INGENIERÍA DEL SOFTWARE

UNIDAD 1: INGENIERÍA DEL SOFTWARE
CICLO LECTIVO 2013

OBJETIVOS DE LA CLASE

- **Modelos prescriptivos de procesos.**
  - **Code-and-fix**
  - **Cascada.**
  - **Modelo en V.**
  - **Modelos incrementales.**
  - **Modelos evolutivos.**
  - **Modelos especializados.**
  - **Proceso Unificado (RUP)**
  - **Métodos Ágiles.**

MODELOS PRESCRIPTIVOS Y CICLO DE VIDA

- La preocupación por el "Proceso" (fin de los '80) es más reciente que la definición del "Ciclo de Vida" (fin de los '60).
- En general se asocia a la noción de modelo de proceso o ciclo de vida a un mayor detalle y precisión.
- '*Ordenar el caos en el desarrollo de Software*' (Pressman).

MODELOS PRESCRIPTIVOS DE PROCESOS

- Objetivos
  - Determinar el orden de las etapas involucradas en el desarrollo del software,
  - establecer el criterio de transición para progresar de una etapa a la siguiente (Bohem):
    - criterio para determinar la finalización.
    - criterio para comenzar y elegir la siguiente.
- Un modelo prescriptivo o ciclo de vida apunta a:
  - ¿Qué debemos hacer a continuación?
  - ¿Por cuánto tiempo debemos hacerlo?

MODELOS PRESCRIPTIVOS DE PROCESOS

- Las principales diferencias entre distintos modelos de ciclo de vida están divididas en tres grandes visiones:
  - **El alcance del ciclo de vida** (hasta dónde se desea llegar con el proyecto): sólo saber si es viable el desarrollo de un producto, el desarrollo completo o el desarrollo completo más actualizaciones y mantenimiento.
  - **La cualidad y cantidad** de las etapas en que será dividido el ciclo de vida: según el que sea adoptado y el proyecto para el cual sea adoptado.
  - **La estructura y la sucesión de las etapas**, si hay realimentación entre ellas y si hay libertad de repetirlas (iteración).

MODELOS PRESCRIPTIVOS DE PROCESOS

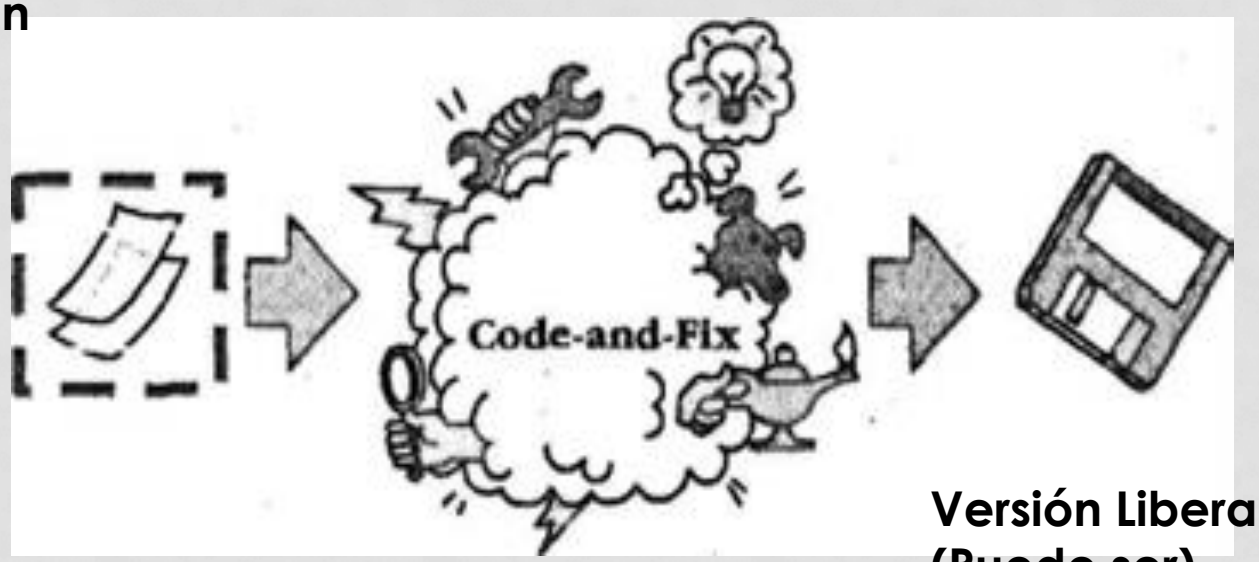
- Prescripciones de la forma en que el desarrollo de software **debería llevarse a cabo**.
- Descripciones de la forma en que el desarrollo **se lleva a cabo realmente**.
- Cada modelo incluye los requerimientos del sistema como **entrada** y el producto liberado al uso como **salida**.

MODELOS PRESCRIPTIVOS DE PROCESOS

1. Modelo en Cascada.
2. Modelo en V.
3. Modelos Incrementales
   a. Modelo Incremental.
   b. Modelo RAD/DRA.
4. Modelos Evolutivos
   a. Construcción de Prototipos.
   b. Modelo en Espiral.
   c. Desarrollo Concurrente
5. Modelos Especializados
   a. Modelo de Desarrollo Basado en Componentes.
   b. Software Orientado a Aspectos
   c. Ciclo de Vida Orientado a Objetos.
6. Rational Unified Process (RUP).
7. Métodos Ágiles.
8. Otros …

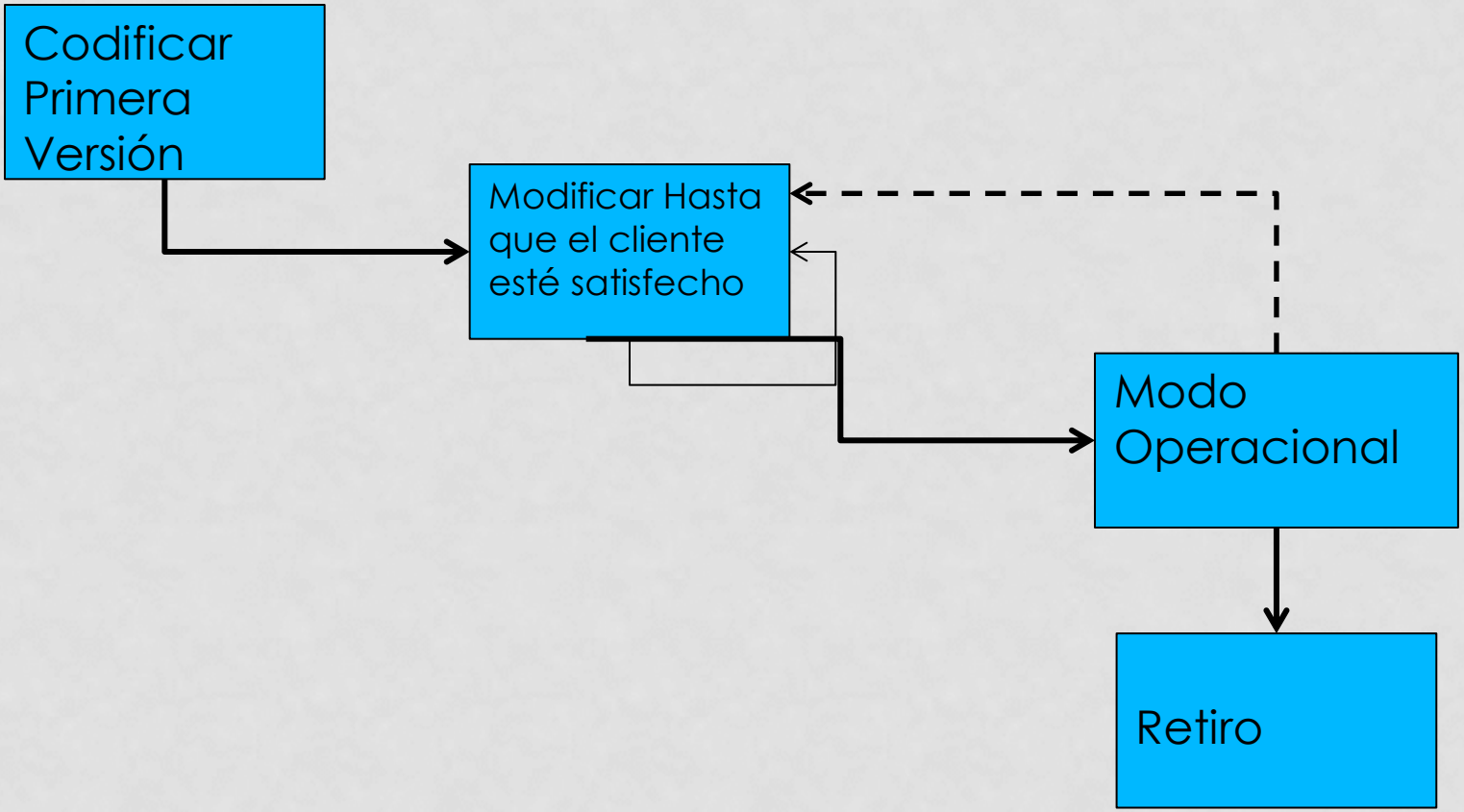
CODE-AND-FIX MODEL

Especificación del sistema (Puede ser)



Versión Liberada (Puede ser)

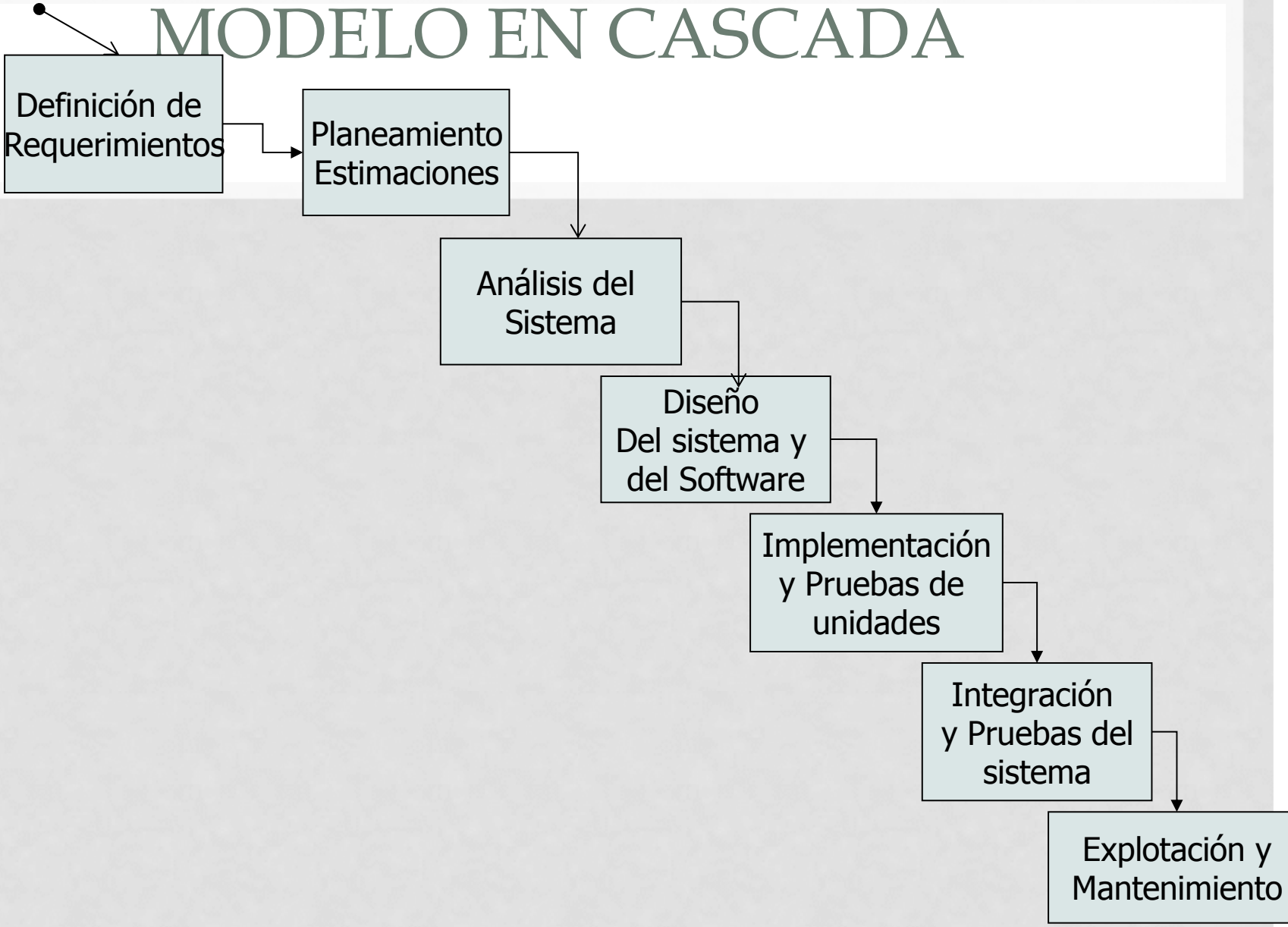
CODE-AND-FIX MODEL



CODE-AND-FIX MODEL

- **Problemas**
  - El código llega a ser muy caro para corregir (los errores no son encontrados hasta muy tarde en el proceso)
  - El código no satisface las necesidades del usuario (no hay fases de requerimientos).
  - El código no fue planeado para modificación, no es flexible.
- ¿Cuándo usar?
  - Existe una programación de desarrollo ajustada → Se desarrolla código rápidamente y se ven 'resultados' inmediatamente.

MODELO EN CASCADA



MODELO EN CASCADA

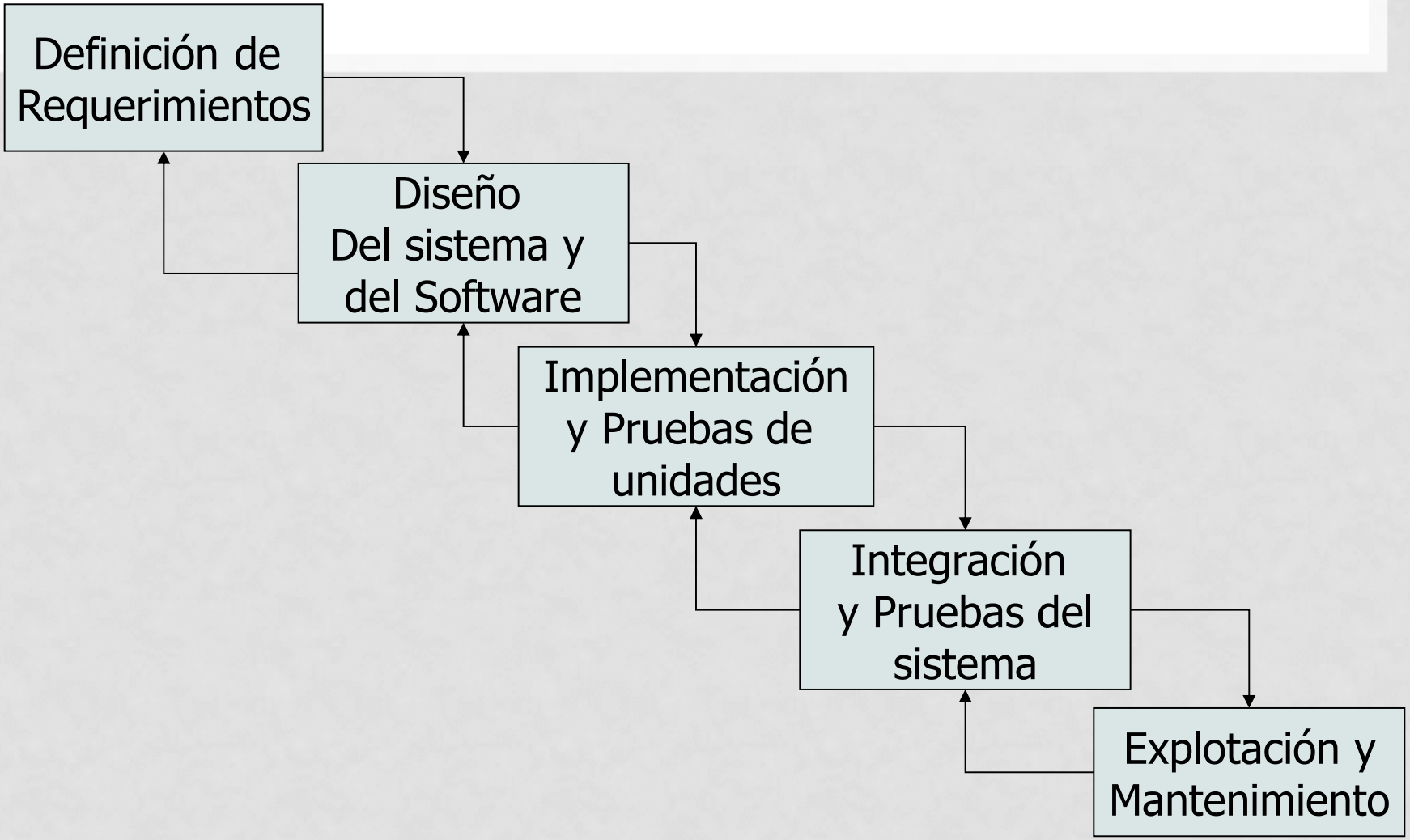
- Ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de *cada etapa debe esperar a la finalización de la inmediatamente anterior.*
- Ventaja:
  - Cada fase genera entradas y documentación a la siguiente.
- Se considera el modelo "clásico" del desarrollo de software
  - + antiguo, + usado
- Enfoque sistemático secuencial

MODELO EN CASCADA

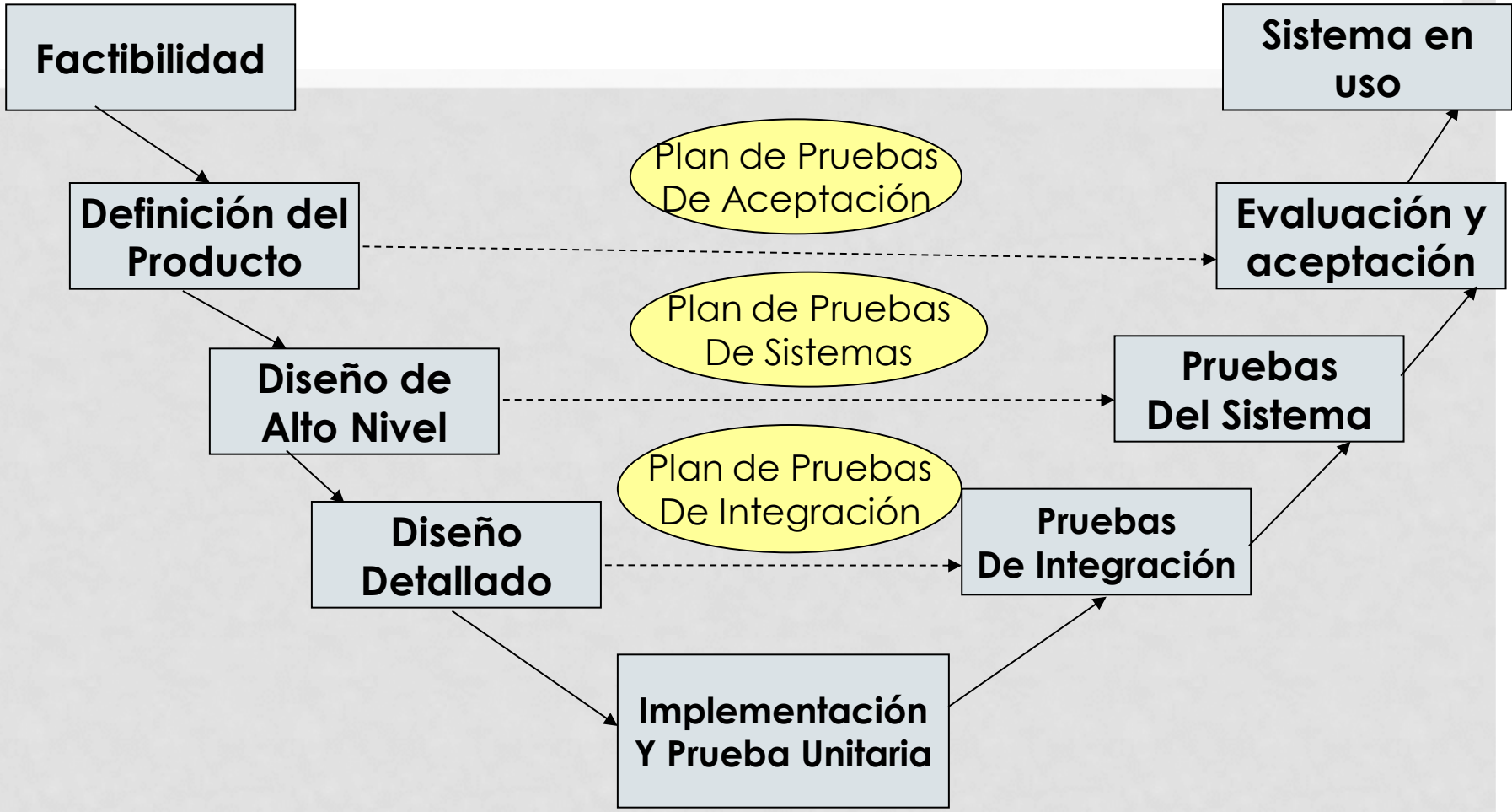
- Críticas:
  - Los proyectos reales raramente pueden seguir el flujo secuencial que se propone.
  - Problemas con requerimientos o diseño ➔ Se resuelven luego de la implementación (con costos inmensamente superiores a su resolución en etapas tempranas).
  - Alta dependencia entre fases ➔ Se tarda mucho tiempo en pasar por todo el ciclo .
  - Producto operativo al final ➔ alta paciencia del cliente. ➔ todo o nada.

- ***Utilidad:*** cuando los requerimientos son bien comprendidos al inicio del proyecto y no son volátiles.

MODELO EN CASCADA CON REALIMENTACIÓN



MODELO EN V



Involucra chequeos y pruebas de cada una de las etapas del modelo de cascada

MODELO EN V

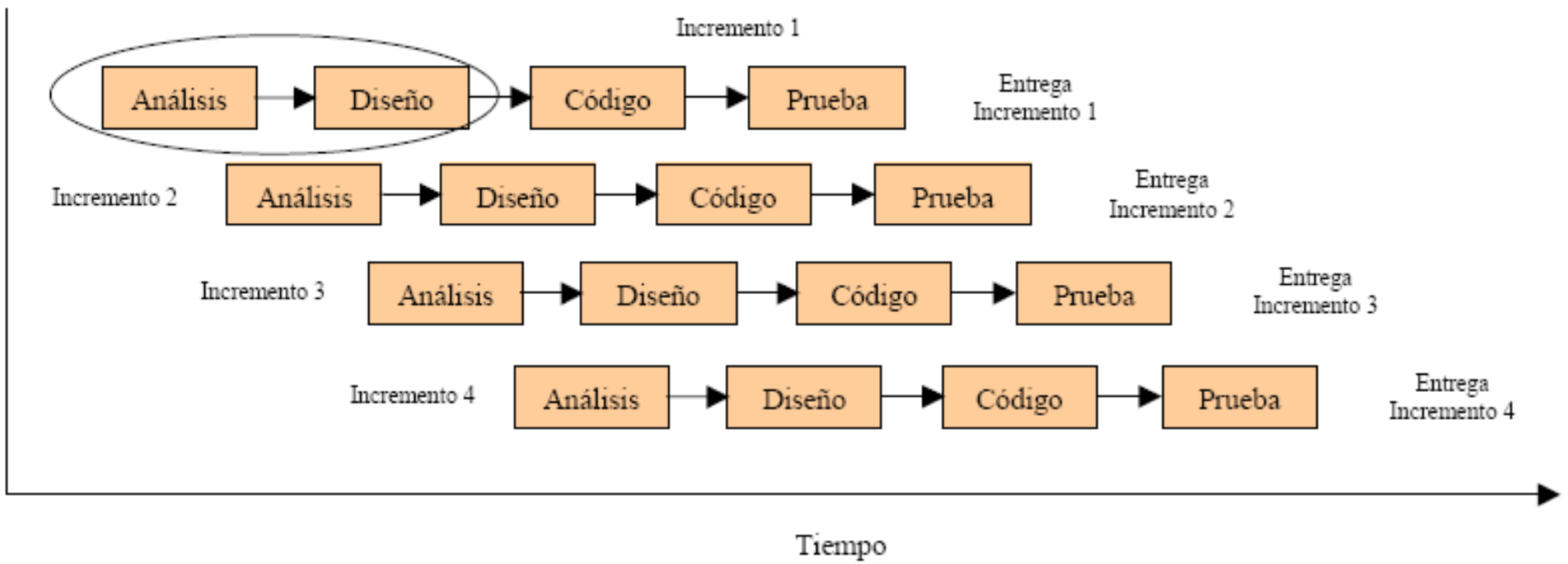
- Una fase además de utilizarse como entrada para la siguiente, sirve para validar o verificar otras fases posteriores.

- Proviene del principio que establece que los procedimientos utilizados para probar si la aplicación cumple las especificaciones ya deben haberse creado en la fase de diseño (o antes).

MODELO EN V

- Desventajas:
  - El riesgo es mayor que el de otros modelos.
    - En lugar de hacer pruebas de aceptación al final de cada etapa, las pruebas comienzan a efectuarse luego de haber terminado la implementación.
  - No contempla la posibilidad de retornar a etapas inmediatamente anteriores.
  - Se toma toda la complejidad del problema de una vez y no en iteraciones o ciclos de desarrollo.
- Ventajas
  - Modelo más robusto y completo que el Modelo de Cascada,
  - Puede producir software de mayor calidad que con el modelo de cascada.

MODELOS INCREMENTALES



MODELOS INCREMENTALES

- Cada secuencia produce un "incremento" de software, según importancia y prioridades del cliente.
- Cada incremento se desarrolla sobre aquél ya entregado.
- Cada iteración o "incremento" produce una versión operativa ➔ se entrega un producto operacional.

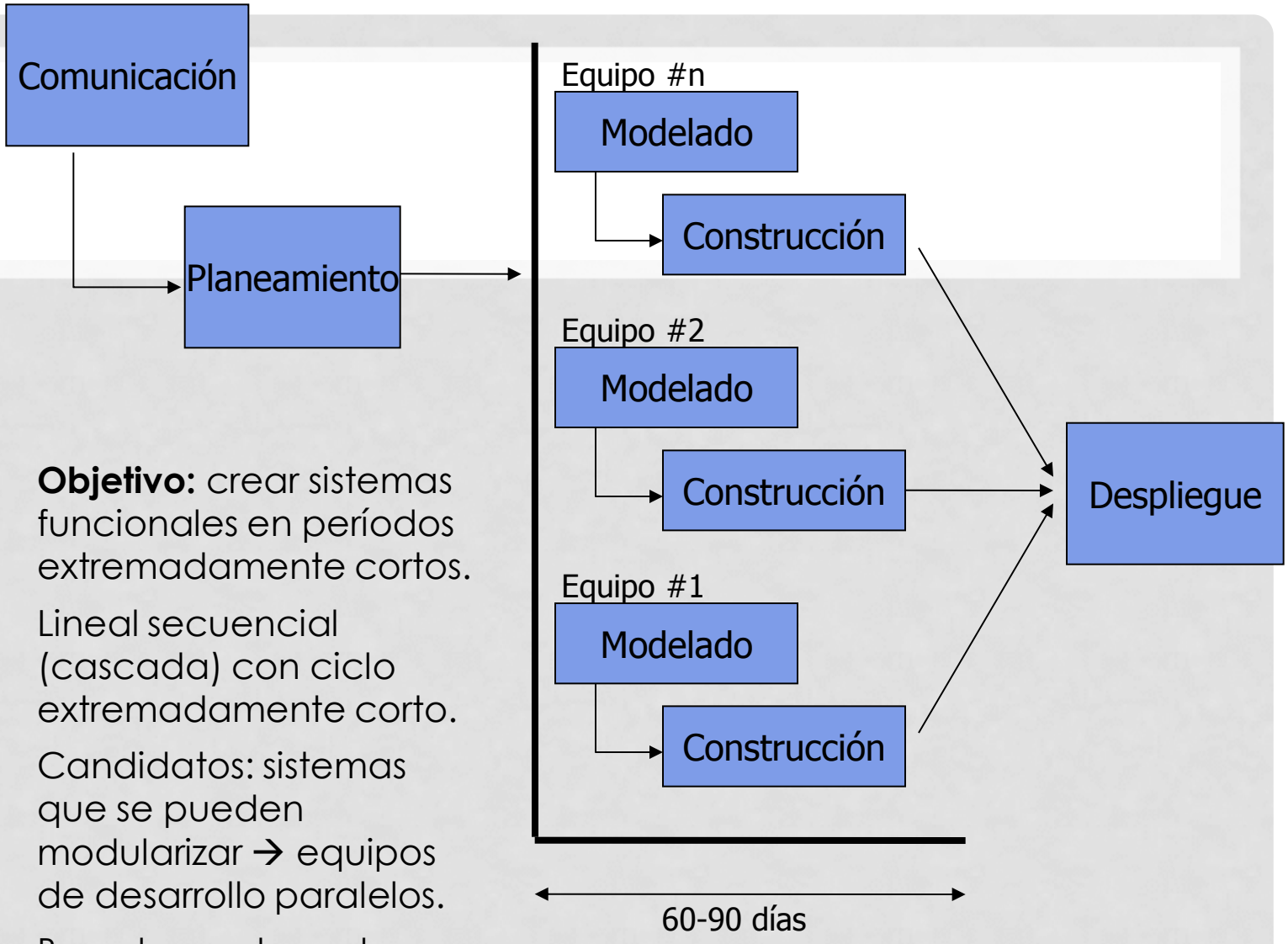
MODELOS INCREMENTALES

- **Incrementos:** proporcionan un subconjunto de funcionalidad ➔ se entrega "*algo de valor*" al cliente con cierta frecuencia.
- El cliente se involucra más ➔ usan los incrementos como prototipos y generan experiencias de validación.
- Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo.
- Los errores en los requisitos se detectan tarde.

MODELOS INCREMENTALES

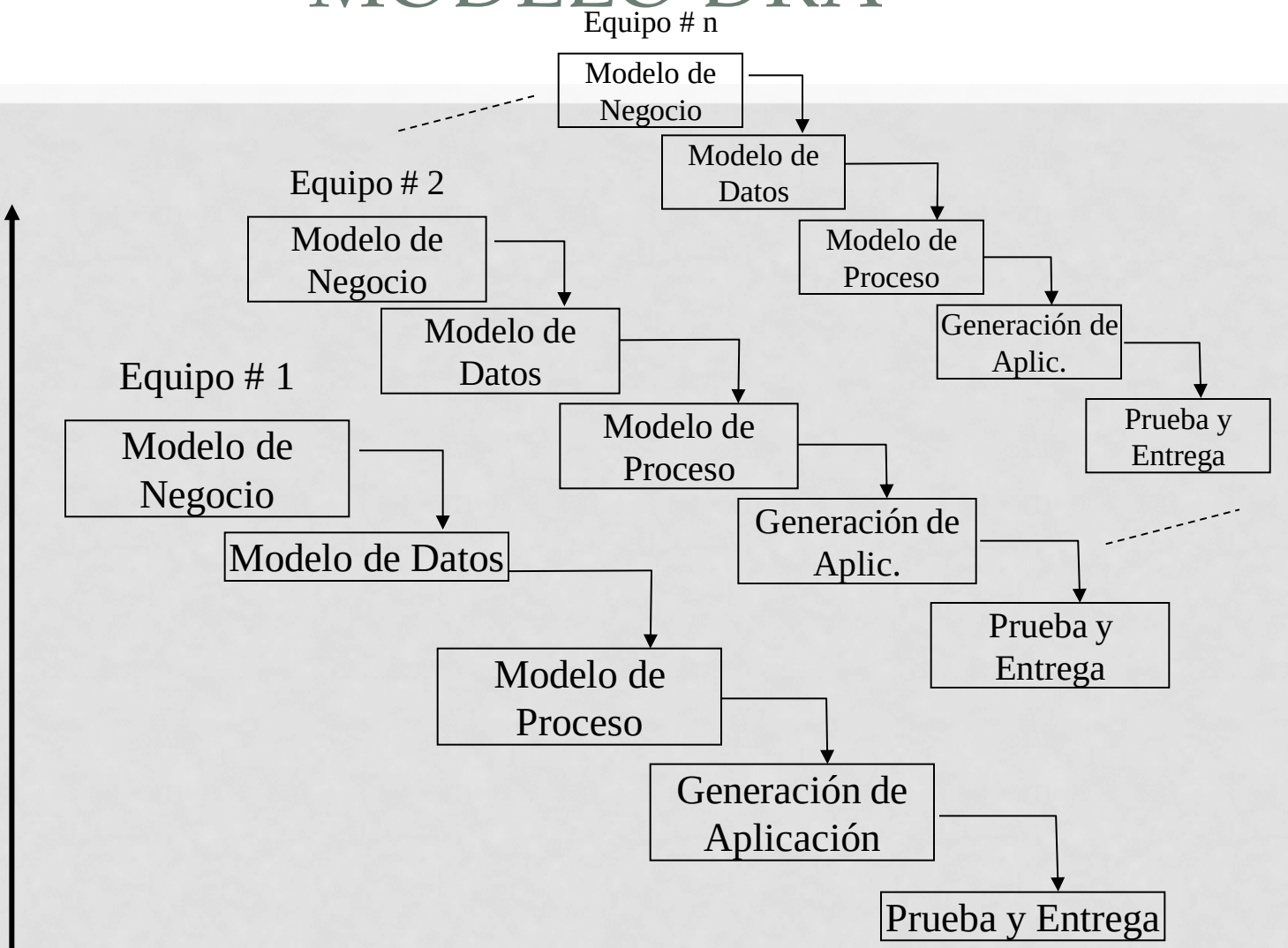
- ***UTILIDAD:***
  - Cuando no se dispone del personal para una instalación completa
  - Cuando se pueden definir incrementos acotados en tamaño, con funcionalidades bien definidas.
  - Cuando no se está seguro de cumplir con plazos de tiempo o se tiene una fecha imposible de cambiar.
  - **No es** recomendable para atributos de calidad críticos.

MODELO DRA: DESARROLLO RÁPIDO DE APLICACIONES



- **Objetivo:** crear sistemas funcionales en períodos extremadamente cortos.
- Lineal secuencial (cascada) con ciclo extremadamente corto.
- Candidatos: sistemas que se pueden modularizar → equipos de desarrollo paralelos.
- Basado en el uso de componentes.
- Enfatiza la reutilización.

MODELO DRA


Equipo # n
Modelo de Negocio
Modelo de Datos
Modelo de Proceso
Generación de Aplic.
Prueba y Entrega
Equipo # 2
Modelo de Negocio
Modelo de Datos
Modelo de Proceso
Generación de Aplic.
Prueba y Entrega
Equipo # 1
Modelo de Negocio
Modelo de Datos
Modelo de Proceso
Generación de Aplicación
Prueba y Entrega
<----------60-90 días---------->
Vera - Colussi

MODELO DRA

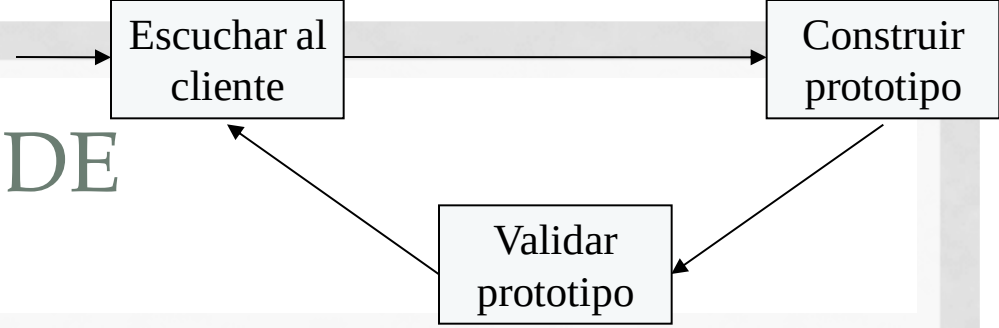
- Candidatos DRA: Aplicaciones modularizables, cuyas funciones principales puedan completarse en menos de 3 meses.
- Críticas:
  - Proyectos grandes → gran número de personas.
  - Requiere alto compromiso de desarrolladores y clientes (actividades rápidas).
  - No apto para todo tipo de sistema (ej. no modularizable, bajo nivel de reuso de componentes).
  - Si se quisiera alcanzar alta performance 'poniendo a punto' las interfaces de las componentes del sistema,
  - Desaconsejable cuando existen altos riesgos técnicos (nuevas tecnologías o interoperabilidad con software existente).

MODELO DE PROCESOS EVOLUTIVOS

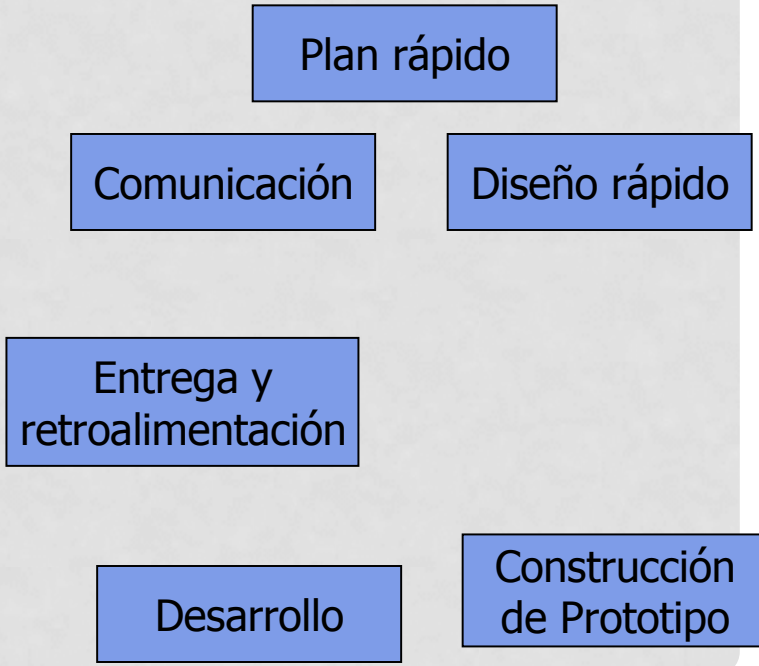
MODELOS EVOLUTIVOS

- Se adaptan más fácilmente a los cambios introducidos a lo largo del desarrollo.
- Iterativos/(¿Incrementales?)
  - Permiten repetir las secuencias de ejecución de etapas/fases
  - En cada iteración se obtienen versiones más completas del software.
- Modelos Evolutivos:
  - Construcción de Prototipos
  - Modelo en Espiral
  - Modelo Concurrente

CONSTRUCCIÓN DE PROTOTIPOS



- No están claros los requerimientos al inicio.
  - Sistemas nuevos, poco conocimiento del dominio
- ➔ Útil cuando hay inseguridades del lado cliente y/o desarrollador.
- Reduce el riesgo.
- Especificación: desarrollo creciente.
- No modifica el flujo del ciclo
- de vida.
- Una vez identificados los requerimientos, se
- construye el producto.



CONSTRUCCIÓN DE PROTOTIPOS

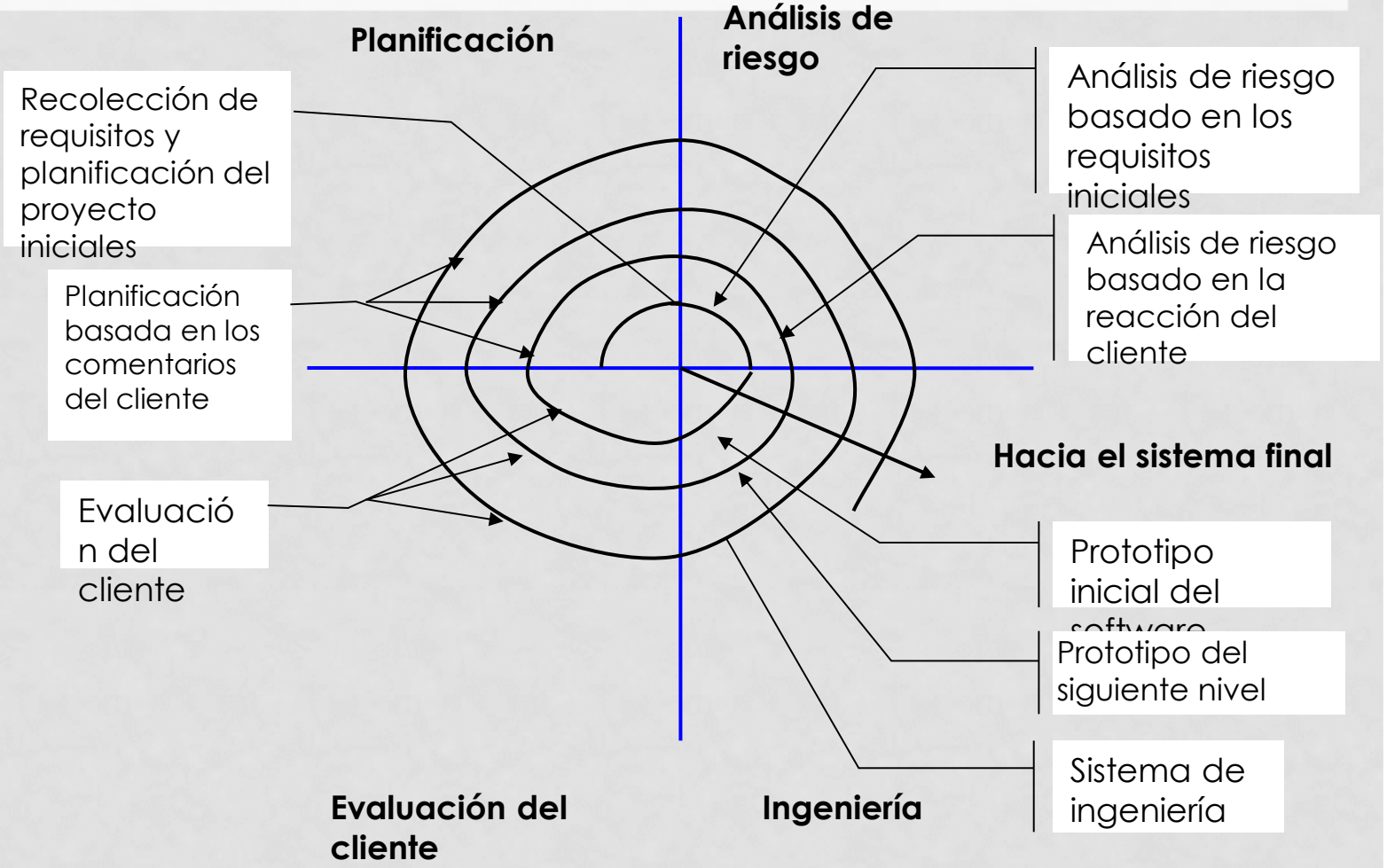
- ***Críticas***:
  - Cliente puede creer que es el sistema.
  - Peligro de malas elecciones iniciales (*quick and dirty*) (compromisos de calidad, implementación, mantenimiento)…
- ***Utilidad***:
  - Usar cuando inicialmente no están claros los *requisitos*.
  - Definir claramente al inicio las reglas de juego con el cliente ➔ prototipo para definir requerimientos

CONSTRUCCIÓN DE PROTOTIPOS

- Para que sea efectiva:
  - Debe ser un sistema con el que se pueda experimentar.
  - Debe ser barato en relación al desarrollo del sistema.
  - Debe poder desarrollarse rápidamente.
  - Énfasis en la interfaz de usuario.
  - Equipo de desarrollo reducido.
  - Herramientas y lenguajes adecuados.

- ***"El prototipado es un medio excelente para recoger el 'feedback' (realimentación) del usuario final"***

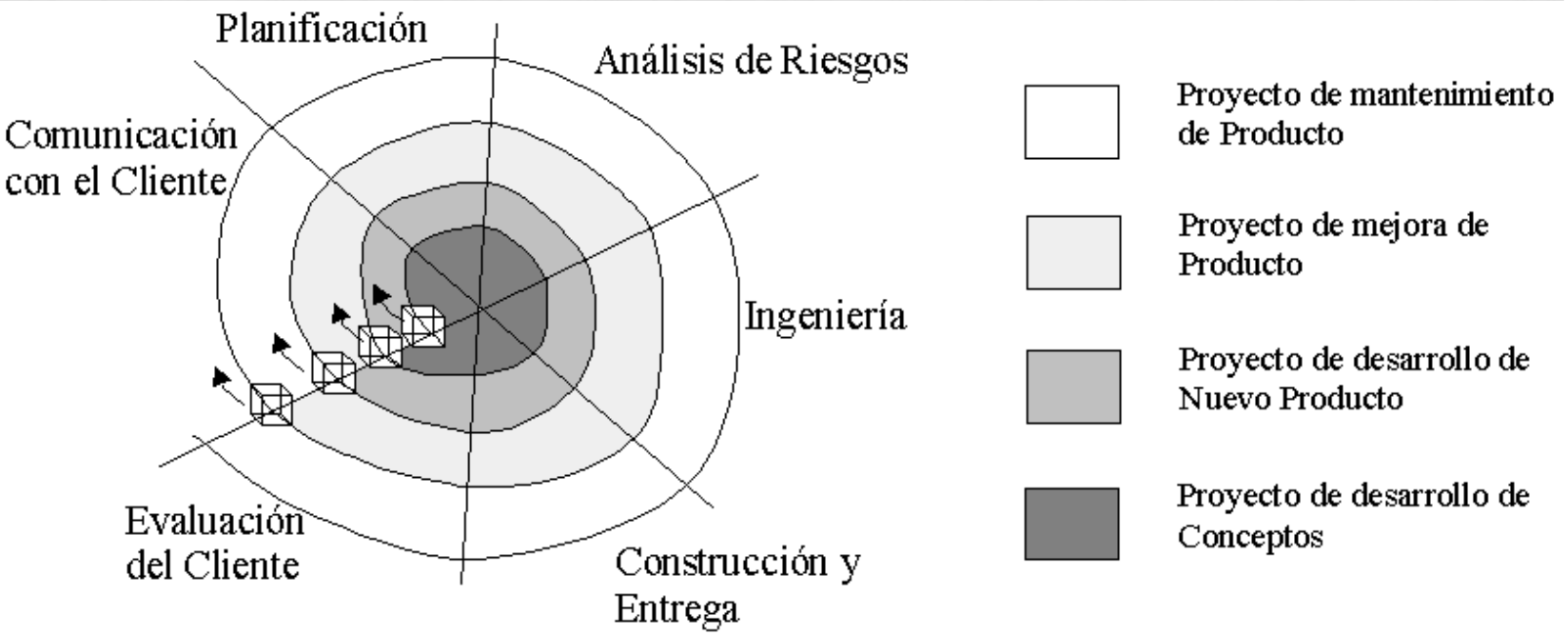
MODELO EVOLUTIVO EN ESPIRAL



MODELO EVOLUTIVO EN ESPIRAL

- Comenzar produciendo una pequeña parte del sistema (completamente funcional)
- Una vez completada, se procede a crear una segunda parte, acoplada a la primera.

➔ En cada iteración se obtiene una versión aumentada del sistema hasta terminar.



MODELO EVOLUTIVO EN ESPIRAL

- Este modelo a diferencia de los otros *toma en consideración explícitamente el riesgo* ➔ actividad importante en la administración del proyecto.
- Cada ciclo empieza identificando:
  - Los objetivos de la porción correspondiente.
  - Las alternativas ➔ al compararlas con los objetivos se analizan los riesgos.
  - Las restricciones.
- Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente.

MODELO EVOLUTIVO EN ESPIRAL - FASES

- *Definición de objetivos:*
  - Se definen los objetivos, las restricciones del proceso y del producto.
  - Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.
- *Evaluación y reducción de riesgos:*
  - Se realiza un análisis detallado de cada riesgo identificado.
  - Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos.
  - Se llevan a cabo los pasos para reducir los riesgos.

MODELO EVOLUTIVO EN ESPIRAL - FASES

- *Desarrollo y validación*:
  - Se escoge el modelo de desarrollo después de la evaluación del riesgo.
  - El modelo que se utilizará (cascada, sistemas formales, evolutivo, etc.) depende del riesgo identificado para esa fase.
- *Planificación*:
  - Se determina si continuar con otro ciclo.
  - Se planea la siguiente fase del proyecto.

MODELO EN ESPIRAL - VENTAJAS

- La calidad y gestión de riesgo son el primer objetivo.
- Permite usar el prototipado en todas las etapas de la evolución para reducir el riesgo.
- Mantiene el enfoque sistemático de los pasos sugeridos por el modelo cascada, pero lo incorpora dentro de un marco iterativo más real.

MODELO EN ESPIRAL - PROBLEMAS

- Requiere de experiencia en la identificación de riesgos.
- Difícil de convencer a los clientes de que es controlable.
- Requiere mucha habilidad para el análisis de riesgos y de esta habilidad depende su éxito.

PROBLEMAS DE LOS MODELOS EVOLUTIVOS

- La planificación es un problema:
  - número incierto de ciclos de iteración requeridos para completar el software.
  - Las técnicas de gestión y estimación se basan en configuraciones lineales de las actividades.
- Los procesos evolutivos no establecen la velocidad máxima de la evolución: calibrar el ritmo es un problema.
  - Demasiado rápido → Caos
  - Demasiado lento → Baja productividad
- Se enfocan en flexibilidad y extensibilidad y no en la alta calidad (cero defecto).

MODELOS ESPECIALIZADOS

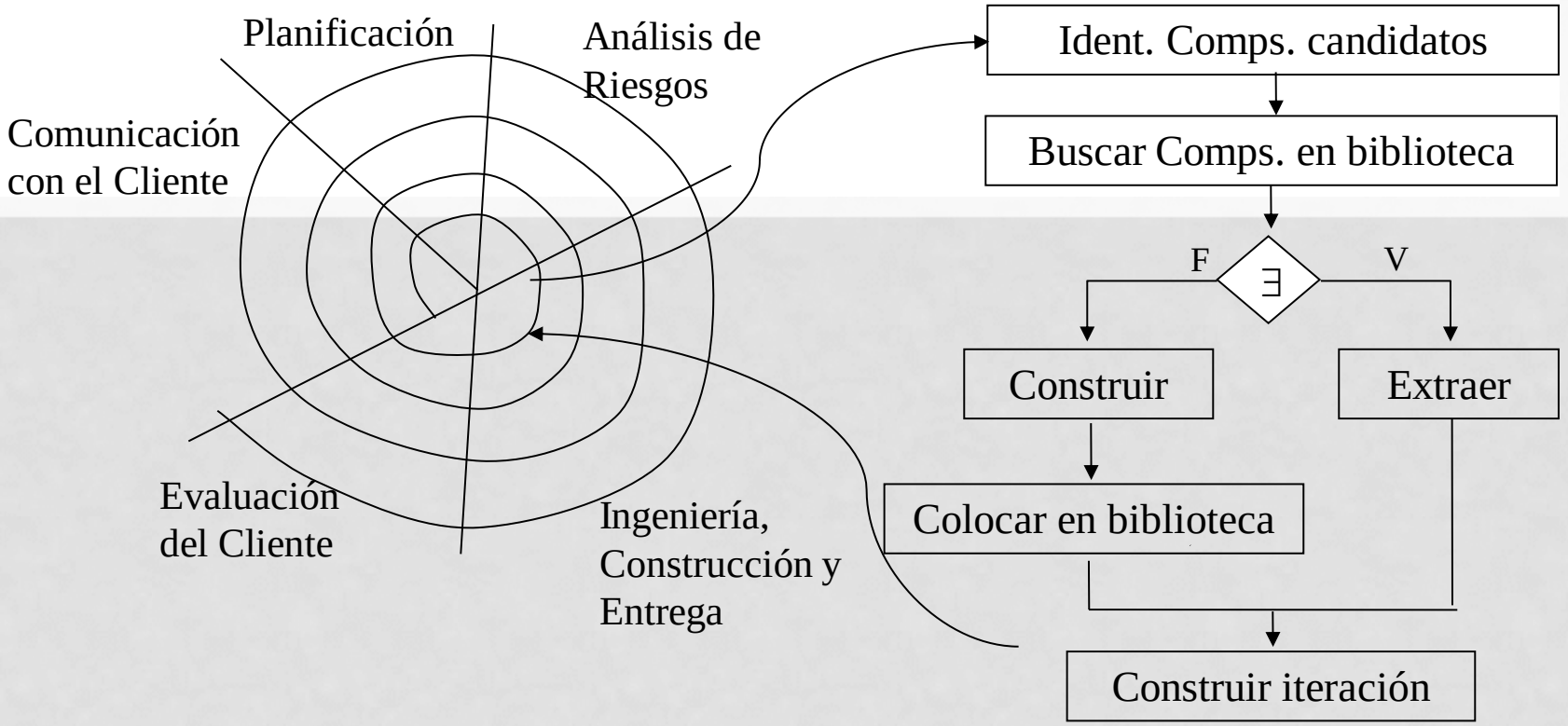
DESARROLLO BASADO EN COMPONENTES

- Incorpora características del modelo en espiral y es evolutivo por naturaleza: Enfoque iterativo para la creación del SW.
- Enfatiza la reusabilidad.
- Configura aplicaciones desde componentes preparados de SFW.
- 70% de reducción del tiempo del ciclo de desarrollo
- 84% de reducción del costo del proyecto.

DESARROLLO BASADO EN COMPONENTES

Pasos

1. Se hace una investigación de los componentes disponibles.
2. Se consideran los aspectos de integración de componentes.
3. Se diseña una arquitectura de software para acoplar los componentes.
4. Los componentes se integran en la arquitectura
5. Se realizan pruebas detalladas.



Basado en modelo en espiral (evolutivo e iterativo)

DESARROLLO BASADO EN COMPONENTES

DESARROLLO BASADO EN COMPONENTES

- Ventaja:
  - Reducen la cantidad de software a ser desarrollado
  - ➔ reducen costos, riesgos, tiempo de entrega.
- Desventaja:
  - Compromisos en los requerimientos.
  - Evolución de componentes, módulos o subsistemas.

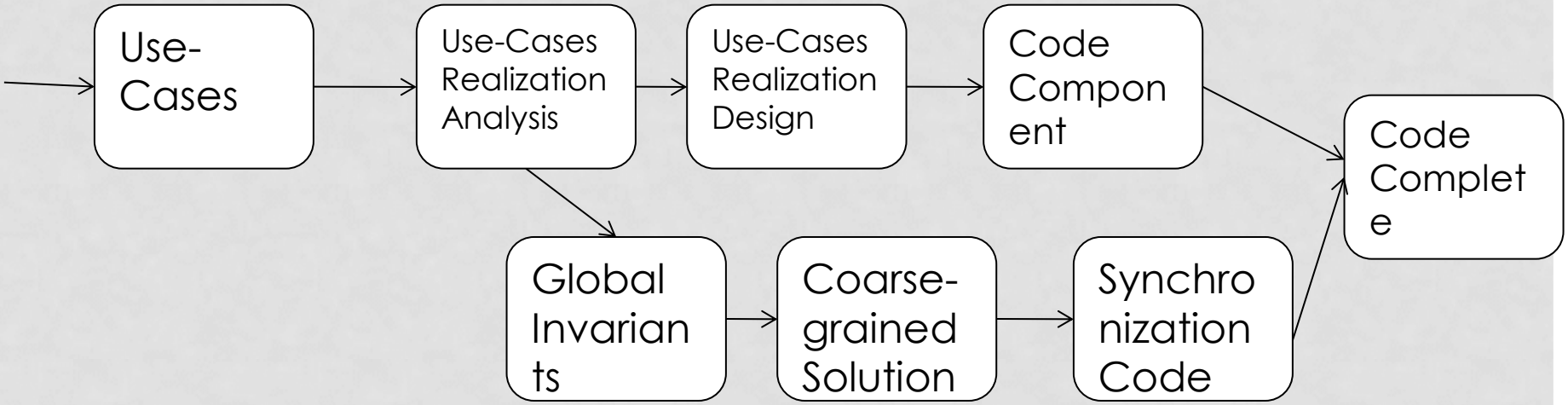
DESARROLLO DE SOFTWARE ORIENTADO A ASPECTOS

- Conforme los sistemas se vuelven más complejos y elaborados
  - Ciertos 'intereses' o 'aspectos' abarcan toda la arquitectura.
- Ejemplo:
  - Seguridad
  - Tolerancia a fallas
  - Reglas de negocio
  - Sincronización de tareas
  - Gestión de la memoria
  - Etc.

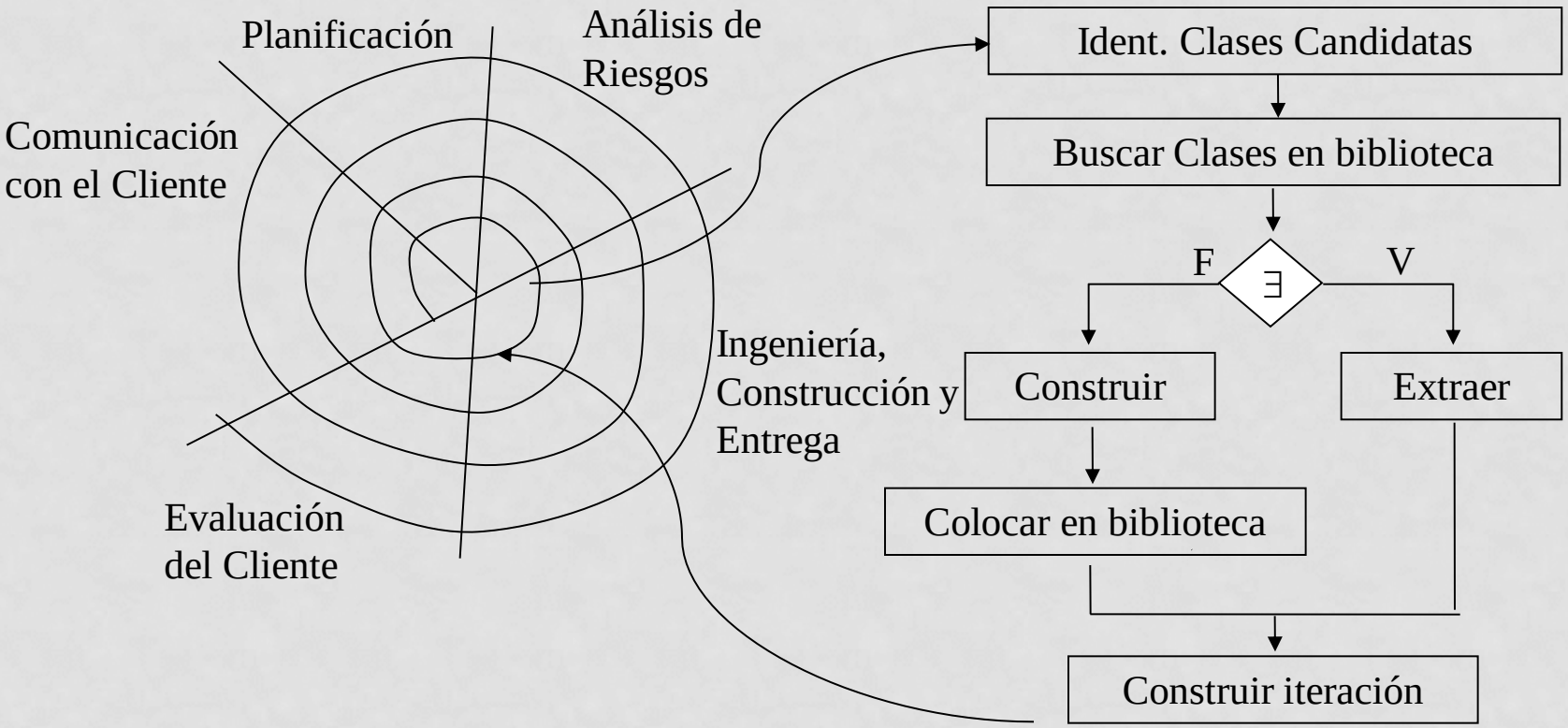
DESARROLLO DE SOFTWARE ORIENTADO A ASPECTOS

- Intereses *transversales*
  - Múltiples funciones, características e información
- Desarrollo de Software Orientado a Aspectos o Programación Orientada a Aspectos
  - Proceso y enfoque metodológico para *definir*, *especificar*, *diseñar* y *construir* aspectos.
- El *modelo en espiral* es apropiado cuando se identifican y construyen los aspectos.
- El *desarrollo concurrente* es apropiado para el desarrollo independiente de los aspectos.

DESARROLLO DE SOFTWARE ORIENTADO A ASPECTOS



CICLO DE VIDA ORIENTADO A OBJETOS



PROCESO UNIFICADO (RUP)

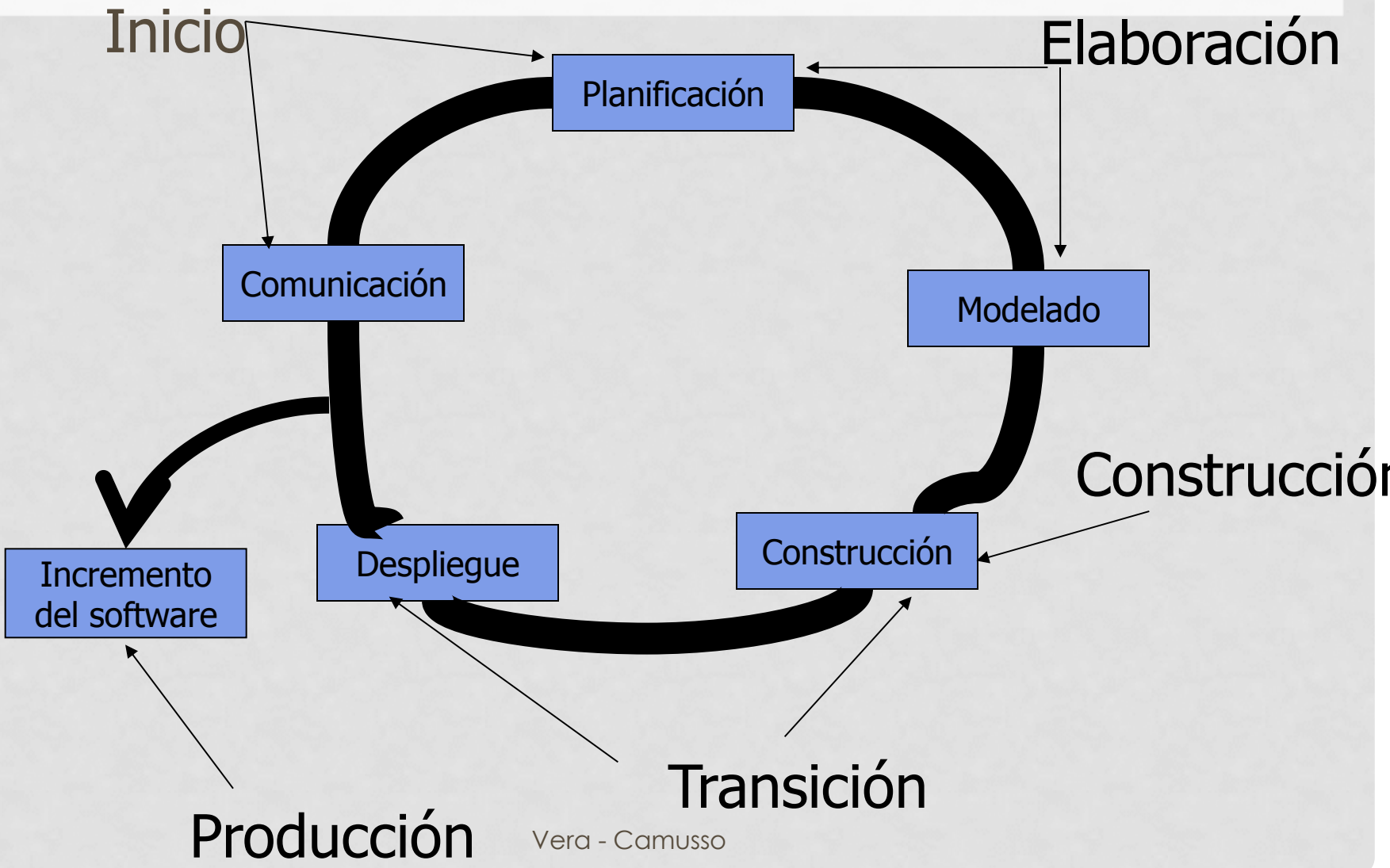
PROCESO UNIFICADO RACIONAL (RUP)

- Rumbaugh, Booch y Jacobson establecieron el UML (Unified Modeling Language) a principios de los '90, y se convirtió en un estándar de la industria en 1997 para desarrollo orientado a objetos.
- UML originalmente no incorpora la definición del proceso de software
- En los años siguientes los autores desarrollaron el PU, basado en UML.
- Es un modelo de proceso de propiedad comercial.

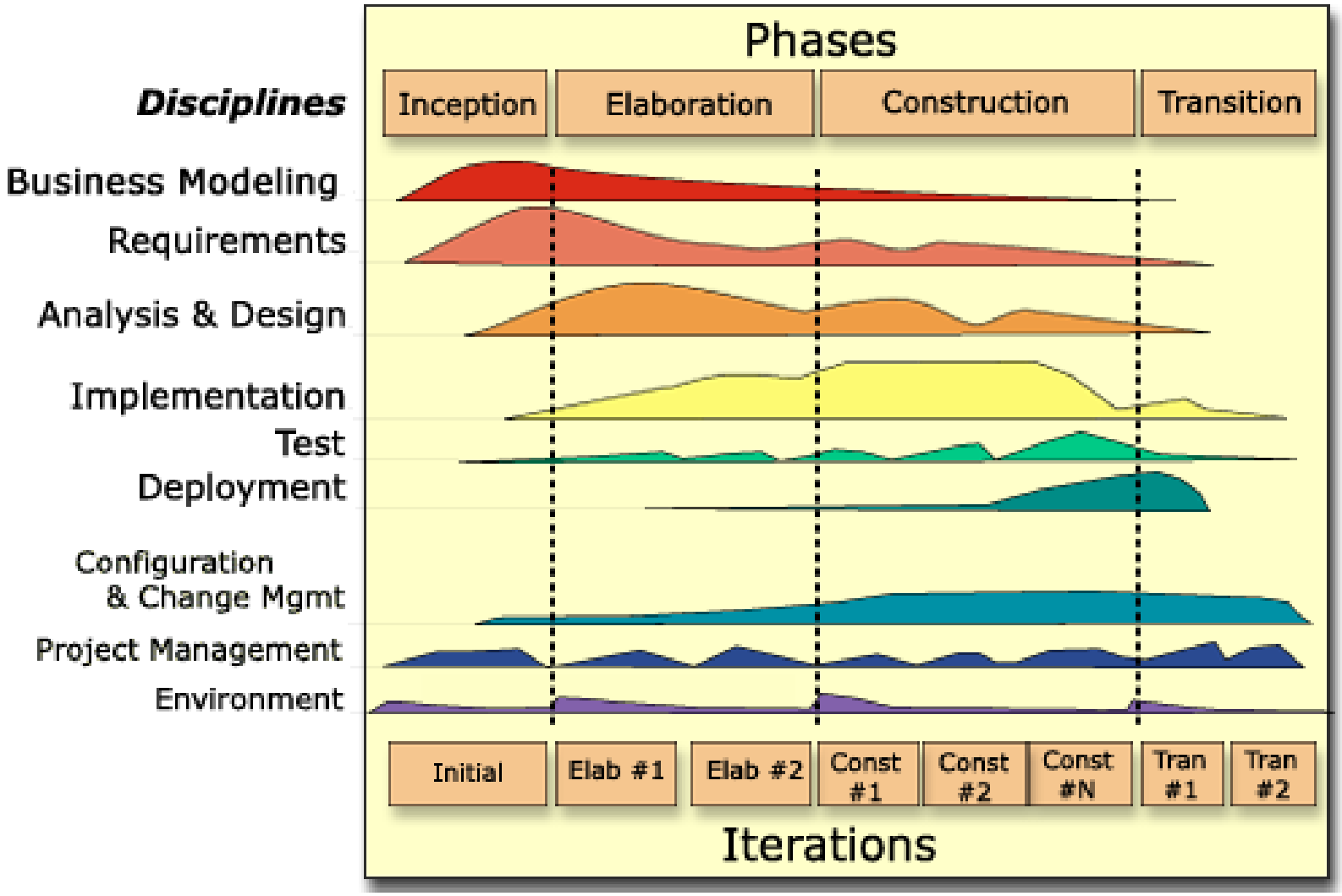
RUP

- Define un proceso de software guiado por casos de uso, centrado en la arquitectura, iterativo e incremental.
- Integra un conjunto de "buenas prácticas" para el desarrollo de software válido para varios tipos de proyectos y organizaciones.
  - Desarrollo iterativo.
  - Gestión de requerimientos.
  - Uso de arquitecturas basadas en componentes.
  - Uso de técnicas de modelado visual
  - Verificación continua de calidad.
  - Control y gestión de cambios

PROCESO UNIFICADO FASES



PROCESO UNIFICADO FASES



MÉTODOS ÁGILES

MÉTODOS ÁGILES

- Surgen en 2000 como alternativa a los procesos de desarrollo de software tradicionales:
  - caracterizados por ser rígidos y
  - dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.
- Punto de partida: Manifiesto Ágil ➔ documento que resume la "filosofía ágil".

METODOLOGÍAS ÁGILES

- **Valorar al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas**.
  - La gente es el principal factor de éxito de un proyecto software.

- **Desarrollar software que funciona más que conseguir una buena documentación.**
  - No producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante.

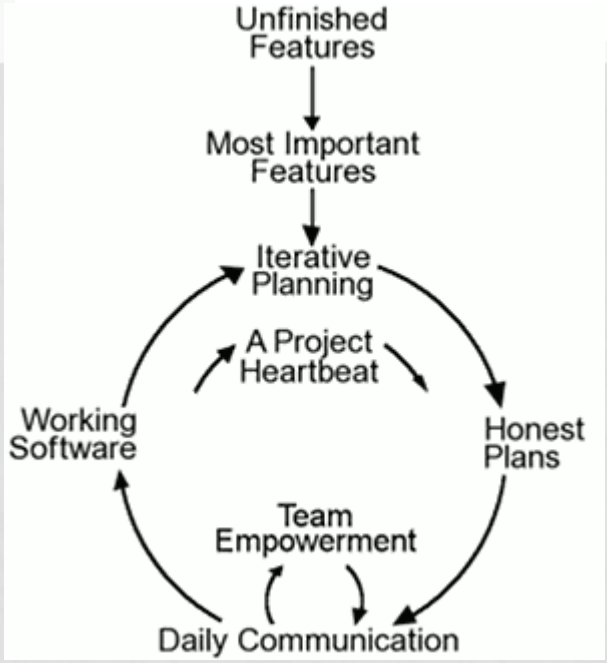
METODOLOGÍAS ÁGILES

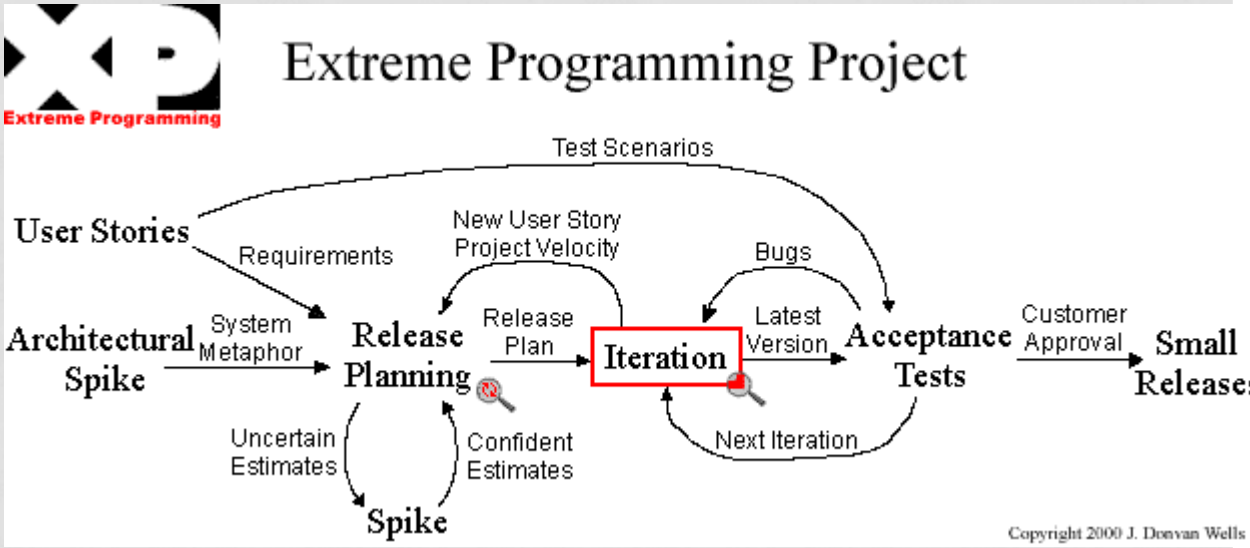
- **La colaboración con el cliente más que la negociación de un contrato.**
  - Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo.
- **Responder a los cambios más que seguir estrictamente un plan.**
  - La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo.

METODOLOGÍAS ÁGILES

- **Programación Extrema (XP – Extreme Programming)**
  - Metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software.
  - Promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores
  - Realimentación continua entre el cliente y el equipo de desarrollo.
  - Comunicación fluida entre todos los participantes
  - Simplicidad en las soluciones implementadas.
  - Flexibilidad para enfrentar los cambios.
  - ***UTILIDAD:*** Requerimientos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

METODOLOGÍAS ÁGILES XP

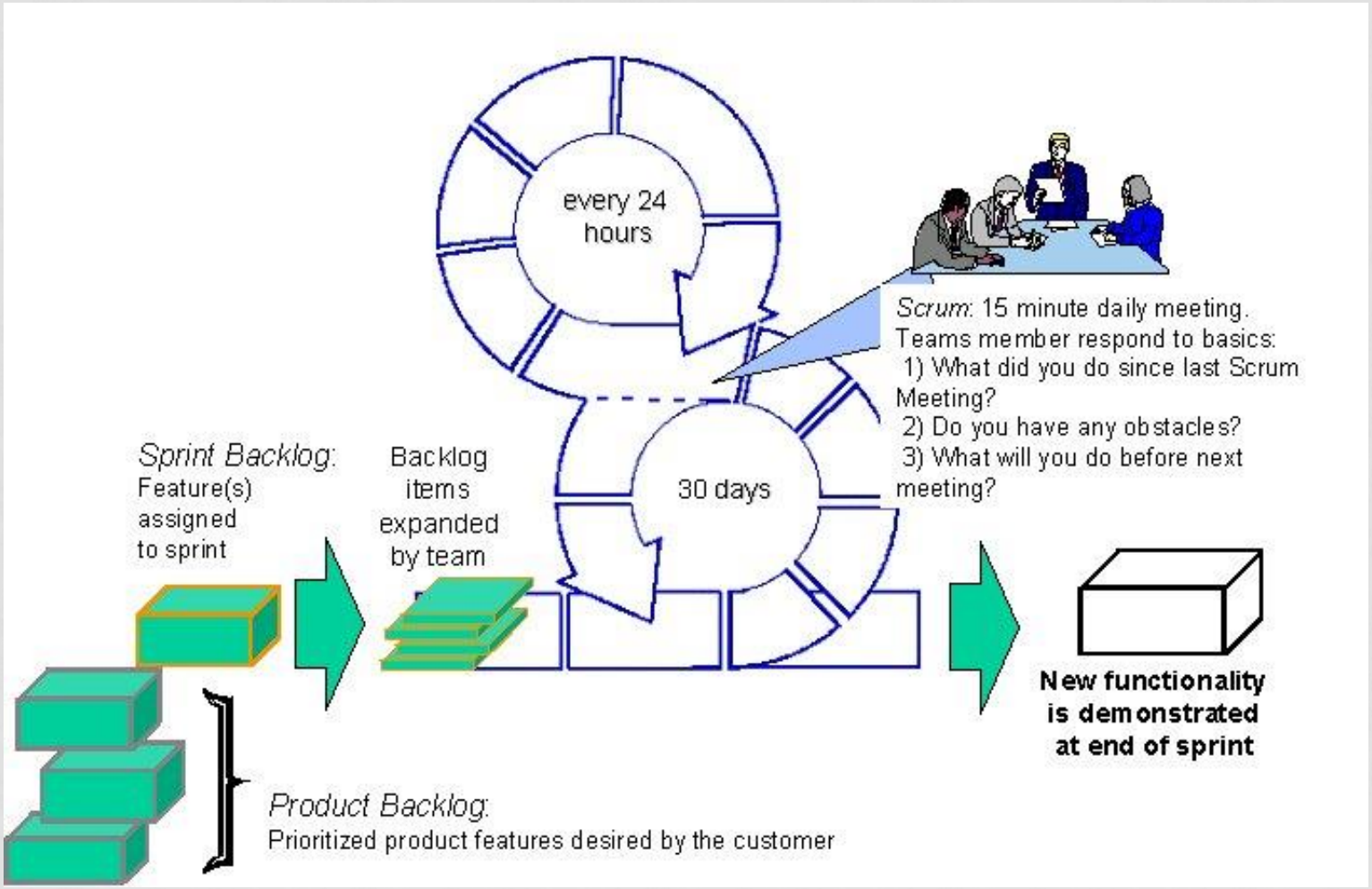




METODOLOGÍAS ÁGILES

- **SCRUM.**
  - Define un marco para la gestión de proyectos.
  - Está especialmente indicada para proyectos con un rápido cambio de requisitos.
  - Características:
    - El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
    - Promueve las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

METODOLOGÍAS ÁGILES - SCRUM



OTROS CICLOS DE VIDA

- Ingeniería Inversa
- Reingeniería
- MDA (Model-Driven Architecture)
- TDD (Test Driven Development)
- Etc.
- Etc.
- Etc.
- Etc.
- …

IMPORTANCIA DEL CICLO DE VIDA

- Para tener éxito en un proyecto, tanto los administradores de proyecto como todos los miembros del equipo de proyecto (además de los clientes y gerentes) deben contar con herramientas de comunicación tales como una terminología común y un acuerdo respecto al ciclo de vida a utilizar.
- El ciclo de vida es la guía que siguen todos los *stakeholders* y que posibilita conocer en todo momento dónde se encuentra el proyecto.

IMPORTANCIA DEL CICLO DE VIDA

- Cada proyecto tiene su "óptimo".
- ¿"iterativo" o "incremental"?
  - ¿Hay margen para especular en la fase de desarrollo?
  - ¿O hay que entregar algo desde el primer día?
- ¿Son entendidos los requerimientos?
- ¿Cuáles son los riesgos?
- ¿Hay una fecha fija de terminación?
- ¿Qué tan experimentado es el equipo y el cliente?
- ¿La tecnología es probada?
- ¿Requerimientos de calidad?

SELECCIÓN Y ADAPTACIÓN DE UN MCV

- La selección del MCV más apropiado es el primer paso, el siguiente es su ajuste o adaptación a las necesidades del proyecto.
- Un método frecuente de adaptación es el de realizar combinaciones entre diferentes modelos.

PROCESO DE SELECCIÓN Y ADAPTACIÓN DE UN MCV

Proceso de selección:

1. Familiarizarse con los distintos modelos.
2. Examinar las siguientes características:
   a. Requerimientos
   b. Equipo de Proyecto
   c. Comunidad de usuarios
   d. Tipo de proyecto (desarrollo, mejoramiento, mantenimiento, etc.)
   e. Riesgos

PROCESO DE SELECCIÓN Y ADAPTACIÓN DE UN MCV

Proceso de selección:

3. Revisar como se ajusta el CV con los estándares de la organización, clientes, tipo de proyecto y demás.
4. Identificar un conjunto de fases y actividades.
5. Establecer los entregables internos y externos.
6. Definir plantillas y guías para los entregables.
7. Determinar revisiones, inspecciones, puntos de verificación y validaciones e hitos.
8. Evaluar la efectividad del CV e implementar mejoras en caso de ser necesario.

BIBLIOGRAFÍA OBLIGATORIA

- [Pressman, 2005] Pressman, Roger S. "Ingeniería del Software: un Enfoque Práctico". **Capítulo 2, 3 y 4**. 6ta. Edición. Mc Graw-Hill. 2005.
- [Sommerville, 2011] Sommerville, Ian. "Ingeniería de Software". **Capítulo 2 y 3**. Novena Edición. Pearson Educación, 2011.

BIBLIOGRAFÍA COMPLEMENTARIA

- ISO/IEC 12207-2008
- IEEE 1074-2006
- [Canós y otros, 2003] Canós, J., Letelier, P., Penadés, M. Metodologías ágiles en el desarrollo de software. En: Proceedings VIII Jornadas de Ingeniería del software y Bases de Datos, JISBD 2003. Alicante, Noviembre de 2003.
- "Architecture-Based Development", Len Bass, Rick Kazman, Technical Report, 1999.
- UML Components – A Simple Process for Specifying Component-Based Software. John Chessman – John Daniels, Addison-Wesley, 2001
- A methodology for secure software design, Eduardo B. Fernández

PROPUESTAS PARA LA PRÓXIMA CLASE

- Método ágil: TDD, casos de éxito de uso.
- RUP: origen, casos de éxito de uso.

¿Quién desea investigar alguno de estos temas?

OBJETIVOS DE LA CLASE

- ✓ **Modelos prescriptivos de procesos.**
  - ✓ **Code-and-fix**
  - ✓ **Cascada.**
  - ✓ **Modelo en V.**
  - ✓ **Modelos incrementales.**
  - ✓ **Modelos evolutivos.**
  - ✓ **Modelos especializados.**
  - ✓ **Proceso Unificado (RUP)**
  - ✓ **Métodos Ágiles.**

UN POCO DE HUMOR…

¿Dudas, consultas?