Ingeniería de Software

Software heredado

- Fueron desarrollados hace varias décadas y han sido modifi cados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación.

“muchos sistemas heredados continúan siendo un apoyo para las funciones básicas del negocio y son ‘indispensables’ para éste”

¿Qué hago si encuentro un sistema heredado de mala calidad?

- **Diseños que no son susceptibles de extenderse,**
- **Código confuso, documentación mala o inexistente**
- **Casos y resultados de pruebas que nunca se archivaron**
- **Una historia de los cambios mal administrada…**

Conforme pasa el tiempo... el Software Evoluciona

- El software debe adaptarse para que cumpla las necesidades de los nuevos ambientes del cómputo y de la tecnología.
- El software debe ser mejorado para implementar nuevos requerimientos del negocio.
- El software debe ampliarse para que sea operable con otros sistemas o bases de datos modernos.
- La arquitectura del software debe rediseñarse para hacerla viable dentro de un ambiente de redes

Aplicar reingeniería del sistema heredado para que sea viable en el futuro.

INGENIERÍA DE SOFTWARE

- Cuando se construye una software nuevo, deben escucharse varias opiniones.
  - Cada una de estas puede tener una idea algo distinta de cuales características y funciones debería tener el software.
    - Entender el problema antes de dar la Solución
- Los nuevos Requerimientos son cada vez mas complejos.
- El software actualmente se halla incrustado en ‘Todo’
- La complejidad de los nuevos sistemas demanda una atención cuidadosa a las interacciones de todos los elementos del Sistema

“Se concluye que el diseño se ha vuelto una actividad crucial.”

- Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos.
  - Que pasa en caso de un fallo de software?

Se concluye que el software debe tener alta calidad

- A medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan.
  - Que pasa mientras mas tiempo pase y mientras mas usuarios se sumen?

Se concluye que el software debe tener facilidad para recibir mantenimiento

- La ingeniería de software es una tecnología con varias capas, cualquier enfoque de ingeniería debe basarse en un compromiso con la calidad
  - Six Sigma.
  - Otras filosofías de mejora continua.

- El fundamento para la ingeniería de software es la capa **proceso**.
  - Es el aglutinante que une las capas de la tecnología y permite el desarrollo racional y oportuno.
  - Forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo.
    - Modelos.
    - Documentos.
    - Datos.
    - Reportes.
    - Formatos.
    - etc.
  - Se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada.

- Los **Métodos** proporcionan la experiencia técnica
  - Incluyen un conjunto amplio de tareas, tales como:
    - Comunicación
    - Análisis de Requerimientos
    - Modelación del diseño
    - Construcción del software
    - Apoyo

- Las **herramientas** de la ingeniería de software proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos.

Incrementos

- Para muchos proyectos de software, las actividades estructurales se aplican en forma iterativa a medida que avanza el proyecto.

- Cada iteración produce un incremento del software que da a los participantes un subconjunto de **características** y **funcionalidad** generales del software.

- Conforme se produce cada incremento, el software se hace más y más completo.

Actividades sombrilla.

- Se aplican a lo largo de un proyecto de software y ayudan al equipo que lo lleva a cabo a administrar y controlar el avance, la calidad, el cambio y el riesgo. Normalmente son las siguientes:
  - Seguimiento y control del proyecto de software.
  - Administración del riesgo.
  - Aseguramiento de la calidad del software.
  - Revisiones técnicas.
  - Administración de la configuración del software.
  - Administración de la reutilización.
  - Preparación y producción del producto del trabajo.

Seguimiento y control del proyecto de software.

Permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto y tome cualquier acción necesa ria para apegarse a la programación de actividades.

Administración del riesgo.

Evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto

Aseguramiento de la calidad del software.

Define y ejecuta las actividades requeridas para garantizar la calidad del software.

Revisiones técnicas.

Evalúa los productos del trabajo de la ingeniería de software a fin de descubrir y eliminar errores antes de que se propaguen a la siguiente actividad

Administración de la configuración del software.

Administra los efectos del cambio a lo largo del proceso del software

Administración de la reutilización.

Define criterios para volver a usar el producto del trabajo (incluso los componentes del software) y establece mecanismos para obtener com ponentes reutilizables.

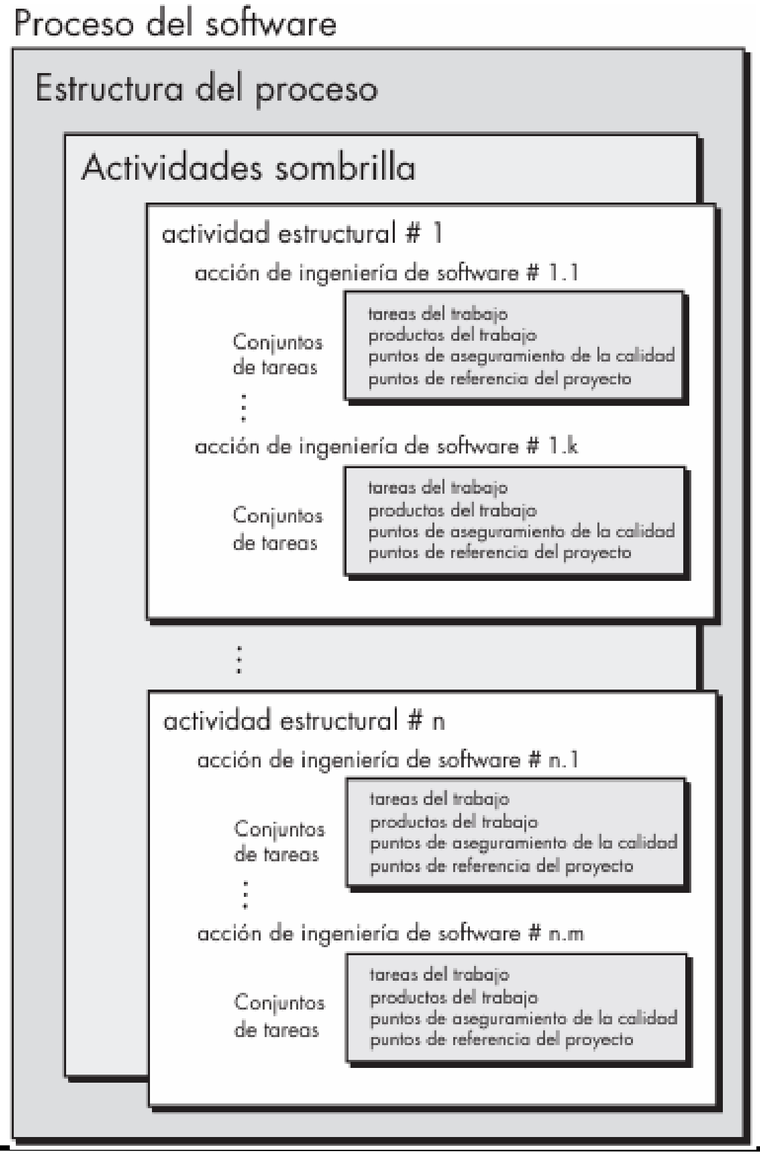
Preparación y producción del producto del trabajo.

Agrupa las actividades requeridas para crear productos del trabajo, tales como modelos, documentos, registros, formatos y listas

Principios Generales del Modelado

1. La razón de que exista
2. MSE
3. Mantener la visión
4. Otros consumirán lo que usted produce
5. Ábrase al futuro
6. Planee por anticipado la reutilización
7. ¡Piense!

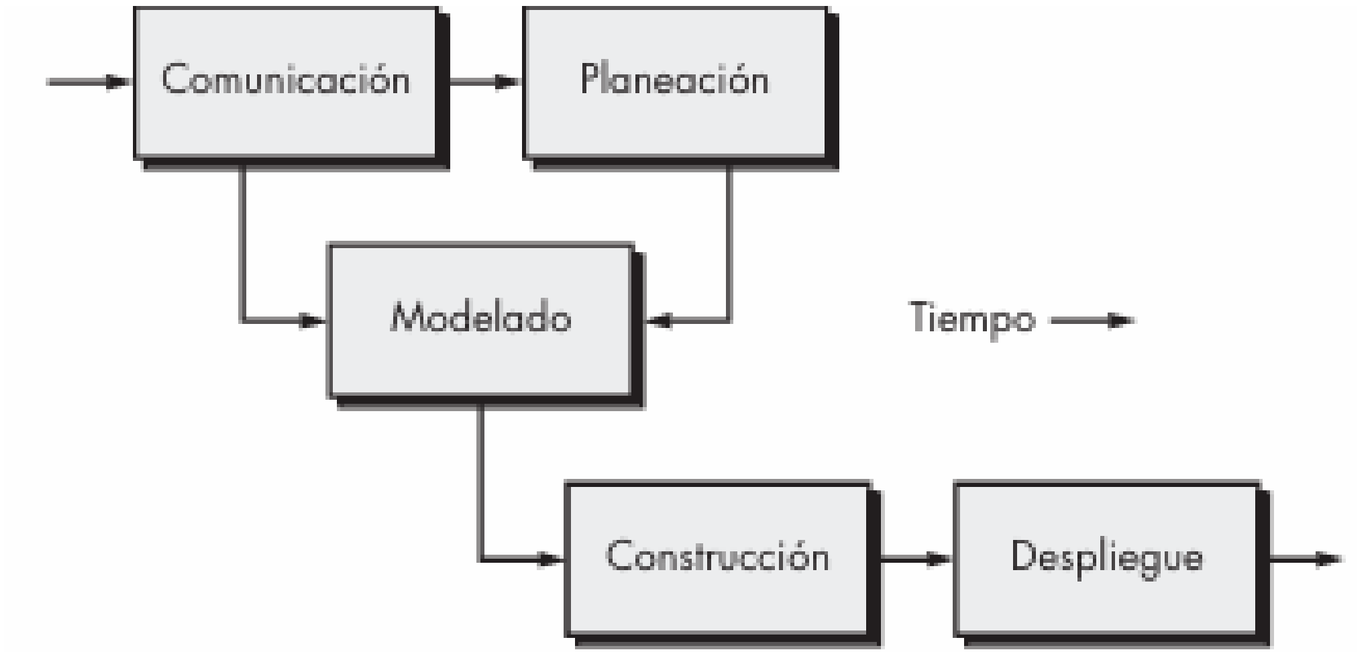
Estructura de un proceso del software



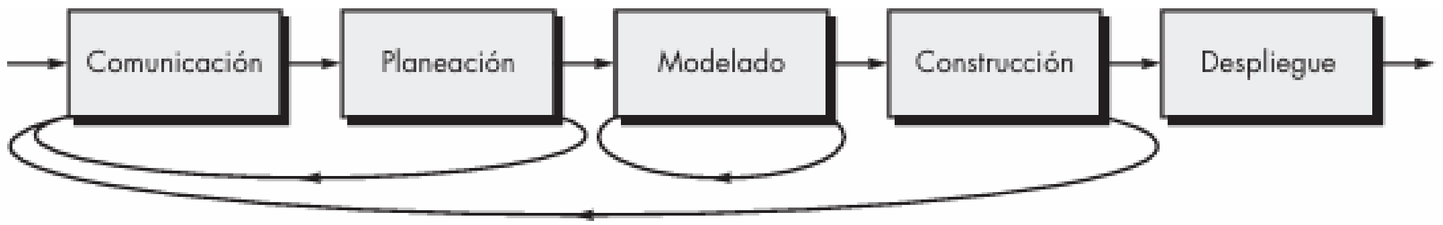
Flujo del proceso

Lineal

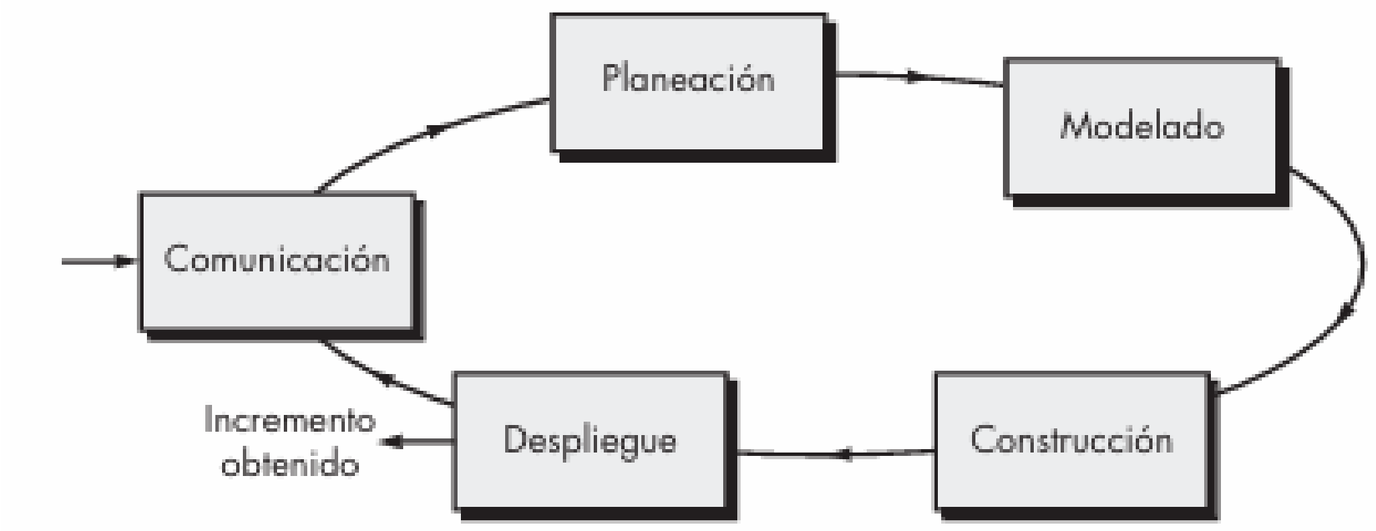
Paralelo



Iterativo

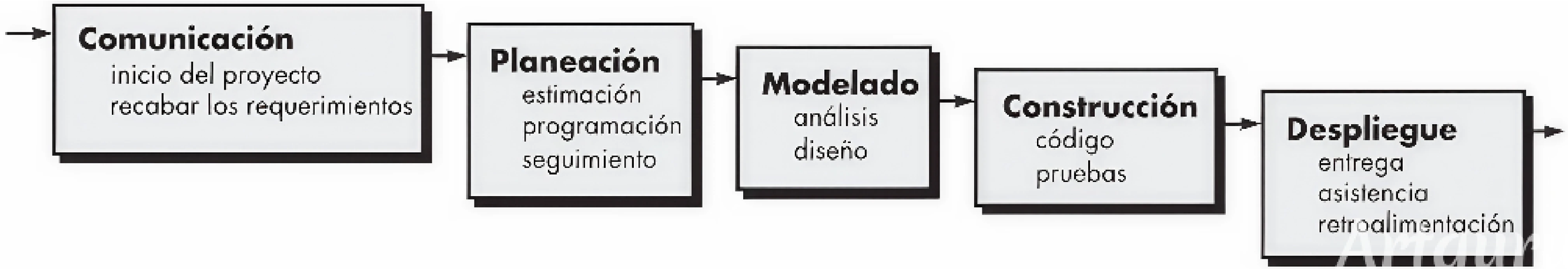


Evolutivo

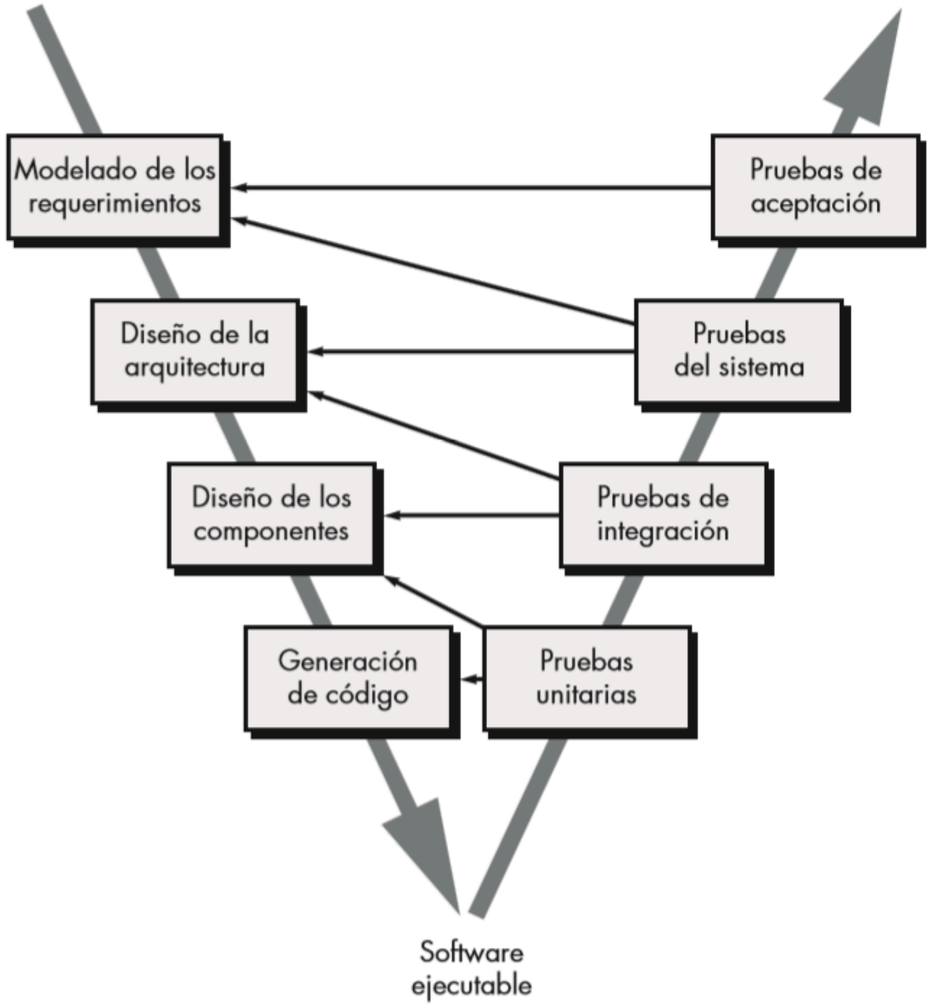


MODELOS DE PROCESO PRESCRIPTIVO

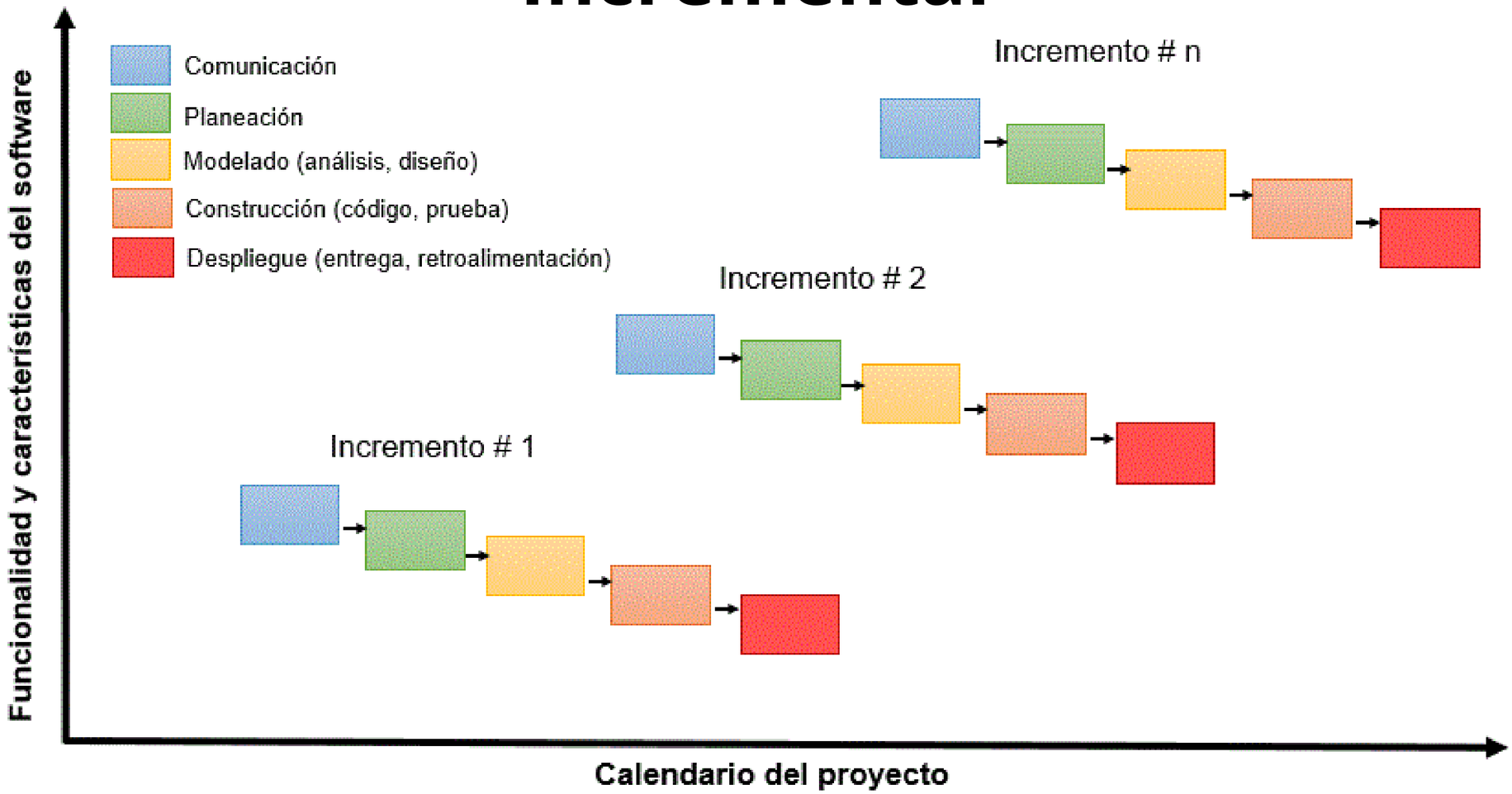
Cascada


Comunicación
inicio del proyecto
recabar los requerimientos
Planeación
estimación
programación
seguimiento
Modelado
análisis
diseño
Construcción
código
pruebas
Despliegue
entrega
asistencia
retroalimentación

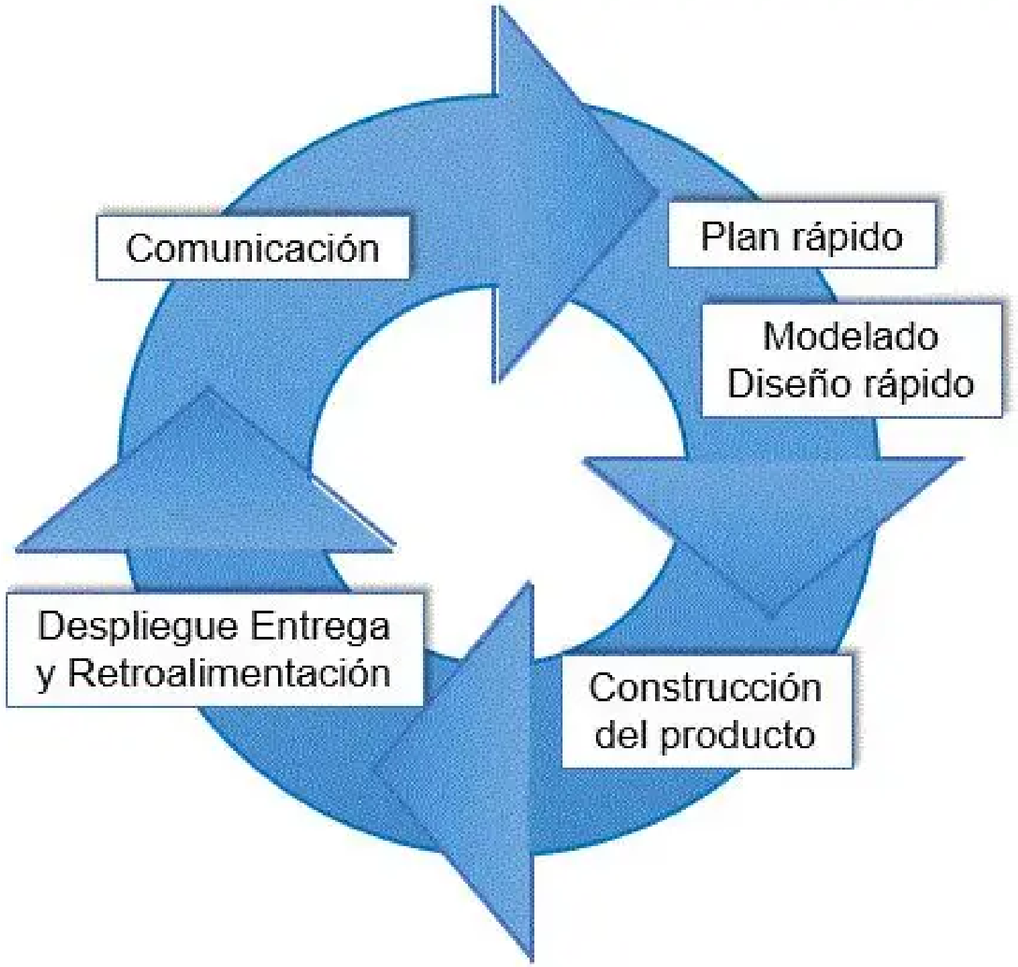
Cascada en V


Modelado de los requerimientos
Pruebas de aceptación
Diseño de la arquitectura
Pruebas del sistema
Diseño de los componentes
Pruebas de integración
Generación de código
Pruebas unitarias
Software ejecutable

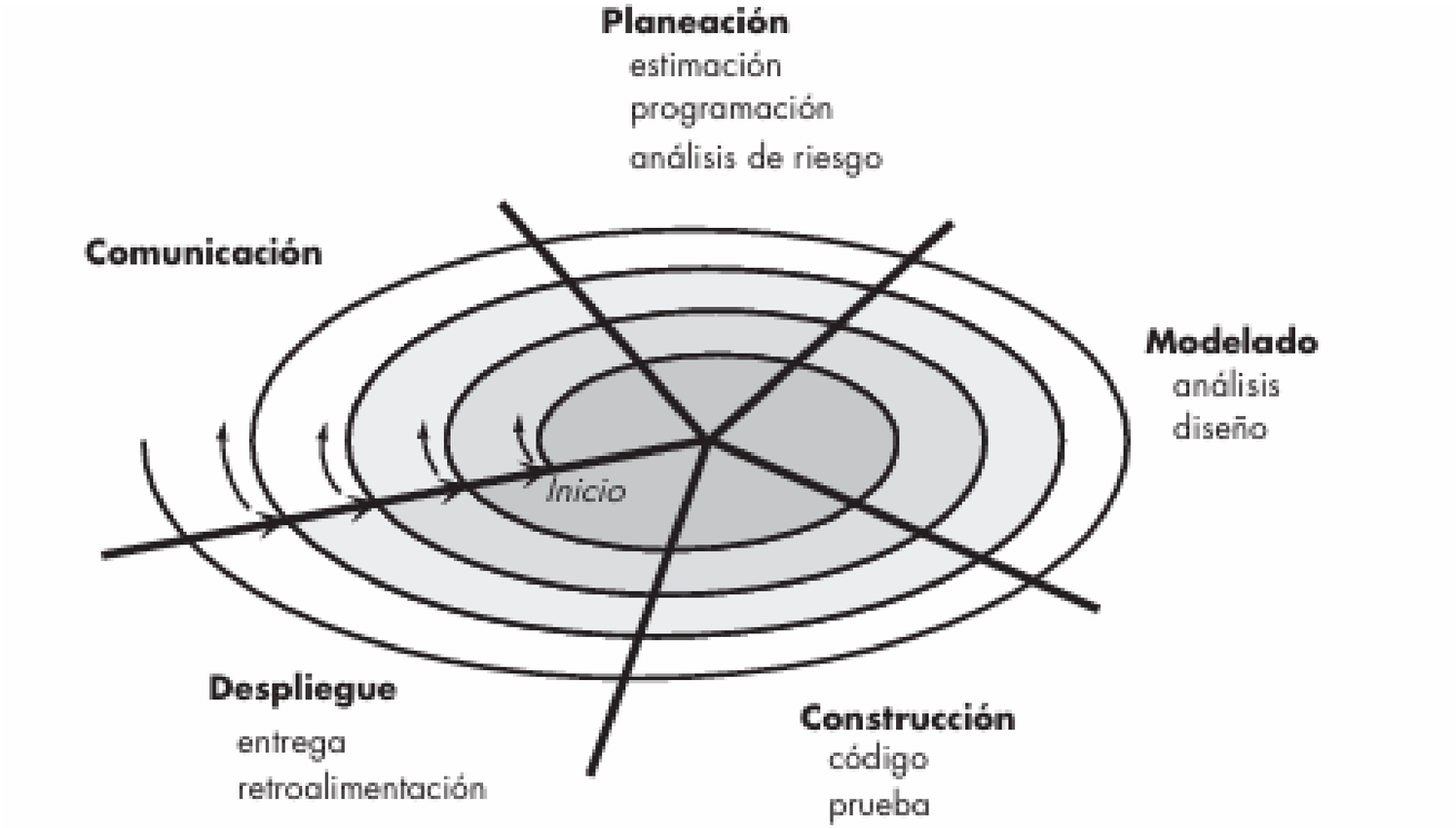
Incremental



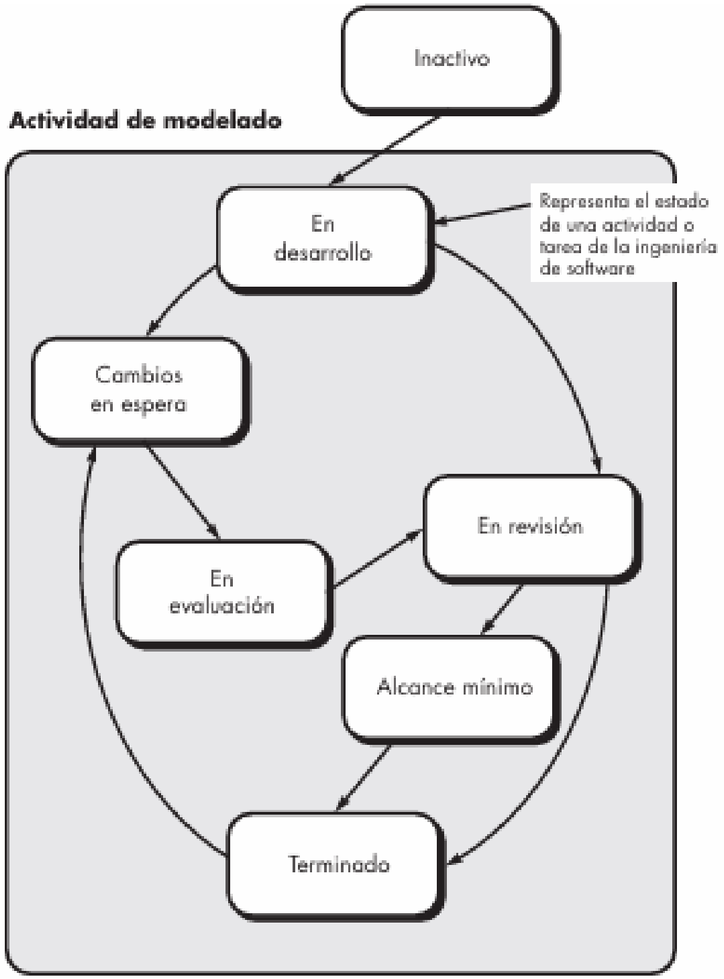
Evolutivo


Comunicación
Plan rápido
Modelado
Diseño rápido
Construcción
del producto
Despliegue Entrega
y Retroalimentación

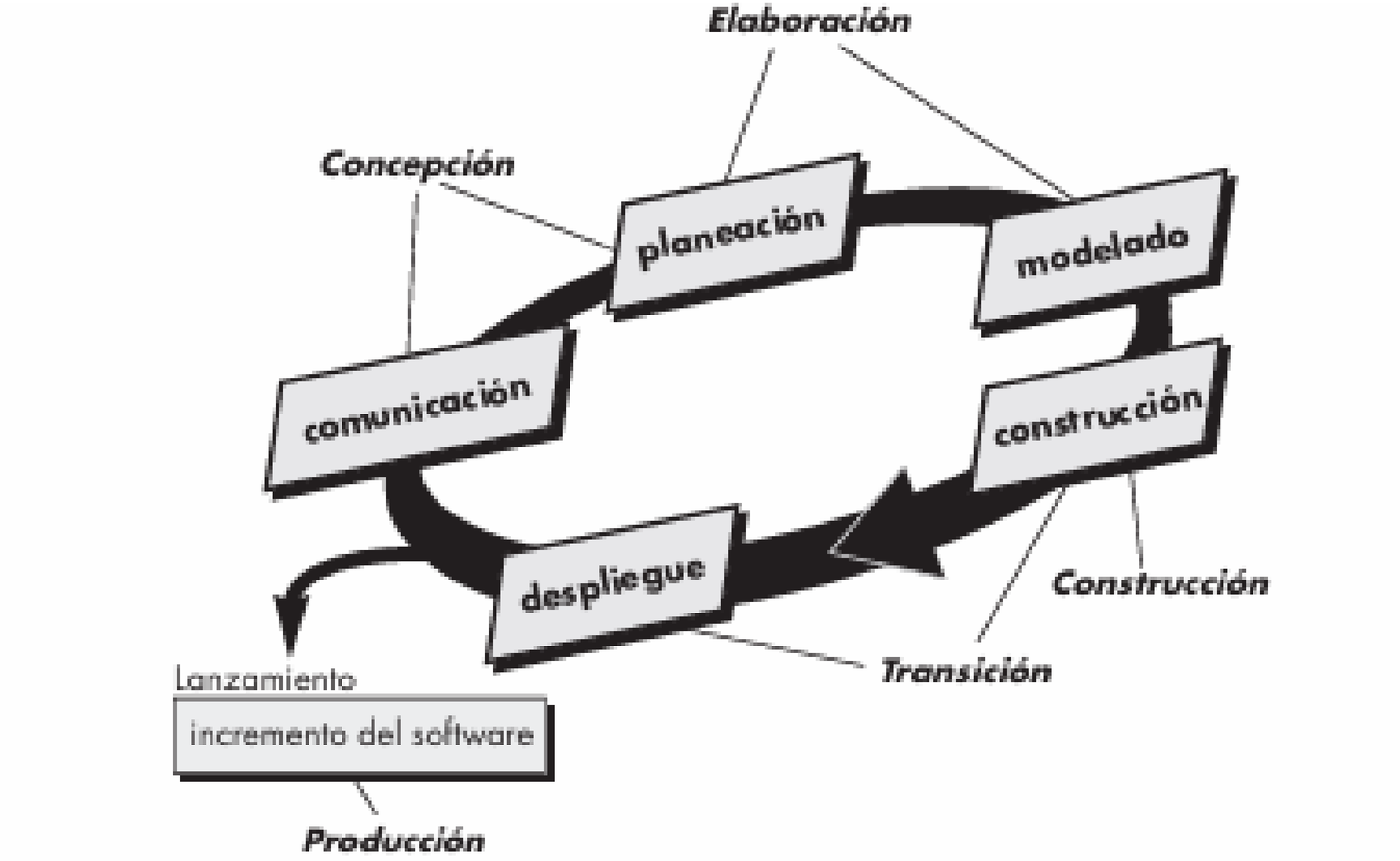
Espiral Comun



Concurrentes



El proceso unificado



Agilidad

¿QUÉ ES UN PROCESO ÁGIL?

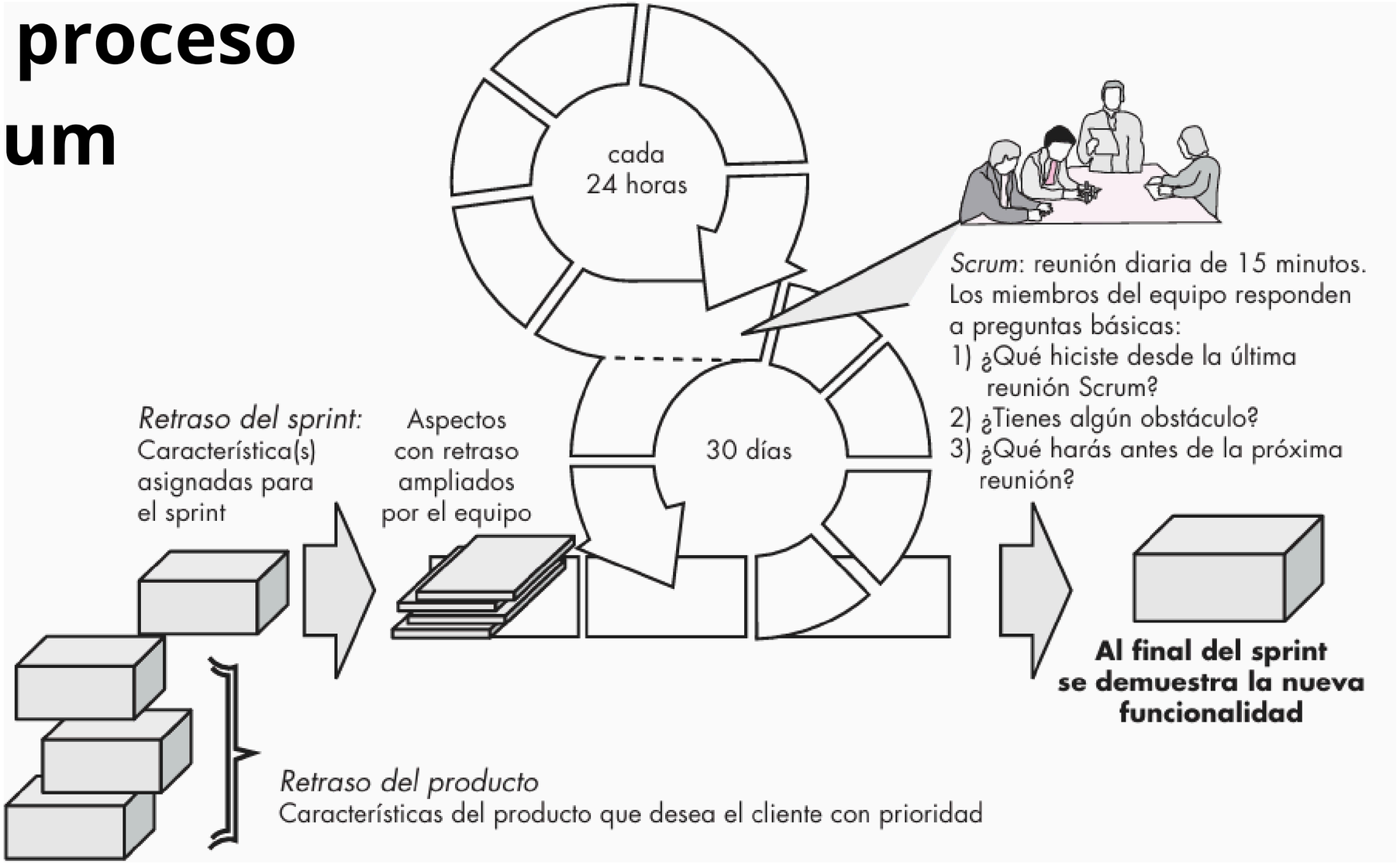
1. Es difícil predecir qué requerimientos de software persistirán y cuáles cambiarán. Tam bién es difícil pronosticar cómo cambiarán las prioridades del cliente a medida que avanza el proyecto.
2. Para muchos tipos de software, el diseño y la construcción están imbricados. Es decir, ambas actividades deben ejecutarse en forma simultánea, de modo que los modelos de diseño se prueben a medida que se crean. Es difícil predecir cuánto diseño se necesita antes de que se use la construcción para probar el diseño.
3. El análisis, el diseño, la construcción y las pruebas no son tan predecibles como nos gustaría (desde un punto de vista de planeación).

Cita:

"Los métodos ágiles obtienen gran parte de su agilidad por basarse en el conocimiento tácito incorporado en el equipo, más que en escribir el conocimiento en planes."

Barry Boehm

Flujo de proceso Scrum



Principios que guían Actividad Estructural

- Que es un **proceso**?
  - Es un conjunto de actividades, acciones y tareas que se ejecuta cuando va a crearse algún producto.

- Que es una **actividad**?
  - Busca lograr un objetivo y se desarrolla sin importar:
    - Dominio de la aplicación.
    - Tamaño del proyecto.
    - Complejidad del esfuerzo.
    - Grado de rigor con el que se usará la ingeniería de software.

- Que es una **acción** ?
  - Es un conjunto de tareas que producen un producto impórtate.
    - Ej. modelo del diseño de la arquitectura

- Que es una **tarea** ?
  - Se centra en un objetivo pequeño pero bien definido que produce un resultado tangible.
    - Ej. Realizar prueba unitaria

- En el contexto de la ingeniería de software, un proceso es un enfoque adaptable que permite que las personas que hacen el trabajo (equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo

- Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

- La estructura del proceso incluye un conjunto de actividades sombrilla que son aplicables a través de todo el proceso del software. Consta de 5 actividades:
  - Comunicación.
  - Planeación.
  - Modelado.
  - Construcción.
  - Despliegue.

Comunicación.

- Muy importante antes de realizar cualquier trabajo técnico.
- Importancia crítica comunicarse y colaborar con el cliente y demás participantes.
- Se busca entender los objetivos de los participantes respecto del proyecto.
- Ayuda a reunir los requerimientos que ayuden a definir las características y funciones del software.

Cliente	Usuario Final
Solicito construcción del Software.	Usará el software construido para algún proposito
Define Objetivos Generales.	Definirá los detalles de operación de modo que se alcance el propósito (Ej: de negocio).
Proporciona requerimientos básicos.	
Coordina obtención de fondos.	

Principios de comunicación

1. **Escuchar**
    a. Concentrarse en las palabras del orador, no pensar en la respuesta.
2. **Prepárate antes de comunicar**
    a. Dedicar tiempo a comprender el problema antes de reunirse con otros.
3. **Alguien debe facilitar actividad**
    a. Cada reunión debe tener un Líder para mantener conversación en una dirección productiva. Mediar conflictos y garantizar que se sigan los principios
4. **Cara a cara es lo mejor**
5. **Tomar Notas y Documentar Decisiones**
6. **Esforzarse por la colaboración**
7. **Mantenerse enfocado, modularize su discusión**
8. **Si algo no está claro, haz un dibujo**
9. c. Una vez que este de acuerdo en algo, seguir adelante.
    a. Si no puede ponerse de acuerdo, seguir adelante.
    b. Sí característica o función no está clara y no se puede aclarar, continúe**.**
10. **La negociación no es un concurso o un juego. Funciona mejor cuando las dos partes ganan.**

Planeación

- Un proyecto de software es un viaje difícil, y la actividad de planeación crea un **“mapa”** que guía al equipo mientras viaja.
- Este mapa se llama **“plan del proyecto de software”**
  - Define el trabajo de ingeniería de software al describir:
    - Las tareas técnicas por realizar
    - Los riesgos probables
    - Los recursos que se requieren
    - Los productos del trabajo que se obtendrán
    - Una programación de las actividades

"Los planes son inútiles, pero la planeación es indispensable"

La planeación sigue los siguientes principios:

1. Comprender alcance del proyecto.
2. Involucrar al cliente en actividad de **PLANIFICACIÓN.**
3. Planificación es **Iterativa.**
4. Estima en base a **lo que sabes.**
5. Considerar **riesgo** al definir el plan.
6. Ser **realista**.
7. Ajuste la **Granularidad** a medida que define el plan.
8. Definir cómo se trata de asegurar **calidad**.
9. Describir cómo acomodar el cambio.
10. Seguimiento del plan con frecuencia y ajustes necesarios. **Los proyectos de software se retrasan un día a la vez.**

Modelado.

- El modelado depende de la envergadura y/o complejidad del producto…. y debe además estar acordé con la explotación que se haga de los modelos.

Principios de modelado

- Se crean para entender mejor la realidad.
- Si el modelo a construir es físico
  - Se construye modelo idéntico a escala menor.
- Si es software, se debe adoptar una forma distinta.
- Modelados deben cumplir objetivos en diferentes niveles de abstracción

Principios del Modelado Ágil

1. Equipo de software tiene por objetivo crear el software, no modelos.
2. Viajar ligero, no crear más modelos de lo necesario.
3. Producir modelo más sencillo que describa al problema o software.
4. Construir modelos susceptibles al cambio.
5. Enunciar propósito explícito para cada modelo que se cree.
6. Adaptar modelos al sistema en cuestión.
7. Construir modelos útiles, pero olvidarse de los "perfectos".
8. No ser Dogmático en cuanto a la sintaxis del modelo.
9. Si algo te dice que el modelo no es correcto (aunque se vea bien), hay que preocuparse
10. Obtener retroalimentación tan pronto como sea posible.

Tipos de Modelado

Modelos de requisitos (análisis)

Requisitos del cliente, representando al software en 3 dominios:

1. Información.
2. Funcional.
3. Comportamiento.

Modelos de Diseño

Características del software que ayudan a los profesionales a construirlos de manera efectiva:

1. Arquitectura
2. Interfaz de usuario
3. Detalles a nivel de Componente.


PUNTO
CLAVE
Los modelos de requerimientos representan los requerimientos del cliente. Los modelos del diseño dan una especificación concreta para la construcción del software.

Principios de modelado: Requisitos

1. Dominio de información del problema debe ser representado y comprendido.
2. Definir funciones que realiza el software.
3. Representar comportamiento del software.
4. Los modelos que representen información, función y comportamiento deben dividirse de manera que revelen los detalles en forma estratificada.
5. Análisis debe pasar de la información esencial al detalle de la implementación.

Principios de modelado: Diseño

1. Debe ser trazable al modelo de requisitos.
2. Considerar arquitectura del sistema a construir.
3. Diseño de datos es tan importante como el de las funciones de procesamiento.
4. Las interfaces deben diseñarse con cuidado.
5. Interfaz de usuario debe ajustarse a las necesidades del usuario final.
6. Diseño a Nivel de componente debe ser funcionalmente Independiente.
7. Componentes => Acoplarse (libremente) entre si y con el entorno externo.
8. Modelos deben ser fácilmente comprensibles.
9. Debe desarrollarse iterativamente. En cada iteración se debe lograr mayor simplicidad.

Construcción

Esta actividad combina la generación de código (ya sea manual o auto matizada) y las pruebas que se requieren para descubrir errores en éste

Principios de Construcción

La actividad de construcción incluye un conjunto de tareas de codificación y pruebas que lleva a un software operativo listo para entregarse al cliente o usuario final.

La codificación puede ser:

1. Creación directa en código fuente.
2. Generación automática de código fuente parecida al diseño del componente que se construye.
3. Generación automática de código ejecutable que utiliza un “lenguaje de programación de cuarta generación” (C++)

Las Pruebas Unitarias:

1. De integración
2. De validación
3. De aceptación

Principios de Codificación

1.- Principios de preparación

1. Entender el problema que se trata de resolver.
2. Comprender los principios y conceptos básicos del diseño.
3. Elegir un lenguaje de programación que satisfaga las necesidades del software.
4. Seleccionar un ambiente de programación que disponga de herramientas que hagan más fácil su trabajo.
5. Crear un conjunto de pruebas unitarias que se aplicarán una vez que se haya terminado el componente a codificar

Principios de Codificación

2.- Principios de programación

1. Restringir sus algoritmos por medio del uso de programación estructurada.
2. Seleccionar estructuras de datos que satisfagan las necesidades del diseño.
3. Entender la arquitectura del software y crear interfaces que son congruentes con ella.
4. Mantener la lógica condicional tan sencilla como sea posible.
5. Seleccionar nombres significativos para las variables y seguir otros estándares locales de codificación.
6. Escribir código que se documente a sí mismo.

Principios de Codificación

3.- Principios de validación

1. Realizar el recorrido del código cuando sea apropiado.
2. Llevar a cabo pruebas unitarias y corregir los errores que se detecten.
3. Rediseñar el código.

Principios de la Prueba

1. **Todas las pruebas deben poder rastrearse hasta los requerimientos del cliente.**
    a. El objetivo de las pruebas de software es descubrir errores.
2. **Las pruebas deben planearse mucho antes de que den comienzo.**
    a. Tan pronto como se termina el modelo de requerimientos.
3. **El principio de Pareto se aplica a las pruebas de software.**
    a. Aislar los componentes sospechosos y probarlos a fondo.
4. **Las pruebas deben comenzar “en lo pequeño” y avanzar hacia “lo grande”.**
    a. Centrarse en componentes individuales y luego en su conjunto.
5. **No son posibles las pruebas exhaustivas.**
    a. Para un programa de tamaño moderado, es imposible ejecutar todas las combinaciones de rutas.

Despliegue

El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación.

Principios de despliegue

1. **Deben manejarse las expectativas de los clientes.**
    a. Asegurarse de que el cliente sepa lo que se va a entregar.
2. **Debe ensamblarse y probarse el paquete completo que se entregará.**
    a. Ensamblar todos los archivos del software y realizar prueba con usuarios reales.
3. **Antes de entregar el software, debe establecerse un régimen de apoyo.**
4. **Se deben proporcionar a los usuarios finales materiales de aprendizaje apropiados.**
5. **El software defectuoso debe corregirse primero y después entregarse.**