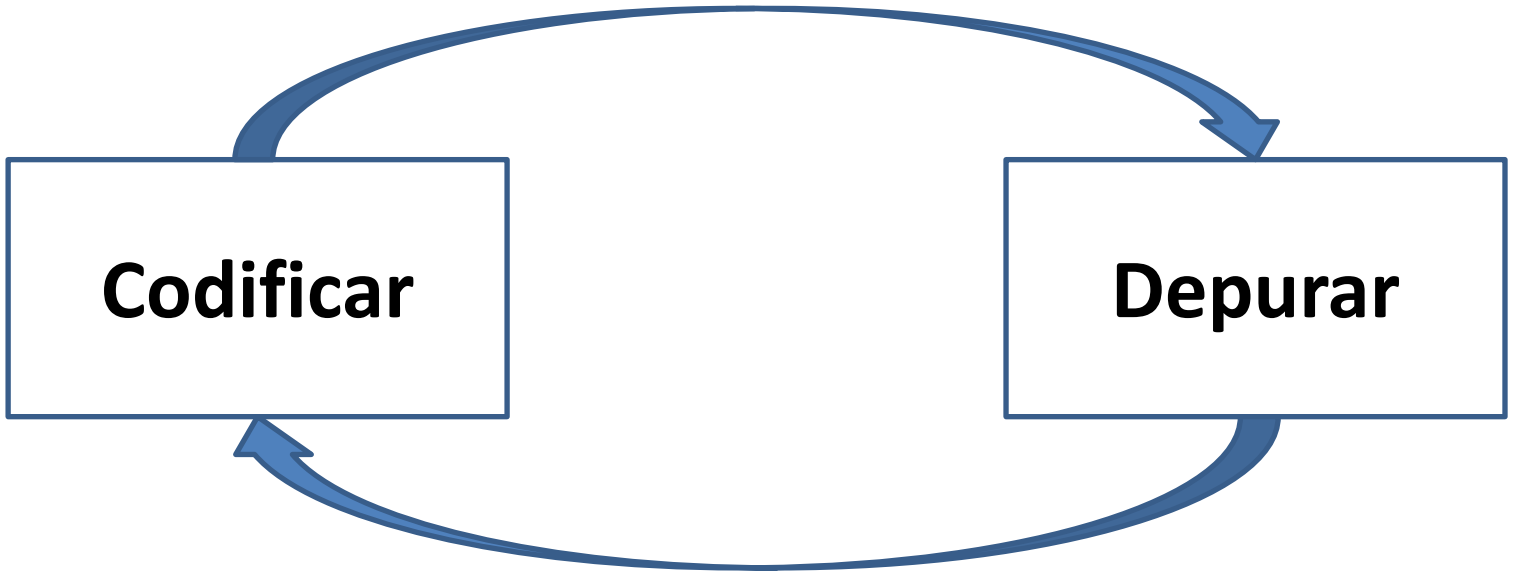
Ingeniería de Software I

Carlos Monsalve
monsalve@espol.edu.ec

Sección 2: Modelos de Desarrollo de Software

Software

¿Cómo lo desarrollamos?

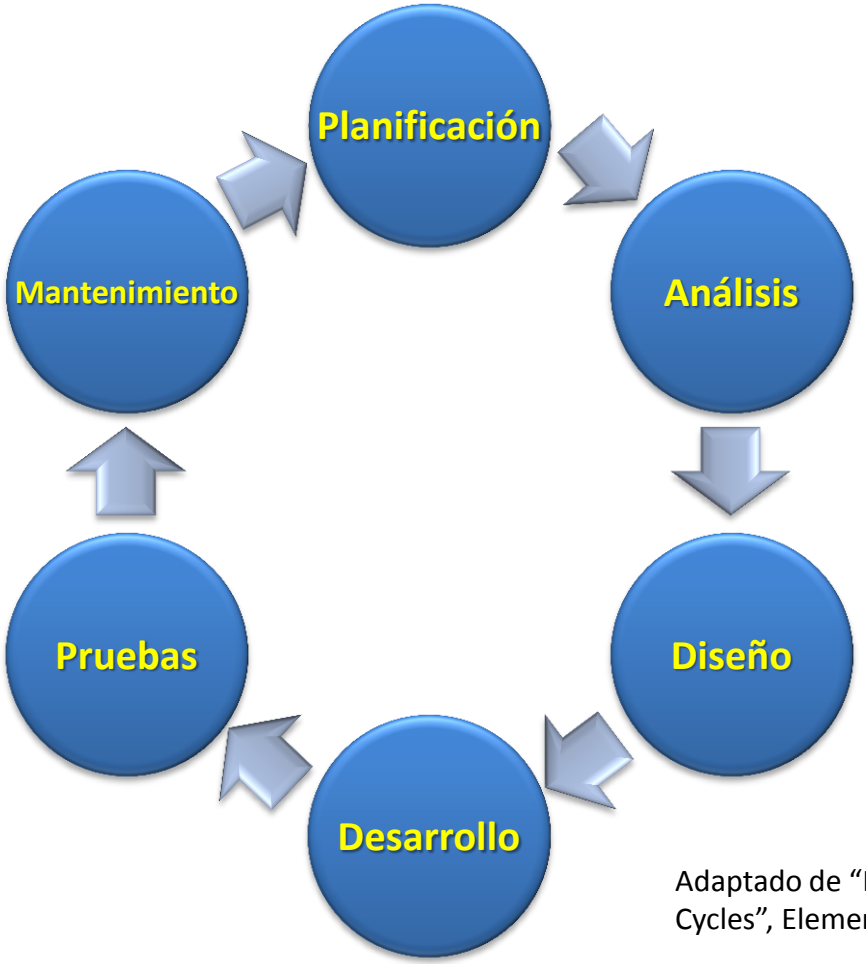


Software | ¿Cómo desarrollarlo?

Software | Respuesta

- Ciclo de vida de desarrollo de software
  - Proceso estructurado y organizado
- Busca garantizar
  - Calidad
    - Tiempos
    - Costos
    - Satisfacción del cliente
  - Eficiencia

Ciclo de Vida | Etapas clásicas



Adaptado de “Introduction to Software Life Cycles”, Element K, 2012

Ciclo de Vida | Planificación

- Análisis del valor del negocio
  - Costo del proyecto
  - Retorno de la inversión
- Elaboración de plan
  - Calendario de actividades
  - Recursos (financieros, humanos, tecnológicos)
  - Presupuesto
- Negociación y aprobación

Ciclo de Vida | Análisis

- Requerimientos
  - Levantamiento
  - Análisis: claridad, completos, dependencias
  - Validación
  - Documentación: especificaciones de requerimientos

- Análisis y definición de:
  - Entradas
  - Salidas
  - Funcionalidades

☑ ¿Qué hacer?

☒ ¿Cómo hacerlo?

Ciclo de Vida | Diseño

- Requerimientos ➡ Especificaciones de diseño
  - Instrucciones para los codificadores
  - ¿Cómo cumplir con cada requerimiento?
  - Componentes a desarrollar
  - ¿Qué tecnologías usar?
  - Arquitectura del sistema

☒ ¿Qué hacer?

☑ ¿Cómo hacerlo?

Ciclo de Vida | Desarrollo

- Diseño ➡ Modelo de trabajo
  - Definir equipos de desarrollo
  - Se desarrollan los componentes (módulos)
  - Integración de componentes ➡ Sistema
  - Elaboración de plan de pruebas
    - Sistema
    - Integración
    - Componentes
  - Documentación de pruebas

Ciclo de Vida | Pruebas

- Objetivos:
  - Encontrar defectos
  - Probar la calidad del sistema
- Plan de pruebas ➡ Actividades de pruebas
  - Pruebas de componentes (unitarias)
  - Pruebas de integración: interfaces, interacciones
  - Pruebas de sistema: interacción con otros sistemas
- Errores son corregidos por desarrolladores

Ciclo de Vida | Mantenimiento

- Actividades:
  - Corrección de problemas
  - Nuevas versiones
  - Soporte a usuarios
- Termina cuando el producto de software muere

Ciclo de Vida | ¿Modelo único?

- Modelo debe responder a limitaciones de proyecto:
  - Tiempo
  - Costos
  - Recursos humanos disponibles
  - Cultura organizacional
- No existe un modelo perfecto para todo proyecto

Ciclo de Vida | Ejemplos de Modelos

- Modelo en Cascada
- Modelo V
- Modelos basados en Prototipo
- Modelos Iterativos

Modelo en Cascada

Etapas

HERENCIA DE HARDWARE

Adaptado de "Introduction to Software Life Cycles", Element K, 2012

Modelo en Cascada

Evaluación

Ventajas

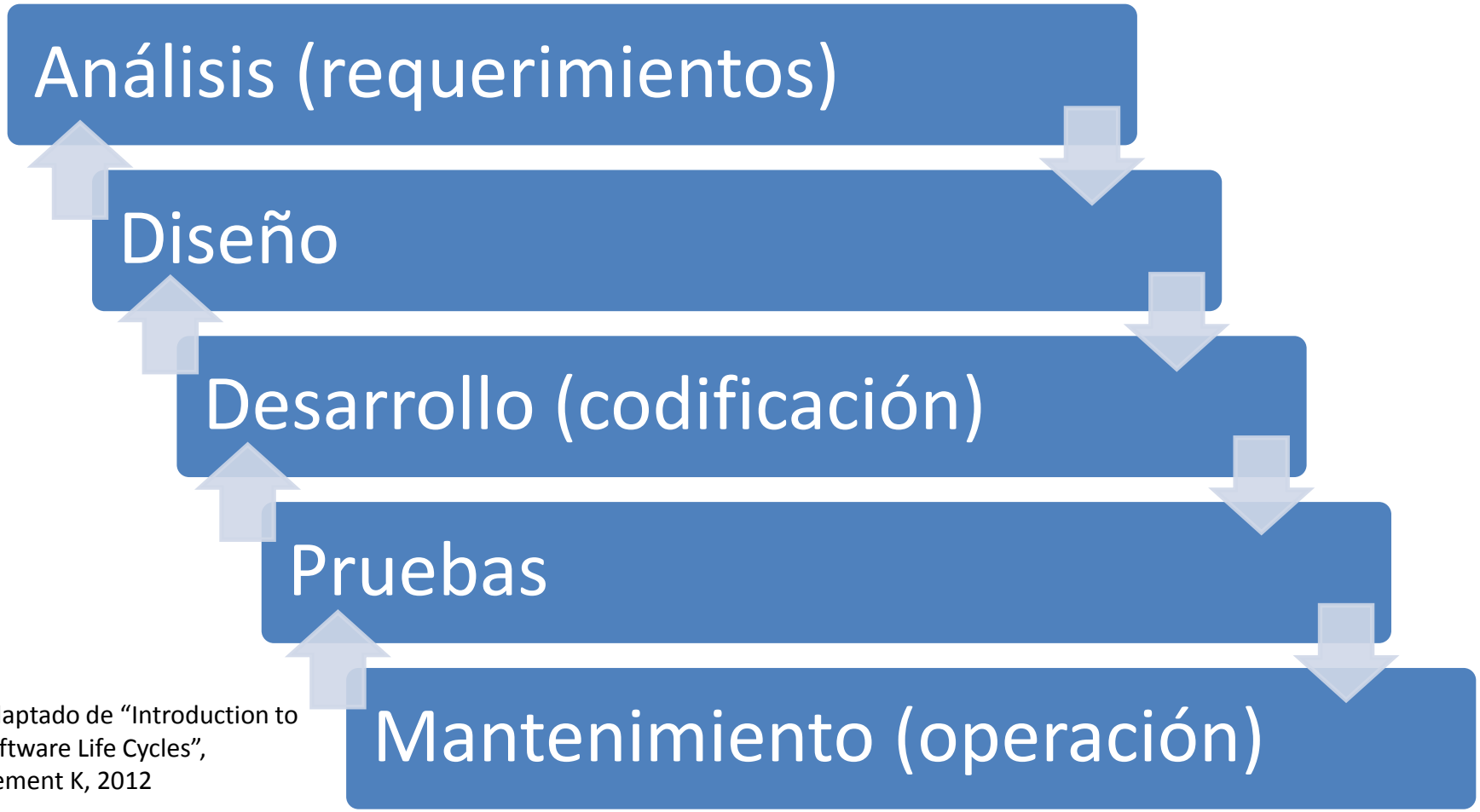
- Fácil de comprender y usar
- Fácil de planificar
- Fácil de estimar
- Sistema bien documentado
- Funciona con sistemas poco dinámicos

Desventajas

- Demanda tiempo
- No hay como retornar
- Carece de flexibilidad
  - Nuevos requerimientos
- Problemas se detectan tardíamente
- Costoso para proyectos pequeños

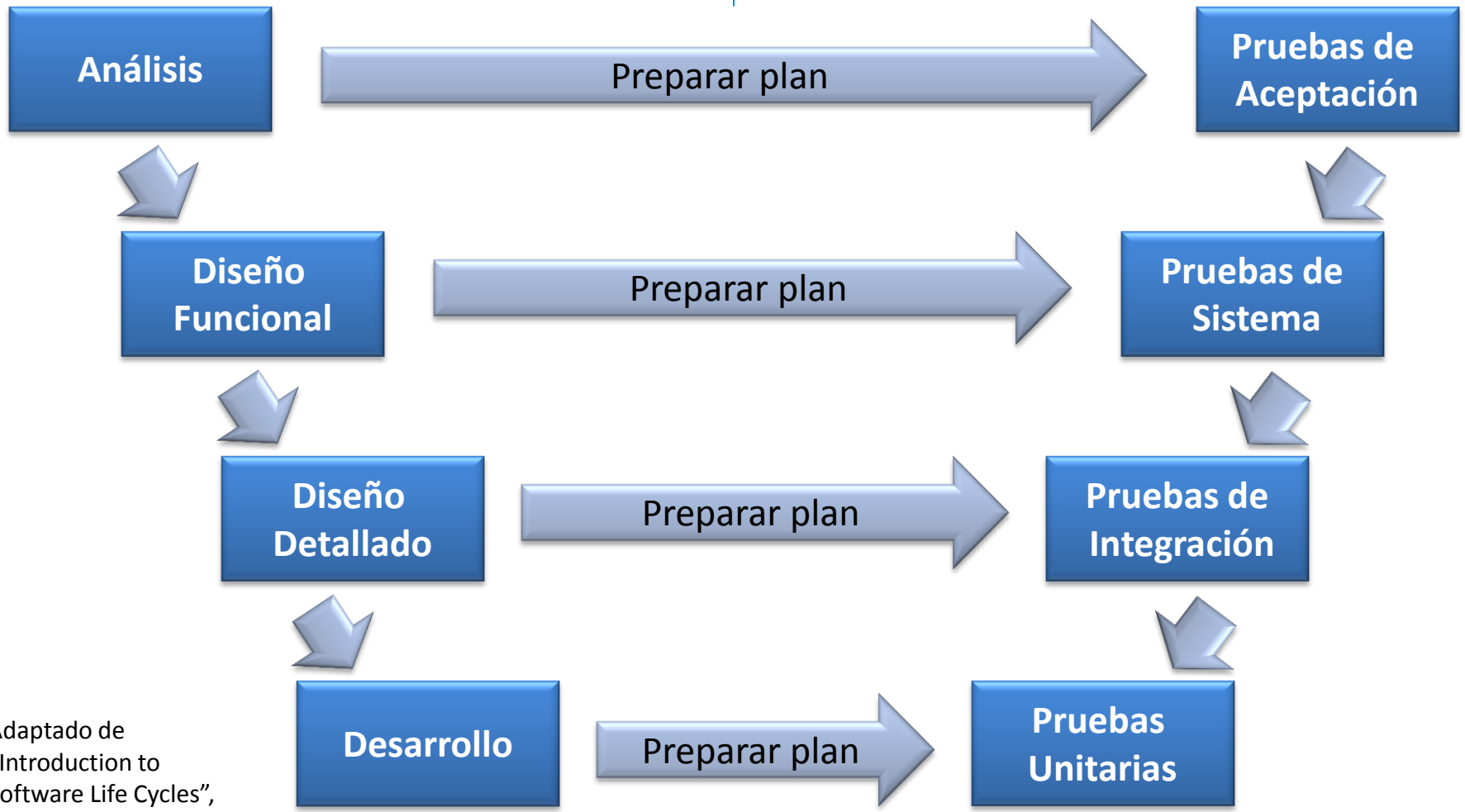
Modelo en Cascada

Variante



Adaptado de "Introduction to Software Life Cycles", Element K, 2012

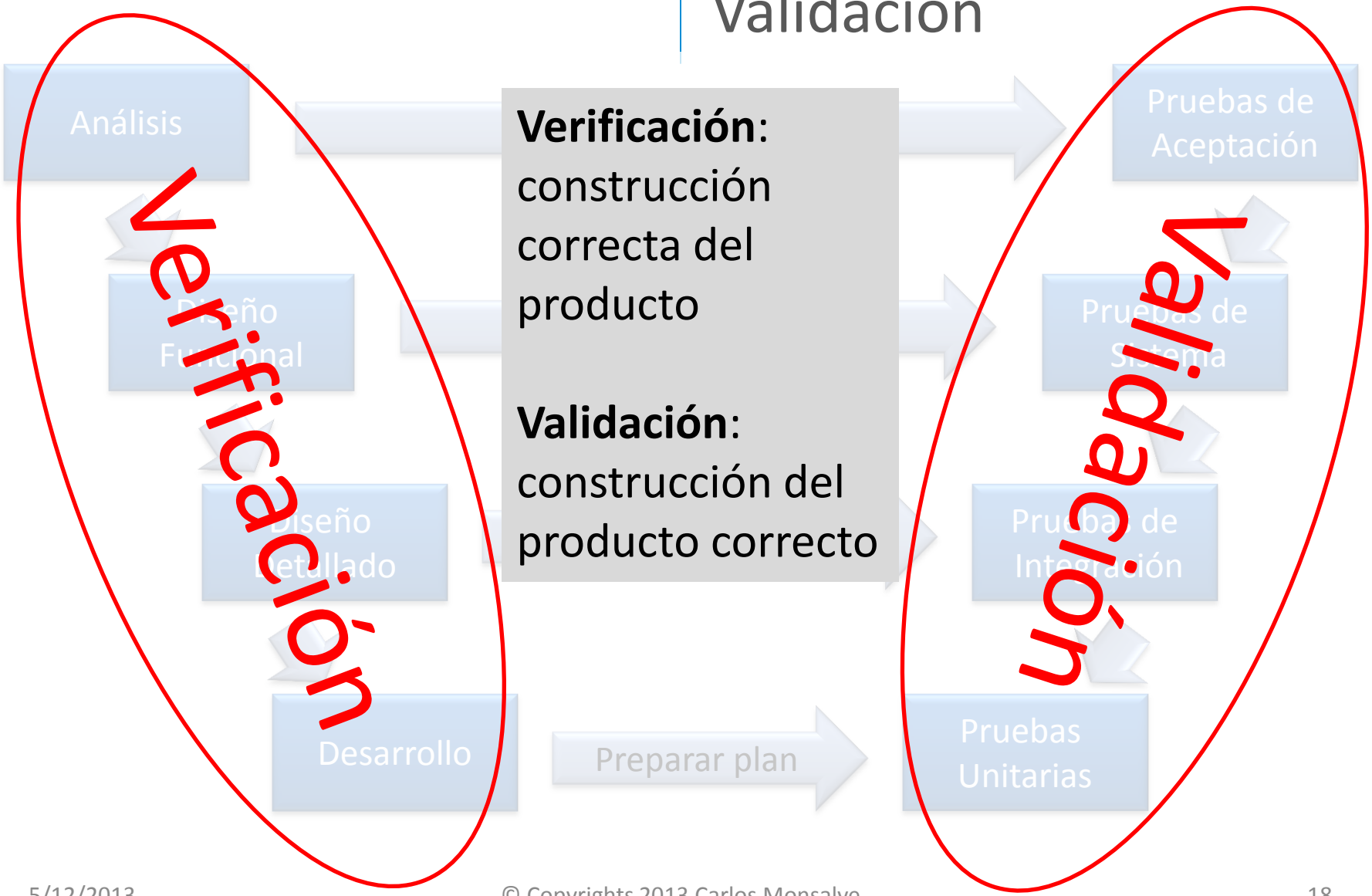
Modelo V | Etapas



Adaptado de "Introduction to Software Life Cycles", Element K, 2012

Modelo V

Verificación & Validación



Verificación: construcción correcta del producto

Validación: construcción del producto correcto

Modelo V | Evaluación

Ventajas

- Identifica problemas tempranamente
- Cierto paralelismo de actividades
- Poca complejidad
- Fácil de usar

Desventajas

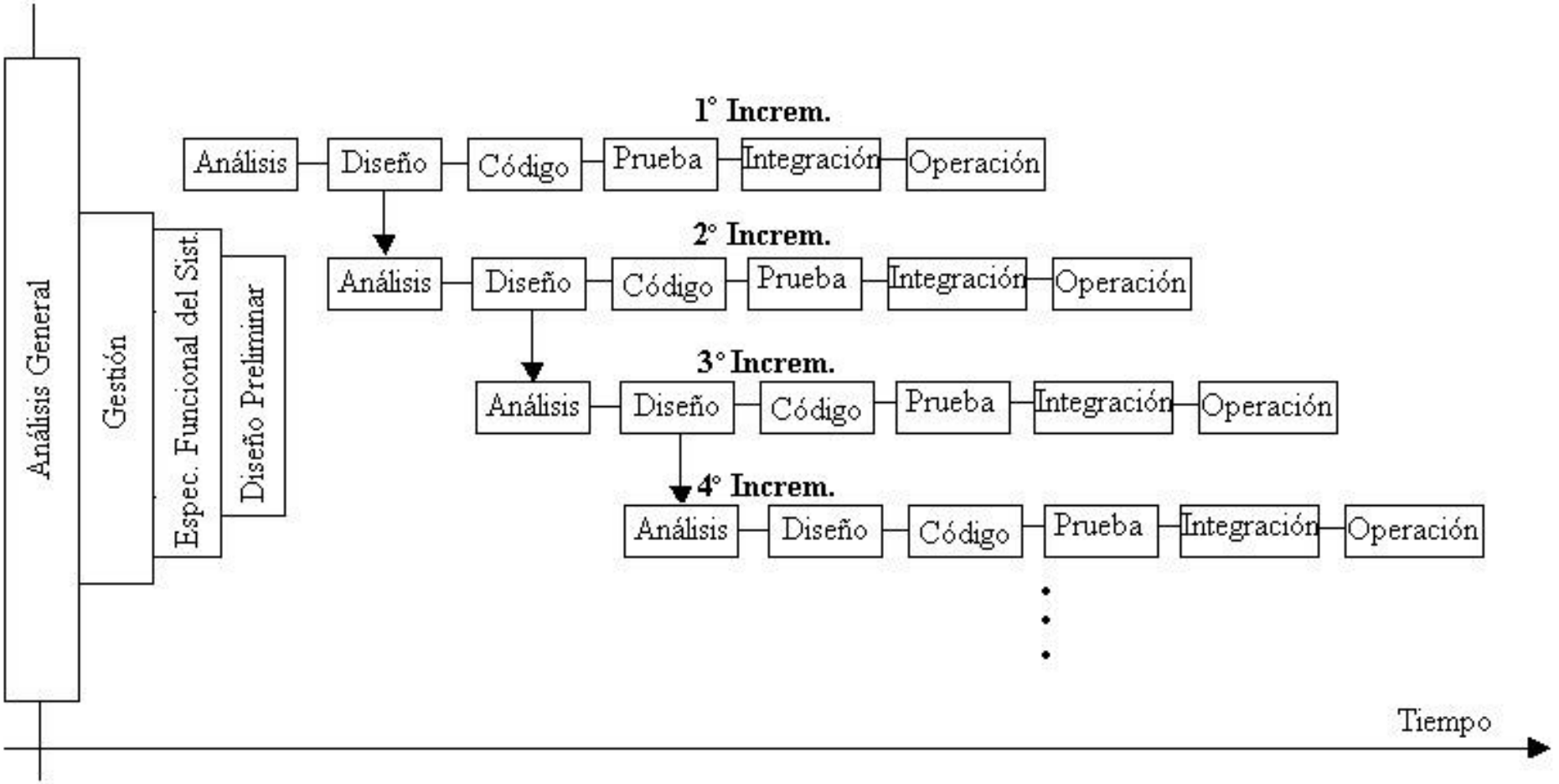
- Rara vez el cliente conoce exactamente lo que requiere desde el inicio
- Muy simple: falso sentimiento de victoria
- Producto es visible algo tarde en el proceso

Prototipos | Tipos

- Descartable: no funcional
- Evolutivo: no todos los requerimientos
- Incremental: por módulo o componente
- Extremo: para el web (html, scripts, servicios)

Modelo Incremental

¿Prototipo?



Modelo Incremental

Evaluación

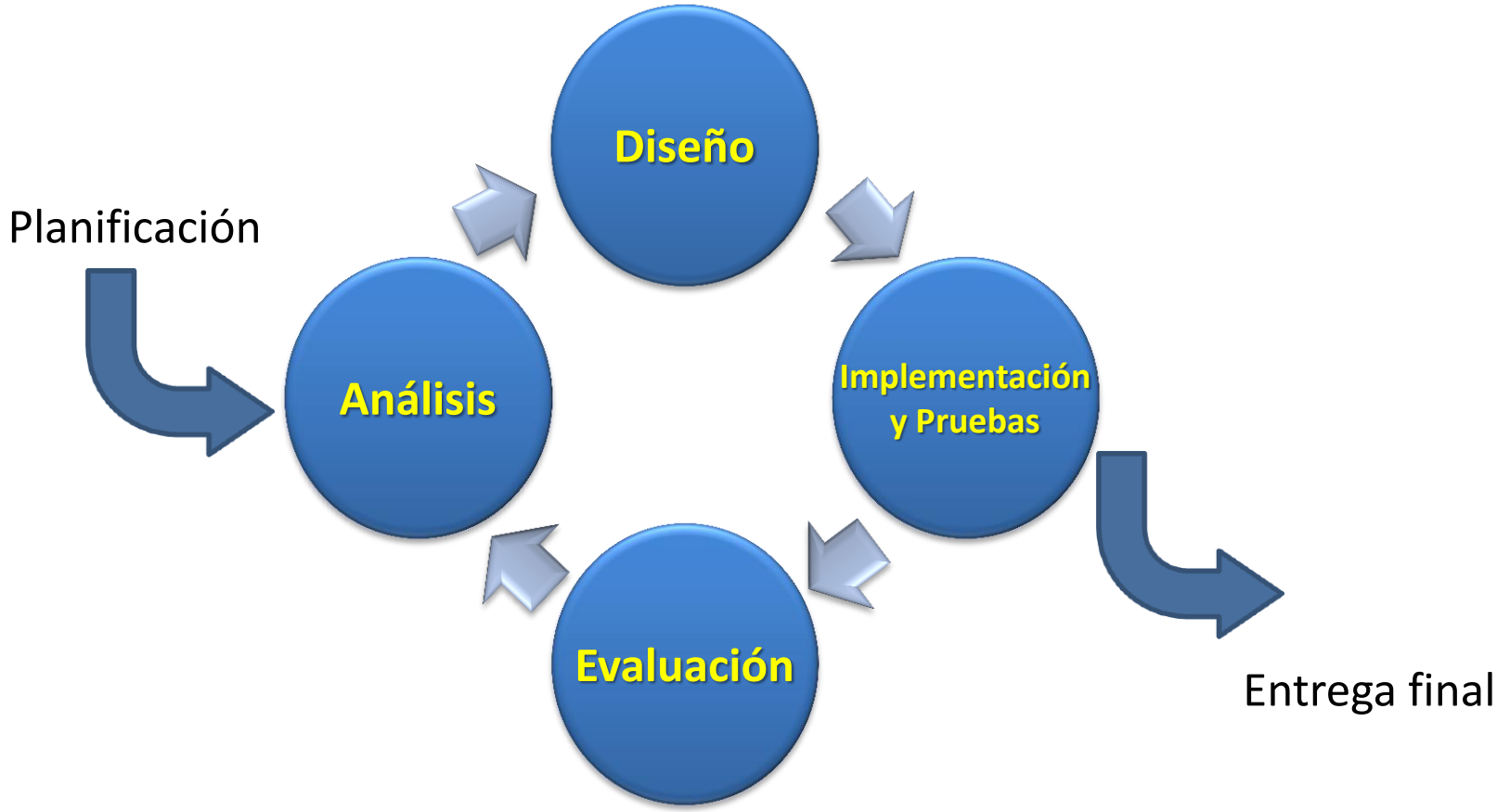
Ventajas

- Funcionalidad parcial
- Facilita un flujo de caja saludable
- Flexibilidad en definir componentes prioritarios
- No requiere especificar y diseñar en detalle todo el sistema

Desventajas

- Entregado un componente no admite nuevos requerimientos
- Retroalimentación de un componente poco ayuda a los otros
- No siempre es fácil dividir sistema en componentes

Modelo Iterativo | ¿Prototipo?



Modelo Iterativo | Etapa de Evaluación

- Evalúa versión actual de software
- Verifica cumplimiento de requerimientos
- Analiza mejoras a realizar
- Analiza elementos a añadir

Modelo Iterativo | Evaluación

Ventajas

- Usuario es parte continua en el proyecto
- Dinamismo: admite cambio de requerimientos
- Componentes son progresivamente mejorados

Desventajas

- ¿Cuándo parar en las mejoras?
- Puede salirse de tiempo y presupuesto
- Disponibilidad de usuario

Modelo Iterativo | Tipos

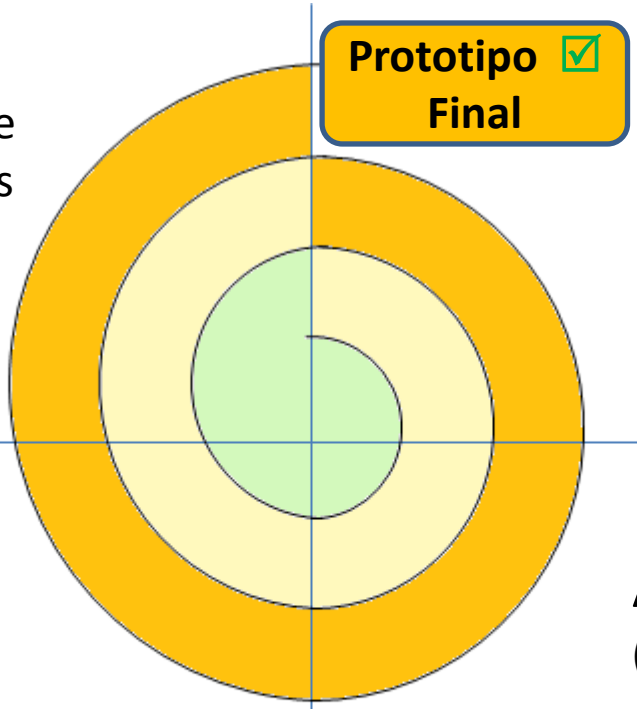
- Espiral
- Ágil
- Otros: RAD, RUP, etc.

Modelo Espiral

Etapas

Evaluación:

Verificación por el cliente
Identificación de defectos



Planificación:

Objetivos
Requerimientos
Limitaciones
Entregables

Análisis de Riesgos

(técnicos y administrativos):

Fortalezas
Debilidades
Riesgos
Alternativas

Ingeniería:

Desarrollo y
Pruebas

Adaptado de "Introduction to Software Life Cycles", Element K, 2012

Modelo Espiral | Evaluación

Ventajas

- Trabaja con prototipos
- Asegura calidad
- Riesgos son minimizados
- Cliente es parte del proceso

Desventajas

- Alto costo
- Demanda tiempo
- Requiere especialistas
- Iteraciones son largas

Modelo Ágil | Introducción

- Manifiesto (2001)
- Mejoramiento continuo:
  - Modelo iterativo (1 a 4 semanas)
  - Modelo incremental
- Colaboración intensiva del equipo
- Desarrollo guiado por pruebas
- Cliente es parte del proceso

Modelo Ágil | Principios

Prioritario

- Equipo de trabajo
- Software que funcione
- Colaboración con el cliente
- Flexibilidad y dinamismo

Menos prioritario

- Procesos y herramientas
- Documentación
- Negociación de contrato
- Planificación y normas

Modelo Ágil | Etapas

- Factibilidad
- Rentabilidad
- Riesgos
- Corto tiempo

- Financiamiento
- Requerimientos iniciales
- Equipo trabajo
- Diseño inicial
- Ecosistema

- Incremental
- Flexible
- Con el cliente
- Guiado por pruebas
- Estilos y convenciones
- Pruebas confirmatorias

- Pruebas de:
  - Sistema
  - Aceptación
- Documentación
- Entrenamiento

- Ambiente real
- Soporte
- Operación
- Monitoreo

- Fin del sistema

Modelo Ágil | Evaluación

Ventajas

- Flexible: acepta cambios
- Rápido a los cambios
- Entregables rápidos
- Sistema satisface al cliente
- No requiere pruebas largas al final

Desventajas

- Tiempo del cliente
- No siempre factible
- Difícil estimar (dinamismo)
- Pruebas de aceptación repetidas

Modelo Ágil | Variantes

- Programación extrema (XP)
- Scrum
- Desarrollo de sistemas dinámicos (DSDM)
- Crystal Clear
- Etc.

XP | Características

- Centrado en el código
- Simplicidad
- Pruebas continuas
- Retroalimentación continua
- Diseño iterativo
- Refinamientos continuos (o rehacerlo)

XP | Retroalimentación

Fuentes

- Cliente
- Revisores
- Personal de pruebas

Medios

- Iteraciones cortas
- Entregables rápidos
- Pruebas continuas

XP | Prácticas

- Planificación
- Entregables pequeños
- Diseño simple
- Pruebas unitarias
- Integración continua
- Refactorización
- Programación por pares
- Propiedad colectiva
- Semana de 40 horas
- Cliente en sitio
- Metáforas
- Estándares de codificación

XP | Planificación

- Definir objetivos y cómo lograrlos
- Dos fases:
  - Planificación de lanzamiento
    - Requerimientos
    - Tiempos
    - Costos
    - Funcionalidades
  - Planificación de iteración
    - Actividades
    - Asignaciones
    - Cronogramas
    - Cliente participa

XP | Refactorización

- Proceso de modificar código sin modificar funcionalidad
- Propósitos:
  - Optimización
  - Simplificación
  - Mejoras de mantenibilidad
- Luego de hacerlo: pruebas unitarias

XP | Programación x pares

- Producir el mismo código
- Trabajando juntos
- Una sola computadora
- Roles:
  - Lógica y codificación
  - Revisor: sugerencias continuas
- Roles se invierten
- Programadores pueden cambiar

XP | Propiedad colectiva

- Todo el equipo es responsable
- No hay responsables de módulos
- Cualquiera puede modificar cualquier parte del sistema

XP | Metáforas

- Enunciado sobre la visión del sistema
- Elaborado y compartido por todo el equipo
- Útil para otorgar nombres
  - Clases y métodos

XP | Evaluación

Ventajas

- Entregables rápidos
- Flexibilidad
- Refactorización
- Nadie es indispensable
- Bajos costos

Desventajas

- Incómodo para programadores
  - Dificultad de adaptarse
- Tiempo del cliente
- Dificultad de estimación y negociación

Ciclo de Vida | ¿Modelo único?

- Modelo debe responder a limitaciones de proyecto:
  - Tiempo
  - Costos
  - Recursos humanos disponibles
  - Cultura organizacional

- No existe un modelo perfecto para todo proyecto

Nuestro Modelo | Puntos claves

- Siempre pensar en el cliente
- A mayor proyecto, mayor planificación
- Permitir iteración entre las etapas
- Evaluación continua
- Incluir las mejores prácticas
- Hacerlo dos veces

Nuestro Modelo

Dinámico pero pensado

- Evitar cambios en medio de una iteración
- Evaluar cambios con métricas
- El recurso más crítico: el humano
  - Cambiarlo si es necesario
- Procurar consenso en el equipo