación de las cambiantes variables del entorno (requerimientos, tiempo, recursos, conocimiento, tecnología). Son ciclos iterativos en los cuales se desarrolla o mejora una funcionalidad para producir nuevos incrementos.

Burn down Chart
La *burn down chart* es una gráfica mostrada públicamente que mide la cantidad de requisitos en el Backlog del proyecto pendientes al comienzo de cada Sprint.

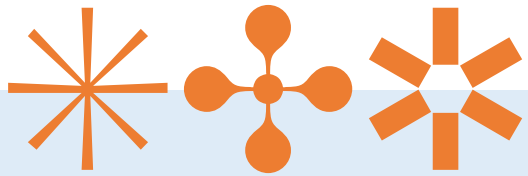


2.7. Modelos Ágiles

Sprint Backlog
Es el punto de entrada de cada Sprint. Es una lista que tiene los ítems de la Product Backlog List que van a ser implementados en el siguiente Sprint.
Los ítems son seleccionados por el Scrum Team, el Scrum Master y el Product Owner en la Sprint Planning Meeting a partir de la priorización de los ítems y los objetivos que se marcaron para ese Sprint.

Stabilization Sprints
En estos Sprints el equipo se concentra en encontrar defectos, no en agregar funcionalidad.

Scrum of Scrums o MetaScrum
Los equipos de Scrum suelen tener entre 5 y 10 personas, sin embargo está metodología ha sido aplicada en proyectos que involucran más de 600 personas.



2.7. Modelos Ágiles

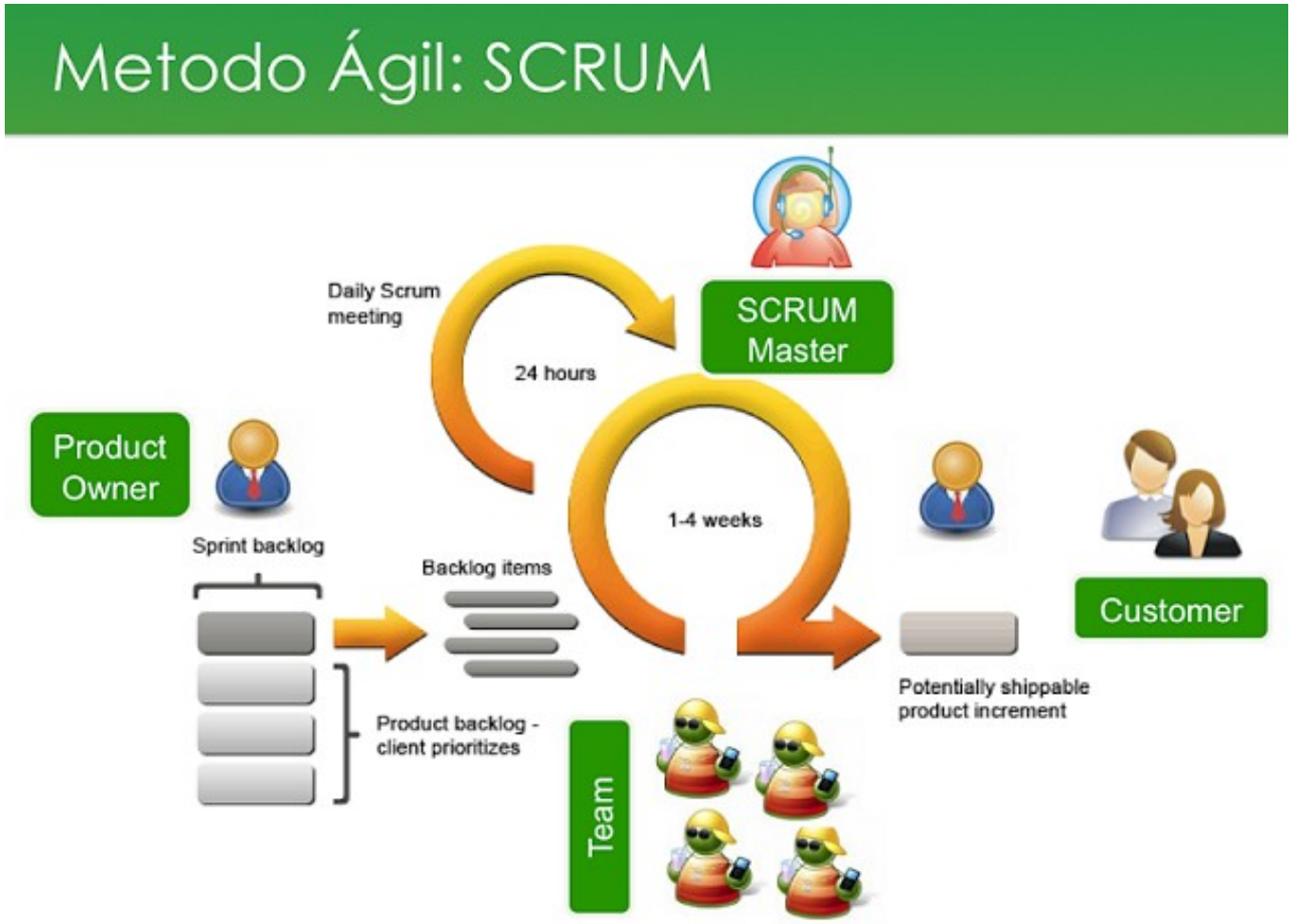
Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (normalmente de 30 días).

Cada uno de estos periodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto.
Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.

2.7. Modelos Ágiles

Características

Equipos autodirigidos
•Utiliza reglas para crear un entorno ágil de administración de proyectos
•No prescribe prácticas específicas de ingeniería
•Los requerimientos se capturan como ítems de la lista Product Backlog.
•El producto se construye en una serie de Sprints de un mes de duración.

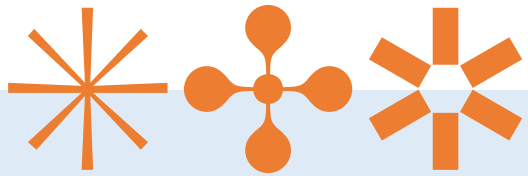


Roles y responsabilidades

Los tres primeros grupos (propietario, equipo y gestor) son los responsables del proyecto, los que según la comparación siguiente (y sin connotaciones peyorativas) serían los “cerdos”; mientras que el resto de interesados serían las gallinas.

Product Owner
El *Product Owner* representa la voz del cliente. Se asegura de que el equipo Scrum trabaja de forma adecuada desde la perspectiva del negocio.

ScrumMaster (o Facilitador)
El *Scrum* es facilitado por un *ScrumMaster*, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint.
El *ScrumMaster* no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga



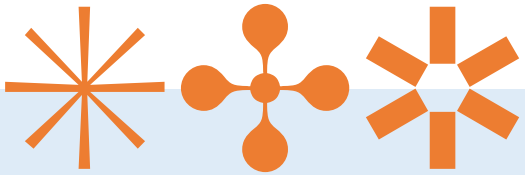
2.7. Modelos Ágiles

Equipo
El equipo tiene la responsabilidad de entregar el producto.

Usuarios
Es el destinatario final del producto. Como bien lo dice la paradoja, El árbol cae en el bosque cuando no hay nadie ¿Hace ruido? Aqui la definicion sería *Si el software no es usado ¿fue alguna vez escrito?*.

Stakeholders (Clientes, Proveedores).
Se refiere a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que lo justifica. Sólo participan directamente durante las revisiones del sprint.

Managers
Es la gente que establece el ambiente para el desarrollo del producto.



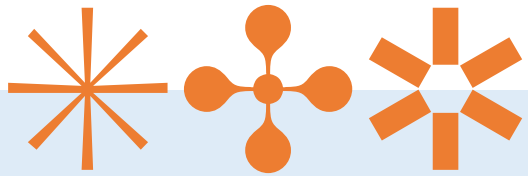
2.7. Modelos Ágiles

Reuniones en Scrum

Daily Scrum
Cada día de un sprint, se realiza la reunión sobre el estado de un proyecto. Esto se llama "daily standup".

Scrum de Scrum
- Cada día normalmente después del “Daily Scrum”
- Estas reuniones permiten a los grupos de equipos discutir su trabajo, enfocándose especialmente en áreas de solapamiento e integración.
- Asiste una persona asignada por cada equipo.

Reunión de Planificación del Sprint (Sprint Planning Meeting)
Al inicio del ciclo Sprint (cada 15 o 30 días), una “Reunión de Planificación del Sprint” se lleva a cabo.

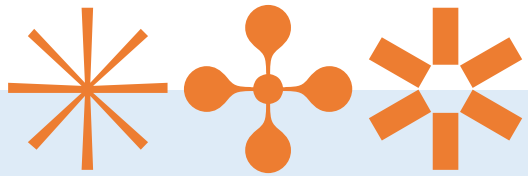


Reunión de Revisión del Sprint (Sprint Review Meeting)

- Revisar el trabajo que fue completado y no completado
- Presentar el trabajo completado a los interesados (alias “demo”)
- El trabajo incompleto no puede ser demostrado
- Cuatro horas como límite

Retrospectiva del Sprint (Sprint Retrospective)

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de cuatro horas.



2.8. Modelos estandarizados

Es el ***Modelo de Procesos*** para la Industria del Software. Un modelo para la ***mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software***.

Desarrollado por la ***Asociación Mexicana*** para la ***Calidad*** en ***Ingeniería de Software*** a través de la ***Facultad de Ciencias de la Universidad Nacional Autónoma de México*** (***UNAM***).

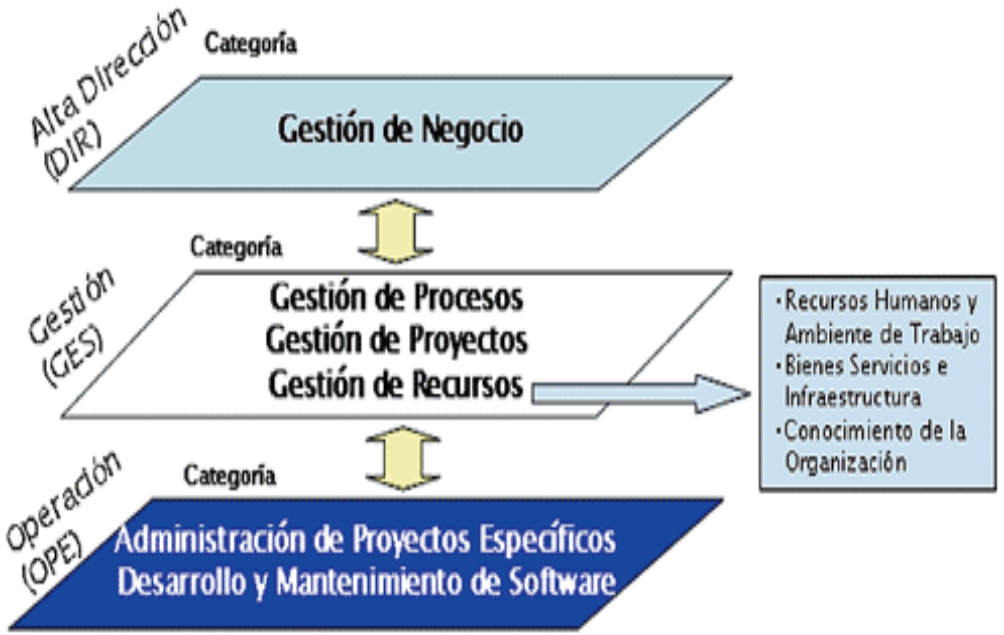
A solicitud de la ***Secretaría de Economía*** para obtener una norma mexicana que resulte apropiada a las características de tamaño de la gran mayoría de empresas mexicanas de desarrollo y mantenimiento de software.

MoProSoft

2.8. Modelos estandarizados

Hanna Oktaba es la coordinadora del MOPROSOFT. Es Doctora en Computación por la Universidad de Varsovia, es Profesor de Carrera del Departamento de Matemáticas de la UNAM.

MOPROSOFT tomó referencias como: ISO9000:2000, Nivel 2 y 3 de CMM® V.1.1, PMBOK, y SWEBOK.



MOPROSOFT es el nombre del modelo en la comunidad universitaria y profesional, y la norma técnica a la que da contenido es la **NMX-059/02-NYCE-2005** que fue declarada Norma Mexicana el 15 de agosto de 2005 con la publicación de su declaratoria en el Diario oficial de la Federación.

2.8. Modelos estandarizados

El MOPROSOT se estructura en 3 categorías:

- **Categoría de Alta Dirección (DIR)**
  Se establecen los lineamientos para los procesos de la Categoría de Gerencia y se retroalimenta con la información generada por ellos en apoyo a la estrategia de la organización.
- **Categoría de Gerencia (GER)**
  Se definen los elementos para el funcionamiento de los procesos de la Categoría de Operación en función de la estrategia de Dirección, recibe y evalúa la información generada por éstos y comunica los resultados a la Categoría de Alta Dirección.
- **Categoría de Operación (OPE)**
  Se realizan las actividades de acuerdo a los elementos proporcionados por la Categoría de Gerencia y entrega a ésta la información y productos generados.

Procesos de Dirección

Gestión-de-Negocios

Su propósito la razón de ser de la organización, sus objetivos y las condiciones para lograrlos, para lo cual es necesario considerar las necesidades de los clientes, así como evaluar los resultados para poder proponer cambios que permitan la mejora continua.

2.8. Modelos estandarizados

Procesos de Gestión

Gestión de Proyectos:

Generar proyectos que contribuyan al cumplimiento de los objetivos y estrategias de la organización

Gestión de Procesos:

Establece procesos que apoyen a las estrategias de la organización así como actividades de mejora en los mismos.

Gestión de Recursos:

Consigue y provee a la organización de los recursos para desarrollar las actividades de acuerdo a las necesidades de cada proceso y proyecto.

Sub-Procesos de Recursos

Recursos Humanos y Ambiente de Trabajo:

Provee y administra los recursos humanos y busca mantener un ambiente de trabajo adecuado en la organización.

Bienes, Servicios e Infraestructura:

Provee, administra y mantiene los recursos de la organización para que la misma pueda operar.

Conocimiento de la Organización:

Provee, administra y mantiene las herramientas y repositorios que conforman la base de conocimiento de la organización.

Procesos de Operación

Administración de Proyectos Específicos

Administra los proyectos internos y externos en base a los planes de cada uno, genera acciones correctivas.

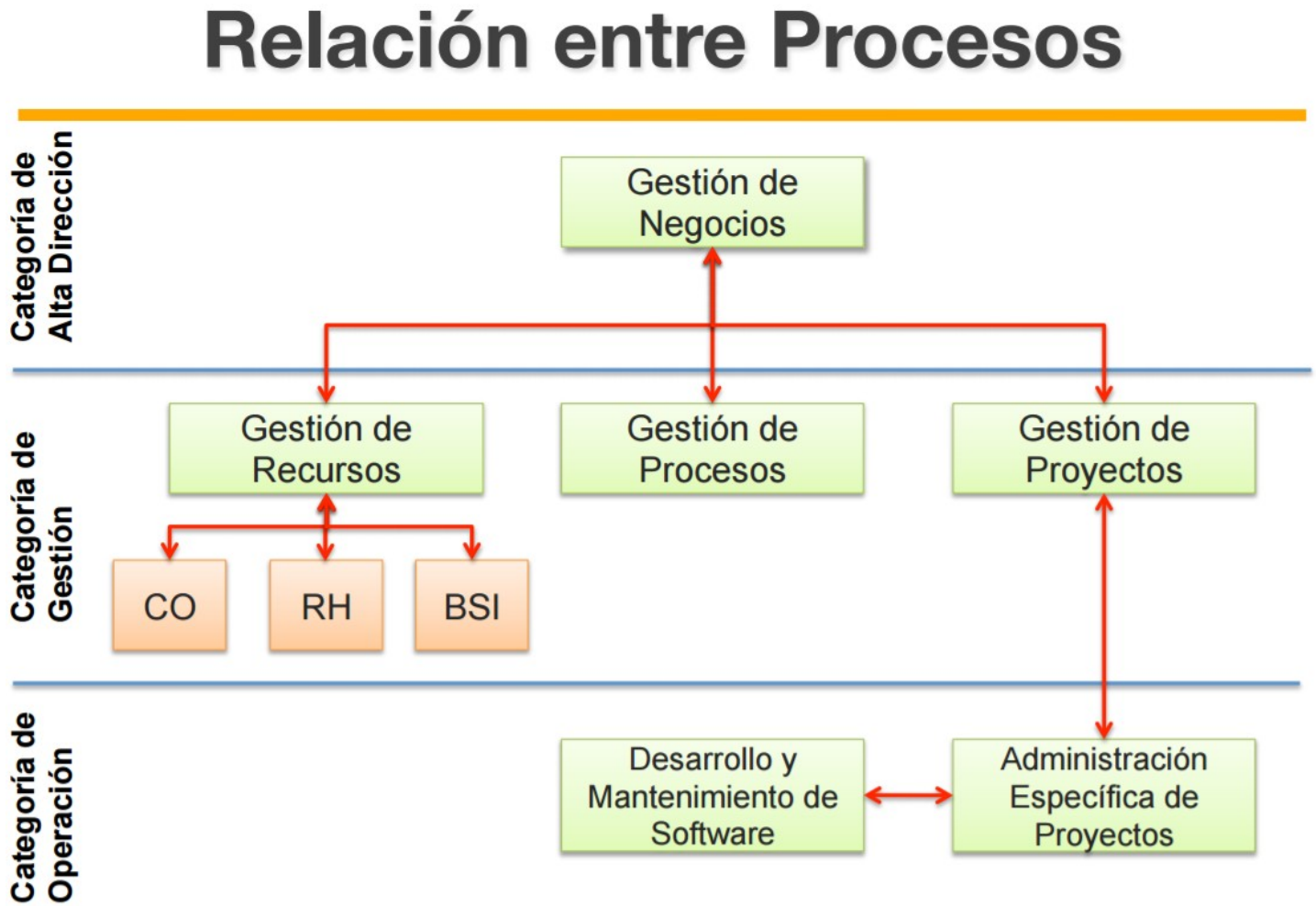
Desarrollo y Mantenimiento de Software:

Genera los productos a través del ciclo de vida de desarrollo del software buscando satisfacer las necesidades del cliente.

Niveles de Madurez

Nivel	Nivel de Capacidad	Descripción	Color
1	Realizado	El proceso se implementa y alcanza su propósito	Amarillo
2	Gestionado	El proceso realizado se administra. Sus productos de trabajo están establecidos, controlados y mantenidos	Azul
3	Establecido	El proceso realizado y gestionado se implementa por medio de un proceso definido	Verde
4	Predecible	El proceso establecido opera bajo límites definidos y conocidos	Rosa
5	Optimizado	El proceso predecible se mejora continuamente	N.A.

2.8. Modelos estandarizados



Roles

Rol	Descripción
Cliente	Es el que solicita un producto de software y financia el proyecto para su desarrollo o mantenimiento.
Usuario	Es el que va a utilizar el producto de software.
Grupo Directivo	Son los que dirigen a una organización y son responsables por su funcionamiento exitoso.
Responsable de Proceso	Es el encargado de la realización de las prácticas de un proceso y del cumplimiento de sus objetivos.
Involucrado	Otros roles con habilidades requeridas para la ejecución de actividades o tareas específicas. Por ejemplo: Analista, Programador, Revisor, entre otros.

2.8. Modelos estandarizados

Actualmente existe una gran variedad de ***modelos para procesos de software***. Podemos entenderlos más fácilmente si los clasificamos en dos tipos: genéricos y específicos.

CMM (*Capability Maturity Model*) - Modelo de Madurez de Capacidades

Marco que describe elementos clave de procesos efectivos de software.

CMM describe un camino evolutivo en 5 niveles de mejora de procesos para lograr su madurez. Cubre prácticas de **planeación**, **ingeniería** y **administración** del desarrollo y mantenimiento de software.

Modelos Genéricos	Modelos Específicos
Abarcan todos los procesos relacionados con el desarrollo de software	Enfocados a la ingeniería de productos de software
CMM – modelo de madurez de capacidades – estándar de facto	UP – proceso de desarrollo
CMMI – modelo integrado	RUP – proceso de desarrollo
ISO 9001-2000 – sistema para administración de la calidad - estándar	PSP – enfocado en individuos
ISO/IEC 15504 – marco para evaluación de procesos de software – en vías de ser estándar, por ahora reporte técnico	TSP – enfocado en equipos (incluye PSP)
MoProSoft – modelo de procesos para la industria de software en México – en vías de ser norma mexicana	
• Nos dicen qué debemos hacer • Se deben usar como referencia para definir procesos en una organización y para autoevaluación. • Medio para evaluar que tan bien o mal esta la organización.	• Nos dicen el cómo debemos hacer las cosas • Se usan como guía para ejecutar proyectos

2.8. Modelos


PRACTICAS DE CMMI
5
Mejora Continua de Procesos
Optimizado
4
Procesos medidos y controlados
Gestionado Cuantitativamente
3
Estandarizados para la Organización
Definido
2
Administrados por proyectos
Gestionado
1
Procesos impredecibles
Desarrollado

2.8. Modelos estandarizados

RUP

El ***Rational Unified Process*** (**RUP**) es un proceso de software desarrollado y comercializado por Rational Software (ahora parte de IBM). **RUP** está diseñado alrededor de seis mejores prácticas para el desarrollo de software:

- Desarrollar de manera iterativa.
- Administrar los requerimientos.
- Utilizar arquitecturas basadas en componentes.
- Modelar el software visualmente.
- Verificar la calidad de manera continua.
- Controlar los cambios.

RUP

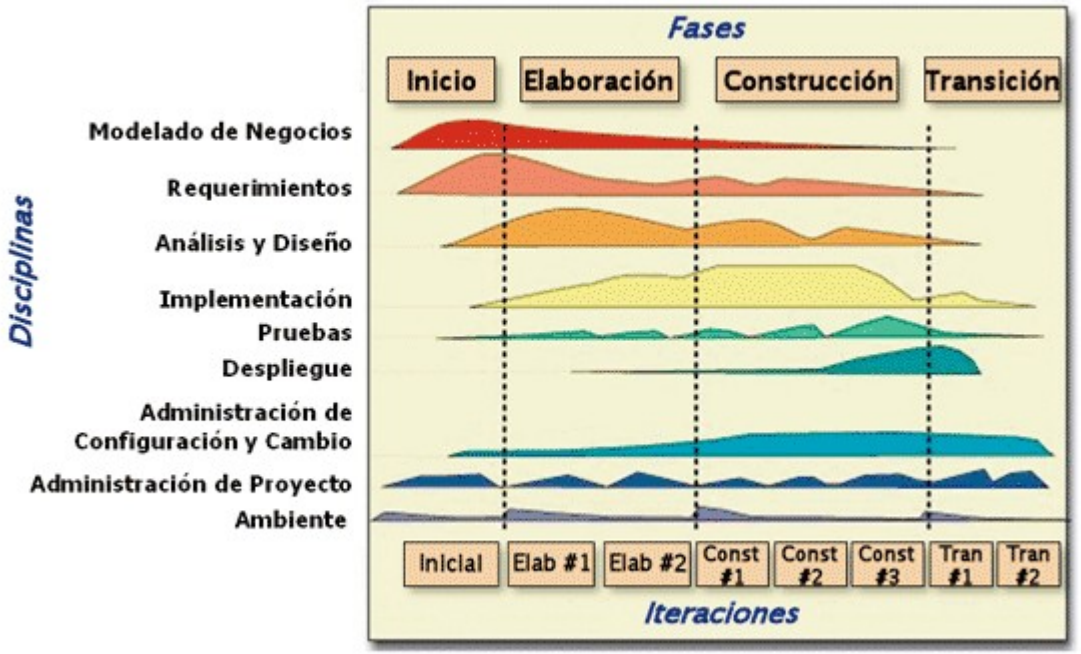
En sí, **RUP** es una guía que define roles, actividades, flujos de trabajo y lineamientos para ejecutar proyectos de software de acuerdo a estas mejores prácticas.

RUP organiza los proyectos de software en dos dimensiones: la del tiempo y la de las actividades. En base al tiempo, los proyectos se dividen en cuatro fases secuenciales:

1. **Concepción** – Definición de alcance, identificación de riesgos.
2. **Elaboración** – Resolución de riesgos, establecimiento de arquitectura.
3. **Construcción** – Generación del producto.
4. **Transición** – Disponibilidad a la comunidad de usuarios finales.

Las actividades se organizan en nueve diferentes disciplinas que son ejecutadas durante las diferentes fases.

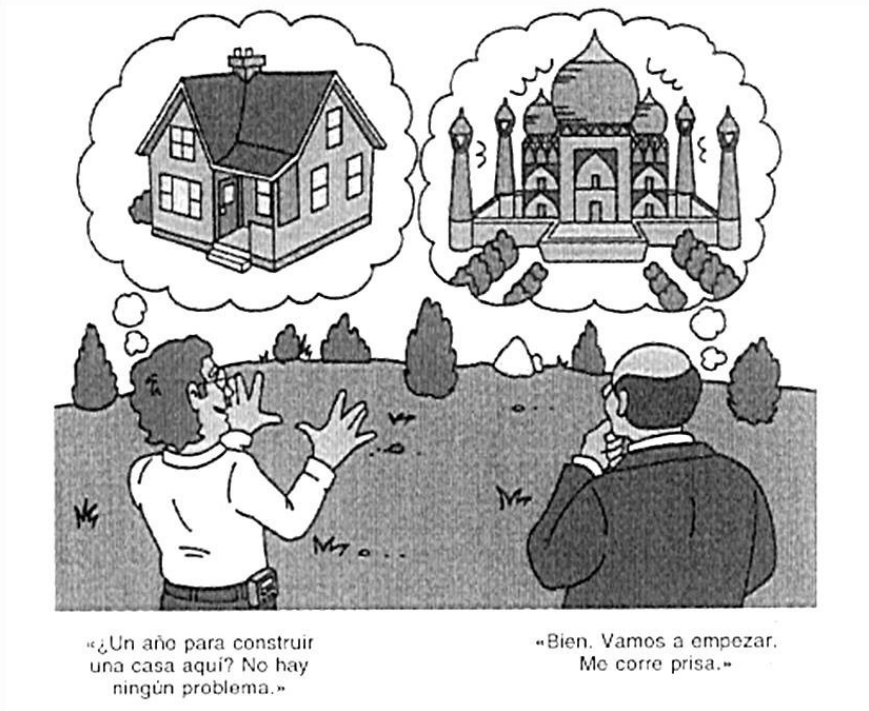
RUP



RUP es un framework (marco de trabajo) que pretende ser personalizado o configurado para organizaciones y proyectos específicos. RUP no se puede aplicar de la misma forma en todos los proyectos de una organización. Es por esto que pretender seguir RUP a través de ir cumpliendo con la lista de artefactos que define, es una estrategia poco efectiva. Lo que las organizaciones deben hacer es entender la razón de ser de RUP – las prácticas citadas anteriormente – y en base a esto aplicar lo que decidan que es conveniente para cada área o proyecto específico.

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Entender los **requerimientos** de un problema es una de las tareas más difíciles que enfrenta el ingeniero de software.



¿Acaso no sabe el **cliente** lo que se necesita?

“Sé que cree que entiende lo que digo, pero lo que usted no entiende es que lo que digo no es lo que quiero decir.”

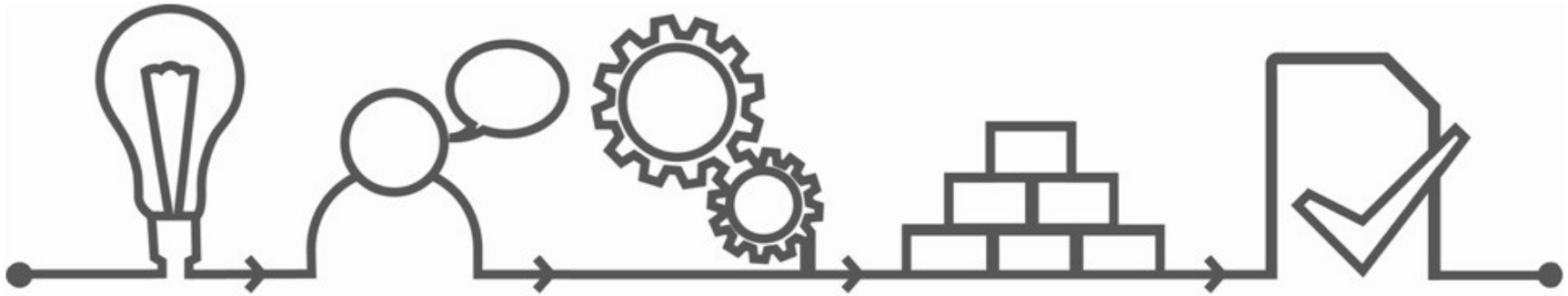
3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

¿Qué es un requerimiento?

Es una característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema.

Es decir, los requerimientos son lo que los clientes/usuarios esperan que haga el sistema.



Especificación de requerimientos
Documentos que reitera la definición e los requerimientos en los términos técnicos apropiados para el desarrollador del sistema.
Es la contrapartida técnica al documento de definición de requerimientos y es escrito por los analistas de requerimientos.

3.1. Proceso de la ingeniería de requerimientos

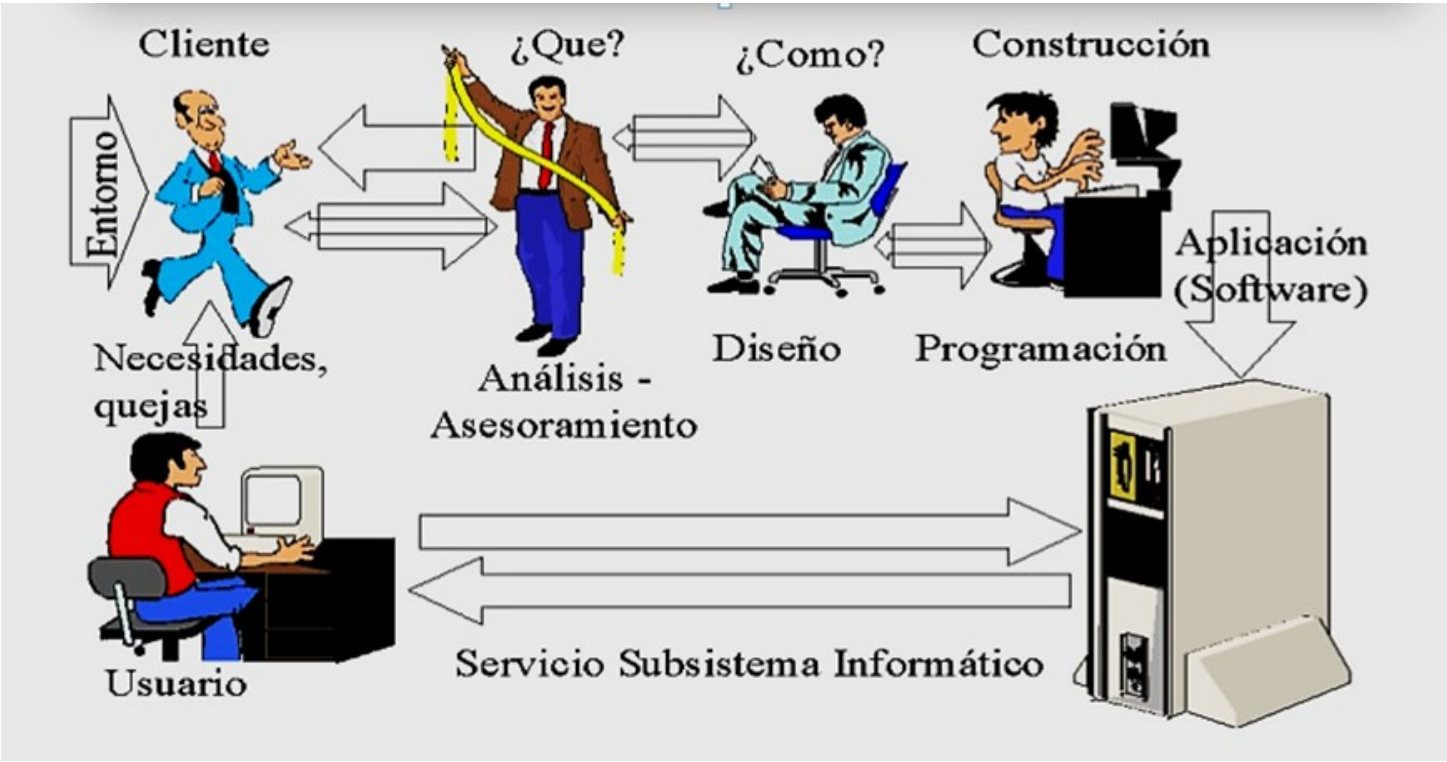
3. ESPECIFICACIÓN DE REQUERIMIENTOS

Según el tipo los requerimientos se clasifican en:

- ✓ ***Requerimientos funcionales***
- ✓ ***Requerimientos no funcionales***
- ✓ ***Requerimientos del dominio***

Según a quien van dirigidos se clasifican en:

- ✓ ***Requerimientos del usuario***
- ✓ ***Requerimientos del sistema***



3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos funcionales

Describen las **funciones** que el sistema va a hacer. Estos requerimientos dependen del tipo de software que se desarrolla, los posibles usuarios del software y del enfoque en la organización al redactar los requerimientos; los **requerimientos funcionales** del sistema describen con detalle la función de este, sus entradas y salidas, excepciones, etc.

Número	Requerimiento	Descripción	Prioridad
RF1	LA ADMINISTRACION LLEVARA UN CONTROL DE NOTAS DE LOS ALUMNOS	El sistema tendrá las notas de los alumnos en cada curso que ha llevado y que esta cursando actualmente,	5
RF2	EL ADMINISTRADOR PODRA MODIFICAR LAS NOTAS	El sistema deberá permitir la modificación de las notas del curso del ciclo vigente	5

R4 - Calcular el saldo total de un cliente del banco
R5 - Avanzar la fecha de la simulación en un mes
R6 - Retirar de la cuenta de ahorros
R7 - Retirar de la cuenta corriente

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos no funcionales

Son aquellos requerimientos que no se refieren directamente las funciones especificas que proporciona el sistema, si no a las propiedades emergentes, como son la fiabilidad, el tiempo de respuesta del sistema y la capacidad de almacenamiento. De forma alternativa define las restricciones del sistema de los dispositivos de entrada y salida

Número	Requerimiento	Descripción	Prioridad
RNF1	USABILIDAD	Debe ser fácil de usar. Con ayudas e interfaces intuitivas.	5
RNF2	SEGURIDAD	El ingreso al sistema estará restringido bajo contraseñas cifradas y usuarios definidos.	5

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Los requerimientos no funcionales se clasifican según su implicancias

Producto
Especifican comportamiento del producto. ej. de desempeño en la rapidez de ejecución del sistema, cuanta memoria se requiere; los de fiabilidad que fijan la tasa de fallas para el sistema sea aceptable, los de portabilidad y de usabilidad.

Organizacionales
se derivan de las políticas y procedimientos existentes en la organización del cliente y del desarrollador. Ej. estándares en los procesos que deben utilizarse, requerimientos de implementación como los lenguajes de programación o el método de diseño a utilizar

Externos
cubre todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo. ej. requerimientos e interoperabilidad, requerimientos legales, requerimientos éticos.

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos del dominio

Son requerimientos que provienen del dominio de aplicación del sistema y que refleja las características y restricciones de ese dominio. Pueden ser funcionales o no funcionales.

Ejemplo

Deberá existir una interfaz de usuario estándar para todas la bases de datos que estará basada en el estándar z39.50.

Debido a las restricciones de derechos de autor, algunos documentos deberán borrarse después de su llegada, se imprimirán de forma local en el servidor y serán distribuidos en forma manual.

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos del usuario

Son declaraciones en lenguaje natural y en diagramas de los servicios que se espera que el sistema provea y de las restricciones bajo las cuales debe operar.

Ejemplo

El cliente necesita en su empresa un sistema para la orden de entrega de los productos que comercializa, que todo lo que se registros hechos por los vendedores llegue de una vez a bodega para agilizar la entrega del mismo pedido, sin tener que acercar se el vendedor a entregar la orden de pedidos a la empresa.

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos del sistema

Establecen con detalle los servicios y restricciones del sistema. El documento de requerimientos del sistema, algunas veces denominado especificación funcional, debe ser preciso. Éste sirve como un contrato entre el comprador del sistema y el desarrollador del software.

Son descripciones más detalladas de los requerimientos del usuario. Sirven como base para definir el contrato de la especificación del sistema y, por lo tanto, debe ser una especificación completa y consistente del sistema. Son utilizados por los ingenieros de software como el punto de partida para el diseño del sistema.

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

*Los **requerimientos** deben ser de alta calidad para la buena comprensión de clientes y desarrolladores.* Con este fin debe comprobarse que los requerimientos posean las características que se desprenden de las siguientes preguntas:

¿Los requerimientos son correctos? El cliente y el desarrollador deben revisarlos para asegurarse que no tienen errores.

¿los requerimientos son consistentes? Es decir, *¿los requerimientos planteados son no conflictivos ni ambiguos?*. Los requerimientos son inconsistentes cuando es imposible satisfacerlos simultáneamente.

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

¿Los requerimientos son completos? El conjunto de requerimientos es completo si todos los estados posibles, cambios de estado, entradas, productos, restricciones están descritos en alguno de los requerimientos.

¿Los requerimientos son realistas? *¿El sistema puede hacer realmente lo que el cliente esta pidiendo que haga?.* Todos los requerimientos deben ser revisados para asegurarse que son posibles.

¿Describe cada requerimiento algo que es necesario para el cliente? Los requerimientos deben ser revisados para conservar solo aquellos que inciden directamente en la resolución del problema del cliente.

¿los requerimientos son verificables? Debemos preparar pruebas que demuestren que se ha cumplido los requerimientos

3.1. Proceso de la ingeniería de requerimientos

3. ESPECIFICACIÓN DE REQUERIMIENTOS

¿Los requerimientos son rastreables? *¿Se puede rastrear cada función del sistema hasta el conjunto de requerimientos que la establece?. ¿Resulta fácil encontrar el conjunto de requerimientos que coinciden a un aspecto especifico del sistema?*

- ✓ ***Necesario***
- ✓ ***No ambiguo***
- ✓ ***Conciso***
- ✓ ***Consistente***
- ✓ ***Completo***
- ✓ ***Alcanzable***
- ✓ ***Verificable***

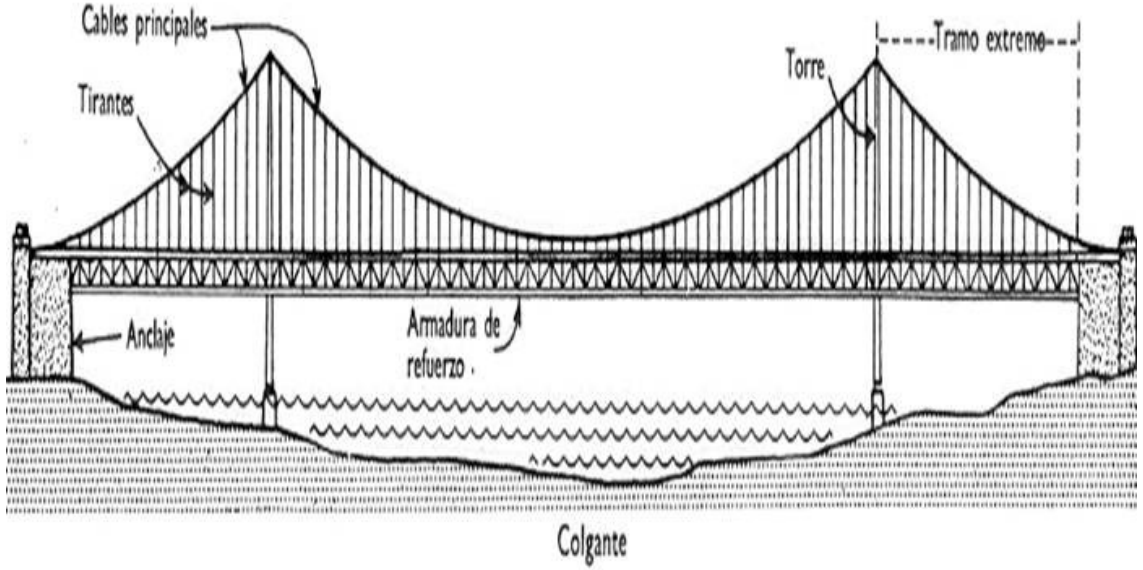
3.1. Proceso de la ingeniería de requerimientos

Introducción a la Ingeniería del Software

Desde la perspectiva del **proceso del software**, la **ingeniería de requerimientos** es una de las acciones importantes de la ingeniería de software que comienza durante la actividad de comunicación y continúa en la de modelado.

Debe adaptarse a las necesidades del proceso, del proyecto, del producto y de las personas que hacen el trabajo.

La **ingeniería de requerimientos** tiende un puente para el diseño y la construcción.



Introducción a la Ingeniería del Software

La **ingeniería de requerimientos** proporciona el mecanismo apropiado para entender lo que desea el **cliente**, analizar las **necesidades**, evaluar la **factibilidad**, negociar una **solución razonable**, **especificar la solución** sin ambigüedades, validar la especificación y administrar los requerimientos a medida de que se transforman en un sistema funcional.

Incluye siete tareas diferentes:

- ✓ Concepción
- ✓ Indagación
- ✓ Elaboración
- ✓ Negociación
- ✓ Especificación
- ✓ Validación
- ✓ Administración

¿Cómo inicia un proyecto de software?

En ciertos casos, una conversación casual es todo lo que se necesita para desencadenar un trabajo grande de **ingeniería de software**.

En la concepción del proyecto, se establece el entendimiento básico del problema, las personas que quieren una solución, la naturaleza de la solución que se desea, así como la eficacia de la comunicación y colaboración preliminares entre los otros participantes y el equipo de software.

Concepción

Preguntar al cliente, a los usuarios y a otras personas

¿Cuáles son los objetivos para el sistema o producto?
¿Qué es lo que va a lograrse?
¿Cómo se ajusta el sistema o producto a las necesidades del negocio?
¿Cómo va a usarse el sistema o producto en las operaciones cotidianas?

Christel y ***Kang*** identificaron cierto número de problemas que se encuentran cuando ocurre la indagación:

Problemas de alcance**.** Los clientes o usuarios finales especifican detalles técnicos innecesarios que confunden, más que clarifican, los objetivos generales del sistema.

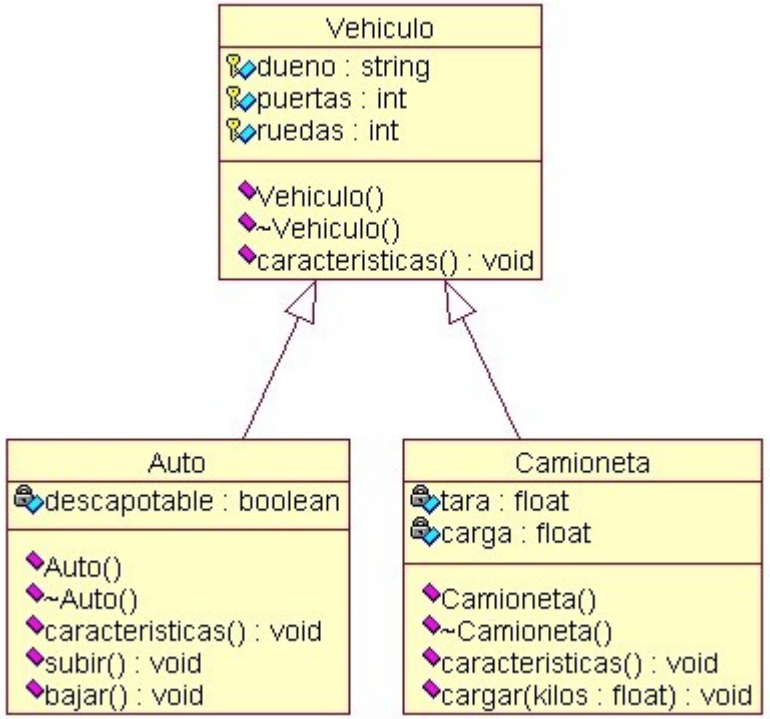
Problemas de entendimiento**.** Los clientes o usuarios no están completamente seguros de lo que se necesita, comprenden mal las capacidades y limitaciones de su ambiente de computación, no entienden todo el dominio del problema, tienen problemas para comunicar las necesidades al ingeniero de sistemas, omiten información que creen que es “obvia”, especifican requerimientos que están en conflicto con las necesidades de otros clientes o usuarios, o solicitan requerimientos ambiguos o que no pueden someterse a prueba.

Problemas de volatilidad**.** Los requerimientos cambian con el tiempo.
Para superar estos problemas, debe enfocarse la obtención de requerimientos en forma organizada.

La información obtenida del cliente durante la concepción e indagación se expande y refina durante la elaboración.

Esta tarea se centra en desarrollar un modelo refinado de los que identifique distintos aspectos de la función del software, su comportamiento e información.

La elaboración está motivada por la creación y mejora de escenarios de usuario que describan cómo interactuará el usuario final (y otros actores) con el sistema.



Introducción a la Ingeniería del Software

No es raro que los clientes y usuarios pidan más de lo que puede lograrse dado lo limitado de los recursos del negocio. También es relativamente común que distintos clientes o usuarios propongan requerimientos conflictivos con el argumento de que su versión es **“esencial para nuestras necesidades especiales”**.

Estos conflictos deben reconciliarse por medio de un proceso de negociación.

Se pide a clientes, usuarios y otros participantes que ordenen sus requerimientos según su prioridad y que después analicen los conflictos.

Con el empleo de un enfoque iterativo que da prioridad a los requerimientos, se evalúa su costo y riesgo, y se enfrentan los conflictos internos; algunos requerimientos se eliminan, se combinan o se modifican de modo que cada parte logre cierto grado de satisfacción.

Introducción a la Ingeniería del Software

En el contexto de los sistemas basados en computadora (y software), el término *especificación* tiene diferentes significados para distintas personas.

Una especificación puede ser un documento escrito, un conjunto de modelos gráficos, un modelo matemático formal, un conjunto de escenarios de uso, un prototipo o cualquier combinación de éstos.

Algunos sugieren que para una especificación debe desarrollarse y utilizarse una “plantilla estándar”, con el argumento de que esto conduce a requerimientos presentados en forma consistente y por ello más comprensible.

Sin embargo, en ocasiones es necesario ser flexible cuando se desarrolla una especificación. Para sistemas grandes, el mejor enfoque puede ser un documento escrito que combine descripciones en un lenguaje natural con modelos gráficos.

No obstante, para productos o sistemas pequeños que residan en ambientes bien entendidos, quizá todo lo que se requiera sea escenarios de uso.

Introducción a la Ingeniería del Software

La calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación.

La validación de los requerimientos analiza la especificación a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto.

El mecanismo principal de validación de los requerimientos es la revisión técnica.

El equipo de revisión que los valida incluye ingenieros de software, clientes, usuarios y otros participantes, que analizan la especificación en busca de errores de contenido o de interpretación, de aspectos en los que tal vez se requiera hacer aclaraciones, falta de información, inconsistencias (problema notable cuando se hace la ingeniería de productos o sistemas grandes) y requerimientos en conflicto o irreales (no asequibles)

Los requerimientos para sistemas basados en computadora cambian, y el deseo de modificarlos persiste durante toda la vida del sistema.

La administración de los requerimientos es el conjunto de actividades que ayudan al equipo del proyecto a identificar, controlar y dar seguimiento a los requerimientos y a sus cambios en cualquier momento del desarrollo del proyecto.