INGENIERÍA DE SOFTWARE I

Tema 3: Modelos de proceso

Grado en Ingeniería Informática
Fecha de última modificación: 6-2-2024

Dr. Francisco José García-Peñalvo / fgarcia@usal.es
Dra. Alicia García-Holgado / aliciagh@usal.es
Dra. Andrea Vázquez-Ingelmo

Departamento de Informática y Automática
Universidad de Salamanca

Resumen

Resumen	Se presentan diferentes modelos de proceso clasificados por categorías. Se parte del modelo clásico o en cascada y diferentes variantes del mismo. Posteriormente, se abordan modelos más evolucionados como pueden ser los modelos evolutivos en los que se considera la naturaleza cambiante del *software*, modelos específicos para sistemas orientados a objetos o modelos basados en reutilización centrados en el uso y desarrollo de componentes reutilizables. Asimismo, se abordan modelos más recientes tales como los procesos ágiles que enfatizan la programación frente al análisis, diseño y documentación, y modelos enfocados al desarrollo de sistemas web
Descriptores	Modelos de proceso; ciclo de vida; fases; modelos evolutivos; reutilización; orientación a objetos; procesos ágiles; ingeniería web
Bibliografía	[Piattini et al., 2004] Capítulo 3 [Pfleeger, 2002] Capítulo 2 [Pressman, 2010] Capítulos 2 y 3 [Sommerville, 2011] Capítulos 2 y 3

Esquema

- Clasificación de los modelos de proceso
- Modelos tradicionales
- Modelos evolutivos
- Modelos para sistemas orientados a objetos
- Modelos basados en reutilización
- Procesos ágiles
- Modelos para la Ingeniería Web
- Aportaciones principales del tema
- Ejercicios
- Lecturas complementarias
- Referencias

1. Clasificación de los modelos de proceso

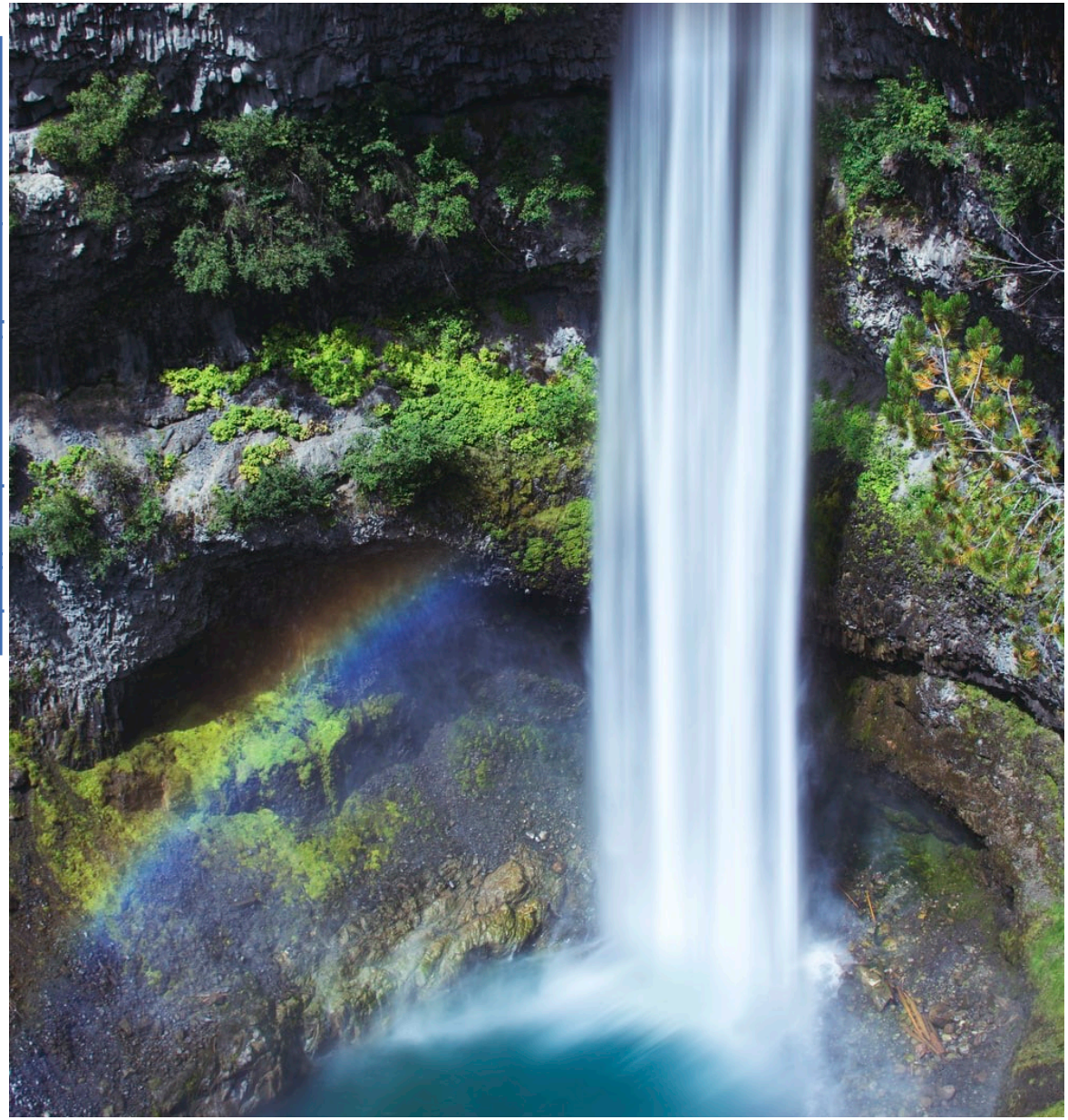
Clasificación de los modelos de proceso (i)

- Existen diferentes formas de clasificar los modelos de proceso en función de sus características y los tipos de sistemas a los que se aplican
  - **Modelos tradicionales**
    - Formados por un conjunto de fases o actividades en las que no tienen en cuenta la naturaleza evolutiva del *software*
  - **Modelos evolutivos**
    - Son modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo
  - **Modelos para sistemas orientados a objetos**
    - Modelos con un alto grado de iteratividad y solapamiento entre fases
  - **Modelos basados en reutilización**
    - Tienen en cuenta la reutilización sistemática del *software*
  - **Procesos ágiles**
    - Enfatizan el desarrollo rápido, ponen el énfasis en la programación
  - **Modelos para sistemas web**
    - Creados específicamente para el desarrollo de aplicaciones web

Clasificación de los modelos de proceso (ii)

- Ejemplos de modelos de proceso de diferentes categorías
  - **Modelos tradicionales**
    - Clásico, lineal o en cascada
    - Estructurado
    - Basado en prototipos
    - Desarrollo rápido de aplicaciones (RAD)
  - **Modelos evolutivos**
    - Incremental
    - Iterativo
    - En espiral
  - **Modelos para sistemas orientados a objetos**
    - De agrupamiento
    - Proceso Unificado
  - **Modelos basados en reutilización**
    - Basado en componentes
    - Proceso Unificado
  - **Procesos ágiles**
    - Programación extrema (XP)
    - Scrum
    - Desarrollo de *software* adaptativo
    - Crystal
  - **Modelos para sistemas web**
    - Modelos de Pressman
    - UML-based Web Engineering

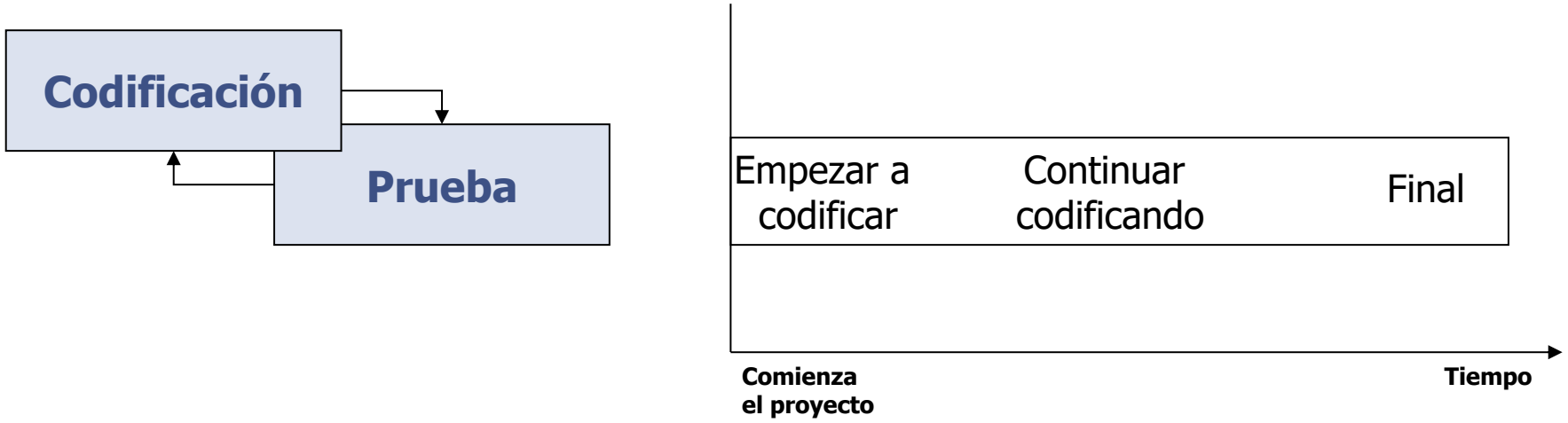
https://unsplash.com/photos/t1NiXOf5fTI



2. Modelos tradicionales

Modelo primitivo (i)

- Se le conoce también con el nombre de **Modelo Prueba y Error** o **Modelo Codifica y Mejora**
- Proceso de desarrollo aplicado en las primeras experiencias de programación
- Supone una iteración de fases codificación-depuración sin ninguna planificación ni diseños previos



Modelo primitivo (ii)

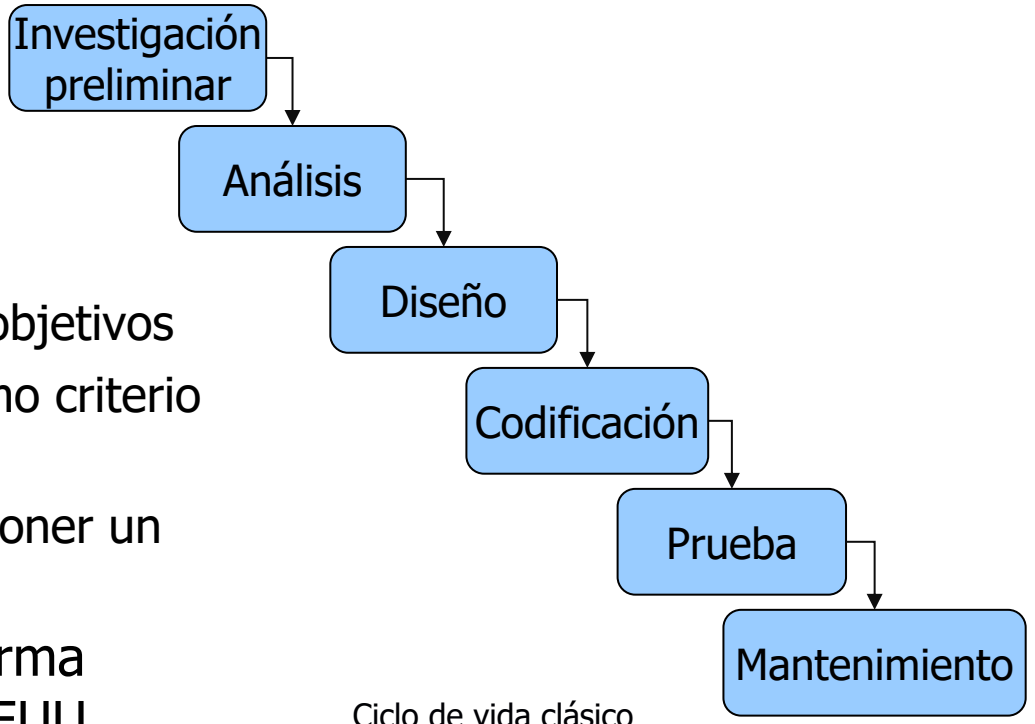
- Inconvenientes
  - Código pobremente estructurado tras varias iteraciones
    - Código espagueti
  - Caro de desarrollar por las numerosas recodificaciones
  - Posible rechazo del usuario al no existir análisis de requisitos
  - Caro de depurar por la falta de planificación
  - Caro de mantener por la falta de estructura y documentación

Modelos lineales o secuenciales (i)

- Han sido ampliamente utilizados
  - Ofrecen grandes facilidades a los gestores para controlar el progreso de los proyectos
  - Proponen un enfoque sistemático, secuencial, para el desarrollo del *software*
  - Comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento
  - Fases separadas en la especificación y el desarrollo
- La filosofía de estos modelos de proceso no es realista
  - No se ajusta al proceso de desarrollo *software*
  - Raramente sigue un flujo secuencial sino que exige diversas iteraciones
  - No ofrece un soporte adecuado a las técnicas de desarrollo basadas en objetos y componentes

Modelos lineales o secuenciales (ii)

- **Modelo clásico** (i)
  - Conocido también como modelo lineal o "en cascada"
  - Versión original se debe a W. Royce [Royce, 1970], pero aparecen después numerosos refinamientos
  - Características
    - Está compuesto por una serie de fases que se ejecutan secuencialmente
    - Paso de fase al conseguir los objetivos
    - Obtención de documentos como criterio de finalización de fase
    - El final de una fase puede suponer un punto de revisión
  - Se encuentra definido en la norma estándar 2167-A del DoD de EEUU



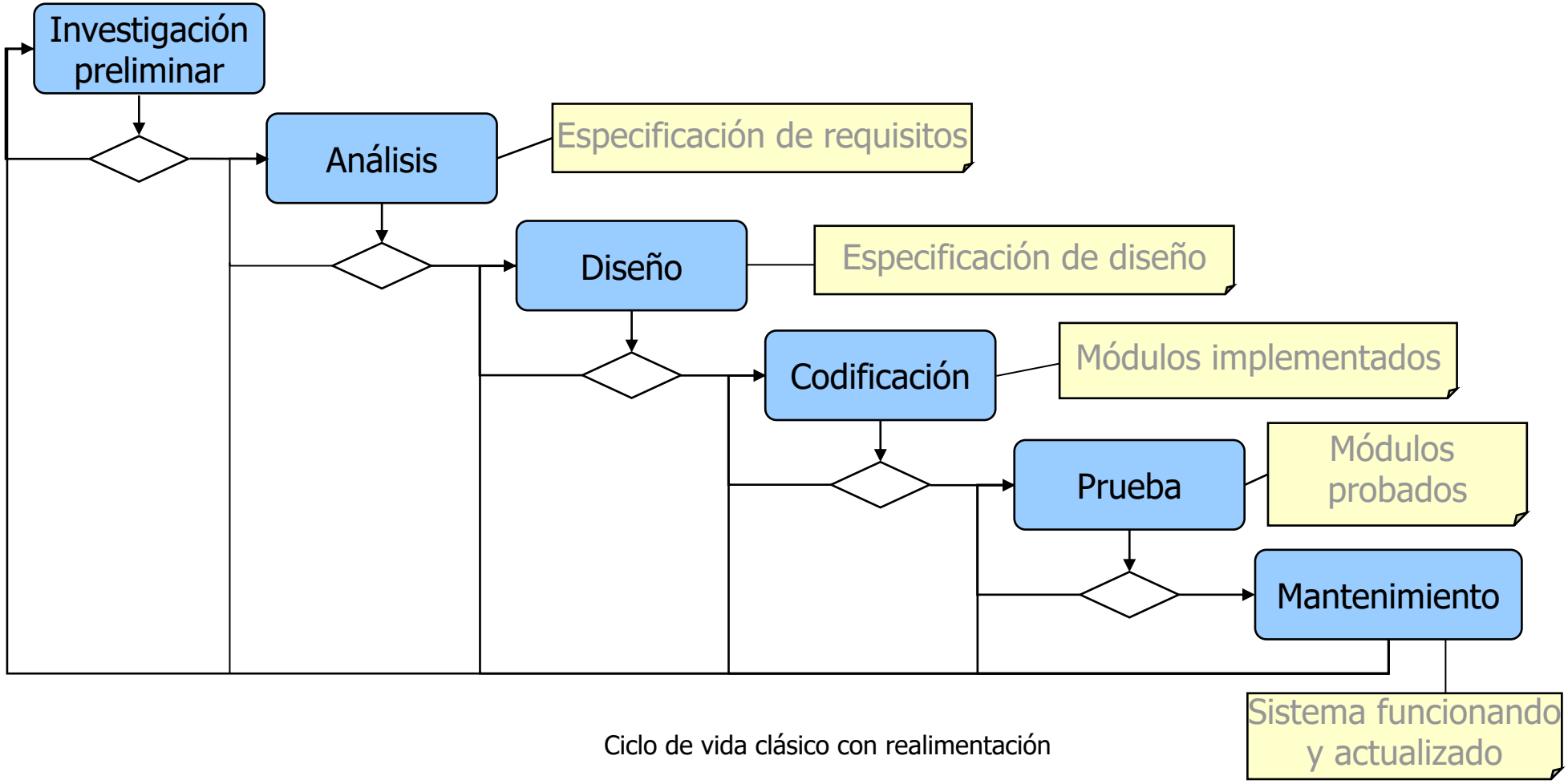
Ciclo de vida clásico

Modelos lineales o secuenciales (iii)

- **Modelo clásico** (ii)
  - Apoyo a los gestores
  - Distintas configuraciones
    - Muchos modelos más complejos son variaciones del modelo en cascada que incorporan lazos de realimentación y fases adicionales
  - Modelo satisfactorio solo en desarrollos conocidos y estables
  - El desconocimiento y el riesgo suele ser alto en el desarrollo del *software*
    - Desconocimiento de las necesidades por parte del cliente
    - Incomprensión de las necesidades por parte del proveedor
    - Inestabilidad de las necesidades
    - Opciones tecnológicas
    - Movimientos de personal
  - La linealidad no se corresponde con la realidad
    - Los retornos de información entre las fases se hacen necesarios para incorporar correcciones hacia arriba, en función de los descubrimientos realizados hacia abajo

Modelos lineales o secuenciales (iv)

- **Modelo clásico** (iii)



Ciclo de vida clásico con realimentación

Modelos lineales o secuenciales (v)

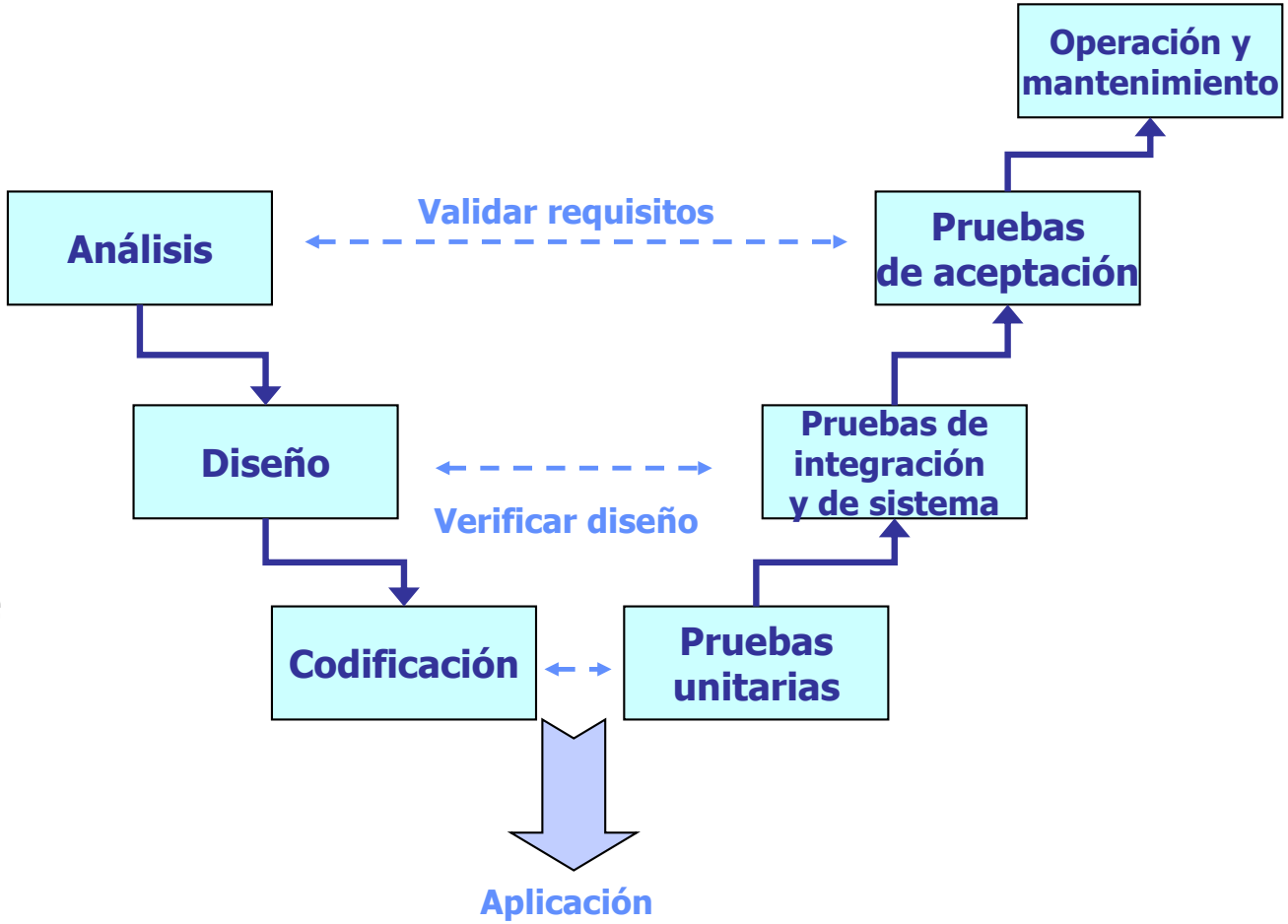
- **Modelo clásico** (iv)
  - Inconvenientes
    - Su progresión secuencial o lineal no refleja la manera en que realmente se desarrolla el *software* [Pfleeger, 2002; Pressman, 2006]
    - Es un modelo que adolece de rigidez [Pressman, 2010]
      - Exige al usuario que exponga explícitamente todos los requisitos al principio, presentando problemas para gestionar la incertidumbre natural propia del comienzo de la mayoría de los proyectos
    - Se tarda mucho tiempo en pasar por todo el ciclo [Piattini et al., 2004]
    - Es un modelo monolítico [Pressman, 2010]
      - Hasta llegar a las etapas finales del desarrollo no habrá una versión operativa del programa, lo que influye negativamente en el descubrimiento a tiempo de errores o incongruencias en los requisitos
    - Impone una estructura de gestión de proyecto al desarrollo del sistema [McCracken y Jackson, 1981]
    - No trata al *software* como un proceso de resolución de problemas [Curtis et al., 1987]

Modelos lineales o secuenciales (vi)

- **Modelo clásico** (v)
  - Consideraciones finales
    - Tiene un lugar destacado en la Ingeniería del *Software*
    - Proporciona una plantilla para adecuar los métodos
    - Es muy utilizado
    - Tiene problemas pero es mejor que desarrollar sin guías

Modelos lineales o secuenciales (vii)

- **Modelo V**
  - Es una variación del modelo en cascada que demuestra cómo se relacionan las actividades de prueba con las de análisis y desarrollo [GMD, 1992]
  - Presenta una implantación ascendente
  - Demuestra que el desarrollo de las pruebas se efectúa de manera síncrona con el desarrollo del programa
  - Mientras que el modelo clásico centra su atención en los documentos y artefactos producidos, el modelo en V lo hace en la actividad y la exactitud



Modelo V

Modelos basados en prototipos (i)

- Un prototipo es un modelo experimental de un sistema o de un componente de un sistema que tiene los suficientes elementos que permiten su uso
- **Objetivos**
  - Son un medio eficaz para aclarar los requisitos de los usuarios e identificar las características de un sistema que deben cambiarse o añadirse
  - Mediante el prototipo se puede verificar la viabilidad del diseño de un sistema
- **Características**
  - Es una aplicación que funciona
  - Su finalidad es probar varias suposiciones con respecto a las características requeridas por el sistema
  - Se crean con rapidez
  - Evolucionan a través de un proceso iterativo
  - Tienen un costo bajo de desarrollo

Modelos basados en prototipos (ii)

- **Enfoques de desarrollo**
  - **Desechable**: El prototipo es una versión rudimentaria del sistema que posteriormente es desechada
  - **Evolutivo**: El prototipo debe convertirse, eventualmente, en el sistema final usado (alternativa al ciclo de vida) [Basili y Turner, 1975]
  - **Mixto** (prototipado operativo) [Davis, 1992]
    - Se aplican técnicas convencionales para los requisitos bien conocidos
    - Combinación de prototipos desechables y evolutivos para los requisitos poco conocidos

	DESECHABLE	EVOLUTIVO
Enfoque de desarrollo	Rápido y sin rigor	Riguroso
Qué construir	Sólo las partes problemáticas	Primero las partes bien entendidas. Sobre una base sólida.
Directrices del diseño	Optimizar el tiempo de desarrollo	Optimizar la modificabilidad
Objetivo último	Desecharlo	Incluirlo en el sistema

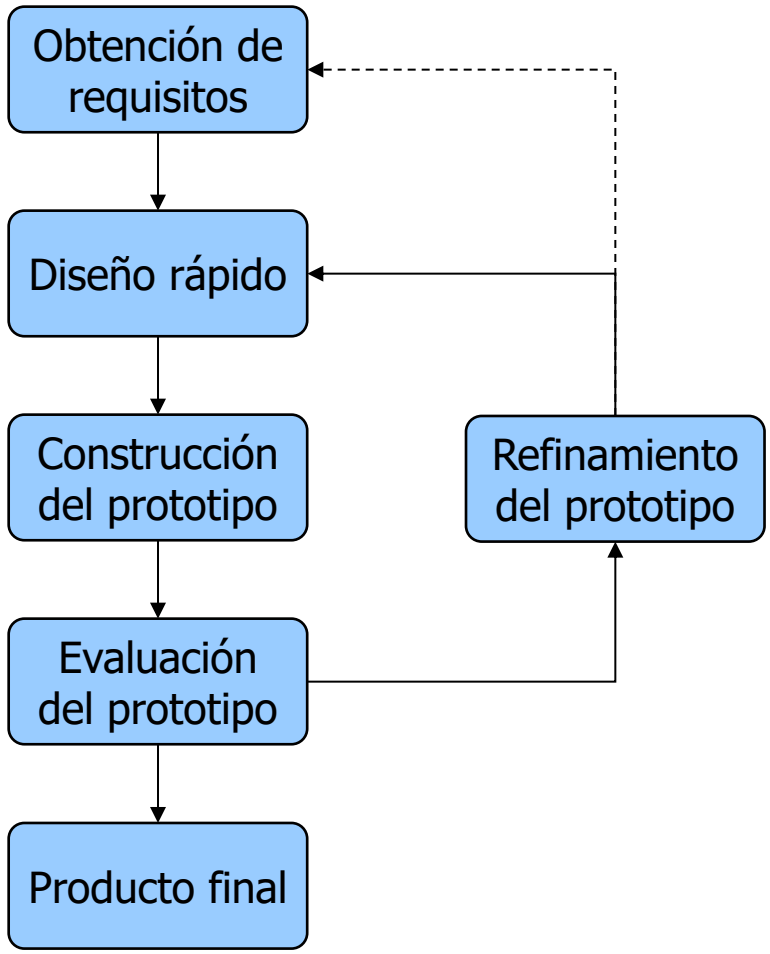
Diferencias entre los prototipos desechables y evolutivos

Modelos basados en prototipos (iii)

- **Prototipos desechables** (i)
  - Características
    - Se desarrolla código para explorar factores críticos para el éxito del sistema
    - La implementación usa lenguajes y/o métodos de desarrollo más rápidos que los definitivos
    - Se usa como herramienta auxiliar de la especificación de requisitos y el diseño
      - Determinar la viabilidad de los requisitos
      - Validar la funcionalidad del sistema
      - Encontrar requisitos ocultos
      - Determinar la viabilidad de la interfaz de usuario
      - Examinar alternativas de diseño
      - Validar una arquitectura de diseño particular
    - Este enfoque suele derivar en un modelo lineal una vez que el prototipo ha cumplido su misión

Modelos basados en prototipos (iv)

- **Prototipos desechables** (ii)
  - Características
    - Idea del *software* en líneas generales desde el punto de vista del usuario
    - Idealmente sirve para identificar los requisitos del *software*
      - Introduce cierta flexibilidad en la introducción de requisitos
    - Proceso iterativo
      - La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo a la vez que el desarrollador comprenda mejor lo que necesita hacer



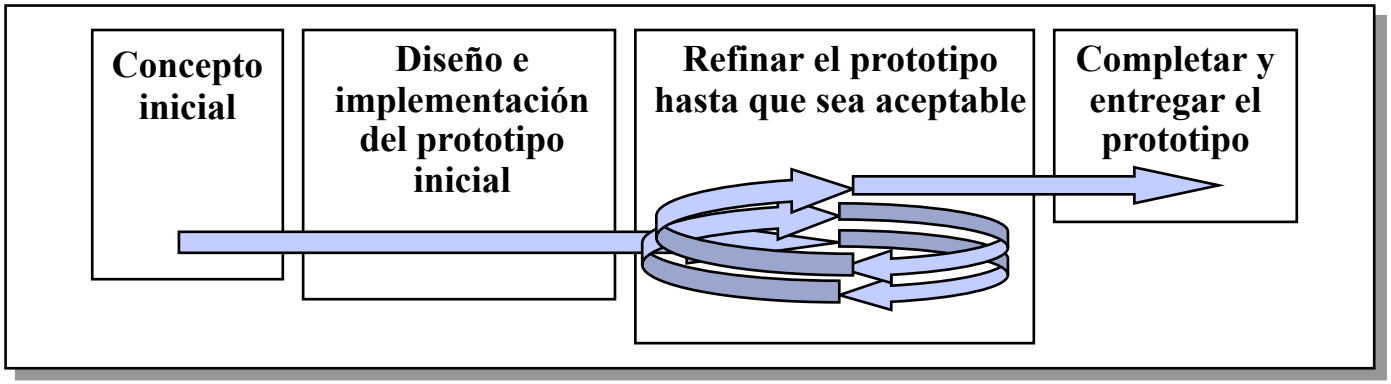
Modelo de prototipos

Modelos basados en prototipos (v)

- **Prototipos desechables** (iii)
  - Aplicaciones
    - Interfaz de usuario
    - Formatos de informes
    - Formatos de gráficos
    - Organización de bases de datos
    - Rendimiento de bases de datos
    - Precisión e implementación de cálculos complejos
    - Partes con respuesta crítica en el tiempo en sistemas de tiempo real
    - Rendimiento de sistemas interactivos
    - Viabilidad de partes del sistema en las que no se tiene experiencia

Modelos basados en prototipos (vi)

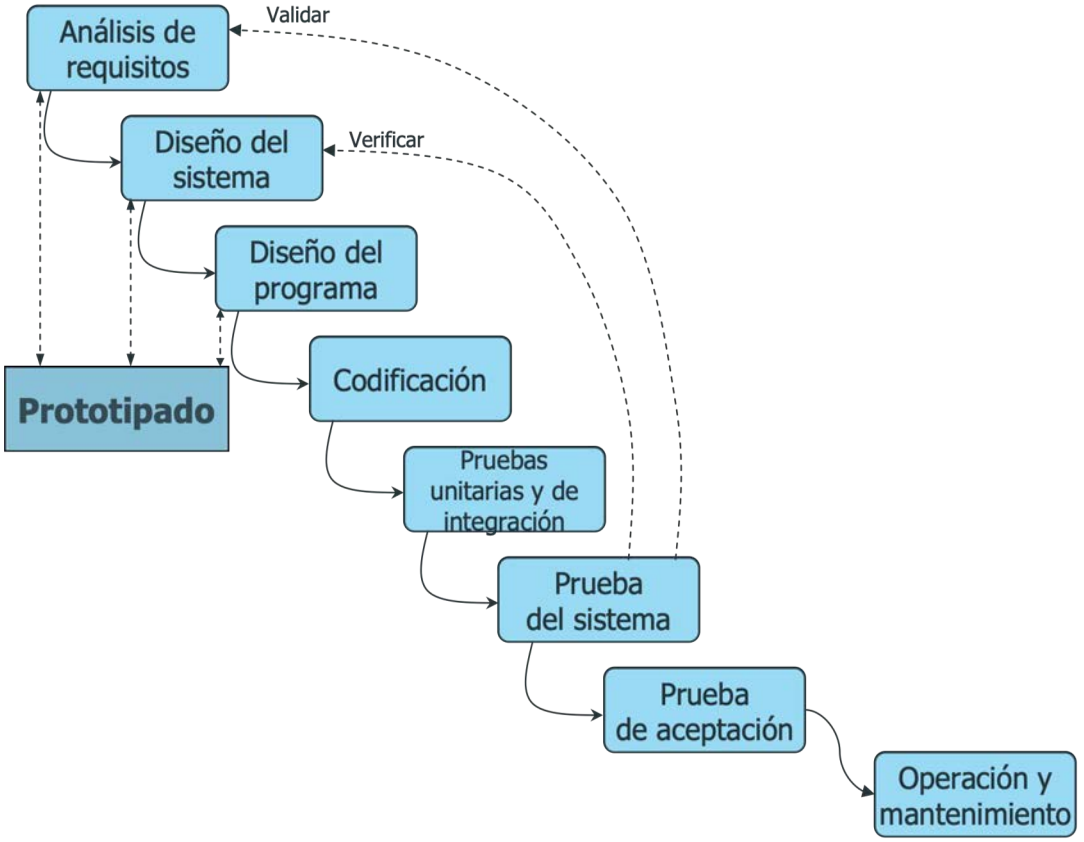
- **Prototipado evolutivo (ciclo de vida iterativo)**
  - Características
    - Enfoque de desarrollo que se utiliza cuando no se conoce con seguridad lo que se quiere construir
    - Se comienza diseñando e implementando las partes más destacadas del sistema
    - La evaluación del prototipo proporciona la realimentación necesaria para aumentar y refinar el prototipo
    - El prototipo evoluciona y se transforma en el sistema final



Modelo de prototipado evolutivo

Modelos basados en prototipos (vii)

- Ventajas del prototipado
  - Permite solventar objeciones del usuario
  - Sirve para formalizar la aceptación previa
  - Introduce flexibilidad en la captura de requisitos
  - Es útil cuando el área de aplicación no está definida, cuando el riesgo de rechazo el alto, o como forma de evaluar el impacto de una aplicación
  - El prototipado es un subproceso que puede incluirse como parte de otros modelos de proceso
    - Por ejemplo puede combinarse con un ciclo en cascada para intentar solventar ciertas carencias de este



Modelo en cascada con prototipado [Pfleeger, 2002]

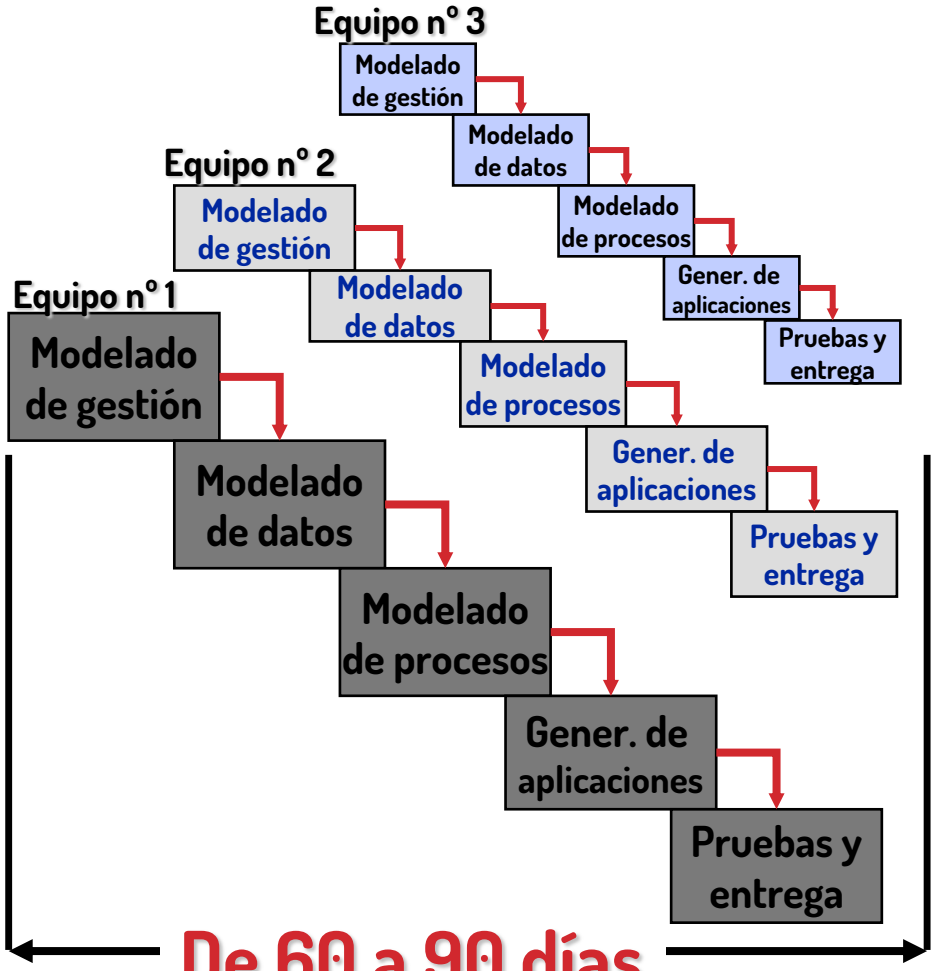
Modelo basados en prototipos (viii)

- Inconvenientes
  - El sistema se puede llegar a deteriorar tendiendo hacia el modelo primitivo
  - Se suele refinar el prototipo hacia el sistema final en lugar de desecharlo y empezar desde el principio
  - El cliente puede encontrar atractivo el prototipo y quedarse con el prototipo como sistema final
  - Relajación de los desarrolladores
  - No disminuye el tiempo entre la definición de los requisitos y la entrega del producto
  - Al usuario le desagrada que se deseche código

Desarrollo rápido de aplicaciones (i)

- El modelo de desarrollo rápido de aplicaciones, DRA (RAD – *Rapid Application Development*) o modelo de la caja de tiempo surgió como respuesta al modelo formal y al ciclo en espiral
- Enfatiza un ciclo de desarrollo extremadamente corto
  - Modelo funcional en 60 o 90 días
- No es un modelo bien definido
  - Secuencia de integraciones de un sistema evolutivo o de prototipos que se revisan con el cliente ➔ descubrimiento de los requisitos
  - Cada integración se restringe a un período de tiempo bien definido (caja de tiempo)
- Características
  - Modelo secuencial: Separación en fases de cada *caja de tiempo*
  - Integraciones constantes
  - Centrado en el código más que en la documentación
  - Desarrollo basado en componentes
  - Uso efectivo de herramientas y *frameworks*
  - Participación activa del usuario

Desarrollo rápido de aplicaciones (ii)



Modelo DRA [Kerr y Hunter, 1994]

- Cuando se utiliza en S.I. Automatizados, comprende las fases [Kerr y Hunter, 1994]
  - Modelado de gestión
  - Modelado de datos
  - Modelado del proceso
  - Generación de aplicaciones
  - Pruebas y entrega

Desarrollo rápido de aplicaciones (iii)

- Las limitaciones de tiempo demandan un ámbito de escalas
- Si una aplicación de gestión puede modularse de forma que pueda completarse cada una de las funciones principales en menos de tres meses, es un candidato del DRA. Cada una de estas funciones puede ser afrontadas por un equipo DRA diferente y ser integradas en una sola aplicación
- Inconvenientes [Butler, 1994]
  - Los proyectos grandes necesitan los recursos humanos suficientes para crear el número correcto de equipos
  - Se requiere de un compromiso de las partes involucradas

https://unsplash.com/photos/aWDggexSxA0

3. Modelos evolutivos

Modelos evolutivos (i)

- El *software*, al igual que todos los sistemas complejos, evolucionan con el tiempo [Gilb, 1988]
- Se caracterizan porque permiten desarrollar versiones cada vez más completas del *software*, teniendo en cuenta la naturaleza evolutiva del *software*
- Presentan la filosofía de poner un producto en explotación cuanto antes
  - Están muy ligados a la idea de prototipado evolutivo
- Existen muchos modelos de proceso evolutivos
- Los modelos evolutivos son iterativos [Pressman, 2010]
  - Se caracterizan por la forma en que permiten a los ingenieros de *software* desarrollar versiones cada vez más completas del producto *software*, que puede ir entregándose al cliente en forma de incrementos

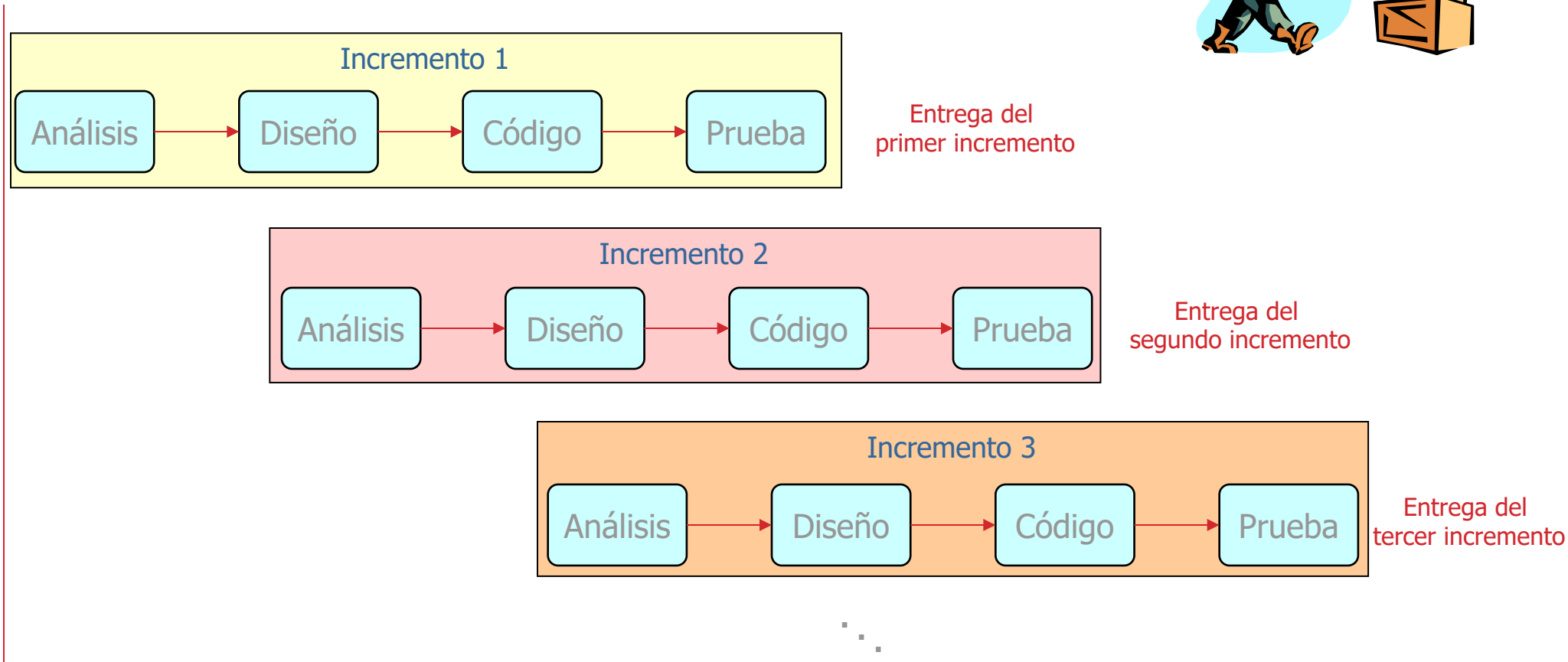
Modelos evolutivos (ii)

- Los desarrollos orientados a objetos se ajustan a un modelo de proceso iterativo e incremental
- Se puede argumentar lo mismo para los desarrollos basados en componentes
- Esto es así porque
  - Las tareas de cada fase se llevan a cabo de una forma iterativa
  - A la vez que existe un ciclo de desarrollo **análisis-diseño-implementación-análisis** que permite hacer evolucionar al sistema
  - En el desarrollo incremental el sistema se divide en un conjunto de particiones
    - Cada una se desarrolla de forma completa hasta que se finaliza el sistema
- Esta idea de iteratividad máxima propia de la orientación a objetos ha sido equiparada por autores como James Rumbaugh (1992) o L. B. S. Raccoon (1995) a las fractales o la teoría del caos

Modelo incremental (i)

- En este modelo, el sistema, tal y como está especificado en la especificación de requisitos del *software*, se divide en subsistemas de acuerdo a su funcionalidad
- Las versiones se definen comenzando con un subsistema funcional pequeño y agregando funcionalidad con cada nueva versión
  - Cada nueva parte entregada se denomina incremento
- Combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos
- Aplica secuencias lineales de forma escalonada mientras progresa el calendario del proyecto
  - Cada secuencia lineal supone un incremento [McDermid y Rook, 1993]

Modelo incremental (ii)

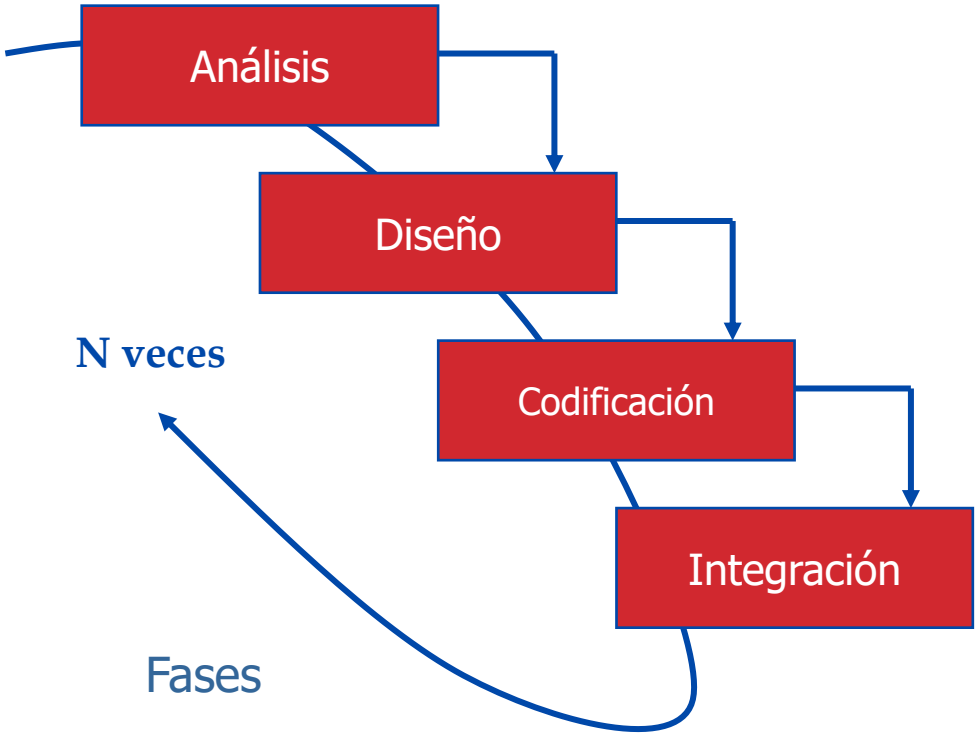


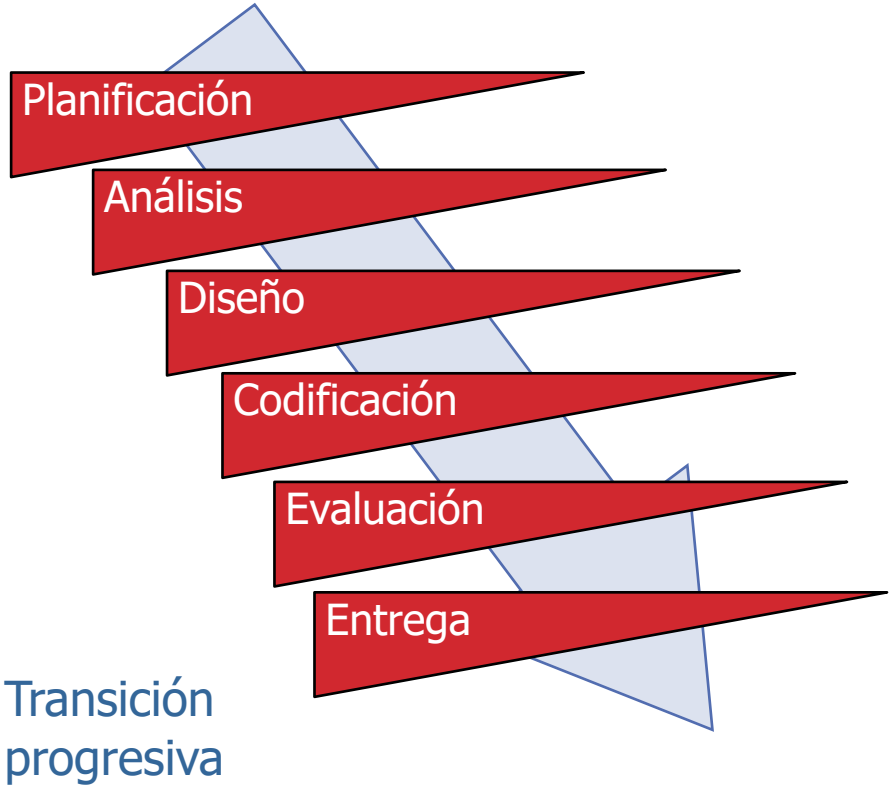
Modelo incremental [Pressman, 2010]

Modelo iterativo (i)

- Entrega un sistema completo desde el principio, para posteriormente cambiar la funcionalidad de cada subsistema con cada versión
- Características del ciclo iterativo [Muller, 1997]
  - Se basa en la evolución de prototipos ejecutables, mensurables y evaluables
  - Se van incorporando cambios en cada iteración
  - Exige más atención e implicación de todos los actores del proyecto
- Minicascada
  - Cada iteración reproduce el ciclo de vida en cascada, pero a una escala menor
  - Los objetivos de cada iteración se establecen en función de la evaluación de las iteraciones precedentes
  - Las fases tradicionales se cubre gradualmente en las diversas iteraciones
  - Las actividades internas se solapan porque dentro de una iteración no necesitan terminarse de golpe, siendo la transición entre dos actividades progresiva

Modelo iterativo (ii)



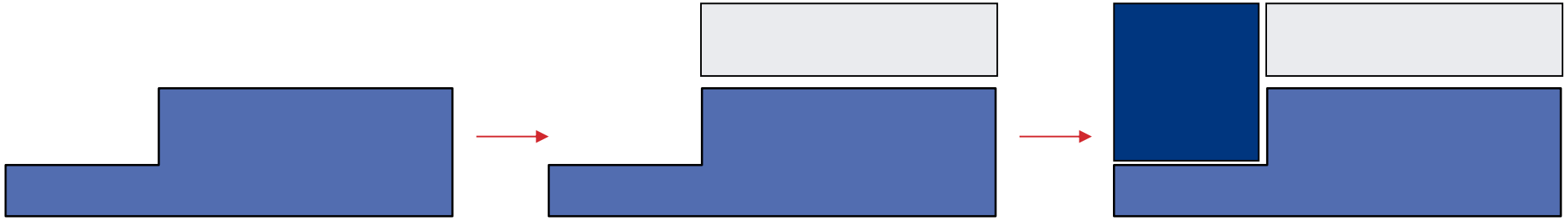


Modelo iterativo (iii)

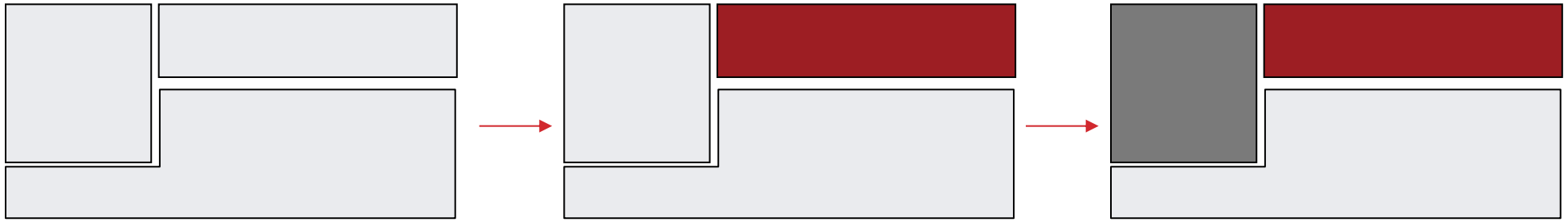
- Evaluación de las iteraciones
  - Deben definirse criterios de evaluación de las iteraciones
  - Una iteración se marca por etapas intermedias que permitan medir los progresos. Debe haber al menos dos etapas
    - Revisión inicial: fija los objetivos y criterios de la iteración
    - Revisión de evaluación: valida los resultados
- Mitos sobre el ciclo de vida iterativo [Muller, 1997]
  - El ciclo de vida iterativo favorece los apaños
  - El ciclo de vida iterativo engendra problemas
  - El ciclo de vida iterativo e incremental exige recomenzar *n* veces hasta que el resultado sea el adecuado
  - El ciclo de vida iterativo es una excusa para no planificar y gestionar un proyecto
  - El ciclo de vida iterativo sólo concierne a los desarrolladores
  - El ciclo de vida iterativo favorece siempre añadir nuevas necesidades, sin fin

Incremental vs. iterativo

Desarrollo incremental: sistema parcial, funcionalidad completa



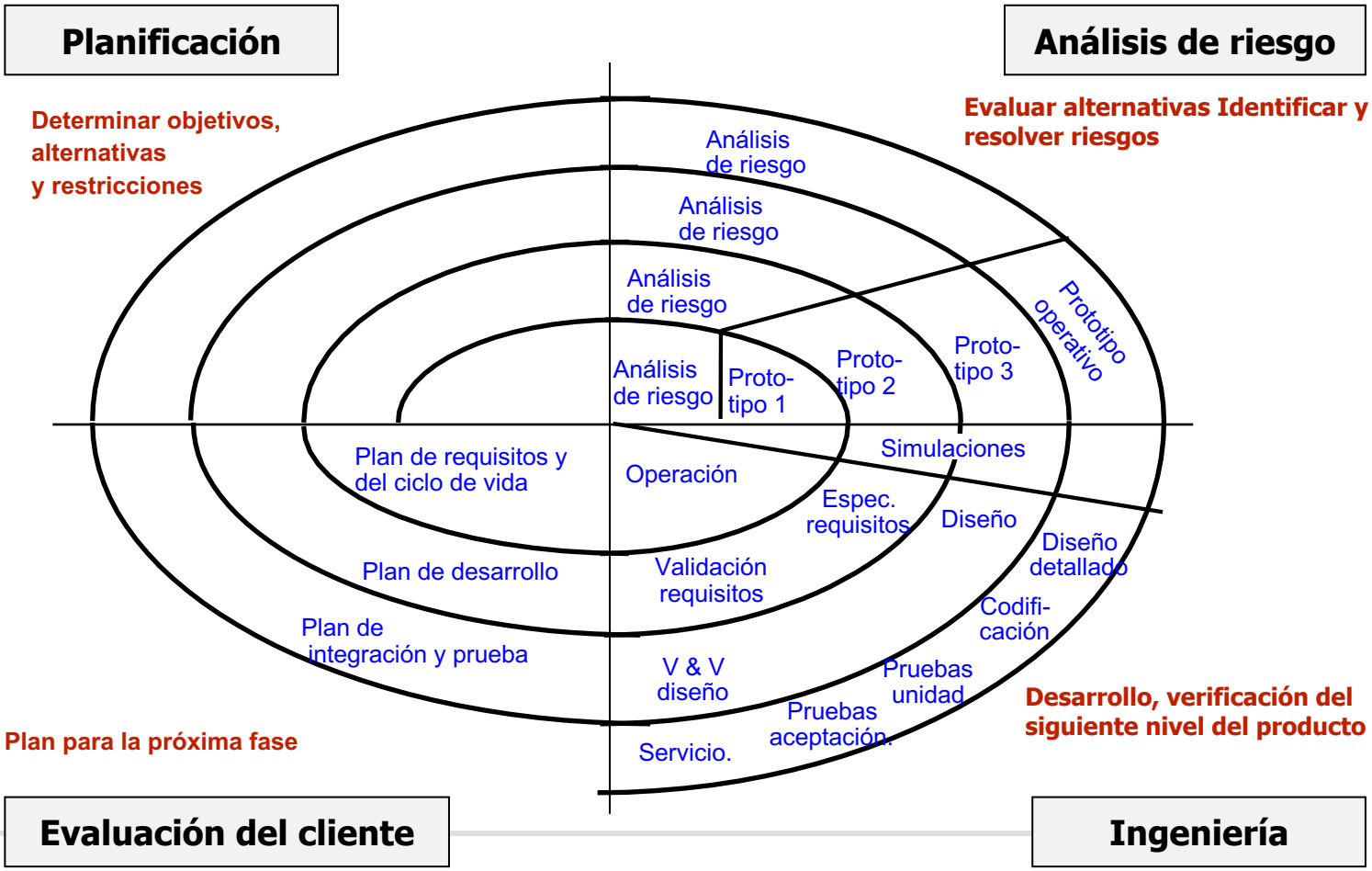
Desarrollo iterativo: sistema completo; funcionalidad parcial



Modelos en espiral (i)

- **Modelo en espiral**
  - Fue propuesto inicialmente por **B. Boehm** [Boehm, 1986, 1988]
  - Es un modelo de proceso de *software* evolutivo, que proporciona el potencial para el desarrollo rápido de versiones incrementales del *software*
  - Características
    - Puede considerarse como un metamodelo de proceso
      - Reúne características del modelo clásico y de prototipos
    - Aparece el análisis de riesgo
    - Se divide en un número de actividades estructurales, también denominadas regiones de tareas. En el modelo original de **Boehm** aparecen cuatro regiones de tareas
      - Planificación, Análisis de riesgos, Ingeniería, Evaluación del cliente
    - El avance se realiza desde el centro de la espiral hacia el exterior

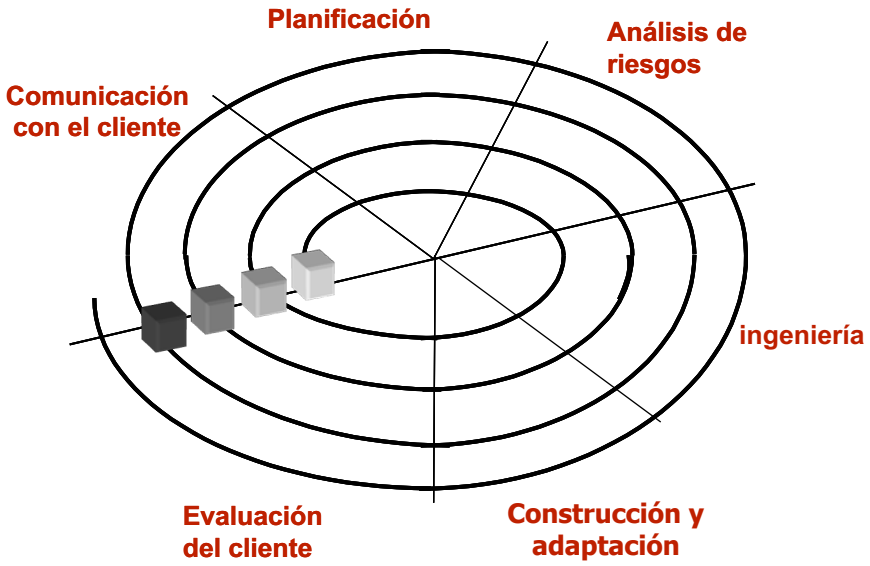
Modelos en espiral (ii)



Ciclo de vida en espiral [Boehm, 1988]

Modelos en espiral (iii)

- **Modelo en espiral de Pressman** [Pressman, 2002]
  - Variante del modelo de Boehm con 6 regiones de tareas
    - Se define un eje con diferentes puntos de entrada para diferentes tipos de proyectos



Puntos de entrada al proyecto

- Proyecto de mantenimiento de productos
- Proyecto de mejora de productos
- Proyecto de desarrollo de productos nuevos
- Proyecto de desarrollo de conceptos

Modelo en espiral de Pressman

Modelos en espiral (iv)

- **Modelo *win-win*** [Boehm et al., 1998]
  - Extiende el modelo en espiral haciendo énfasis en las condiciones de éxito (ganancia) de todas las partes involucradas en el proyecto
  - Consta de cuatro **ciclos**
    - **Ciclo 0. Grupos de aplicación**: determinación de la viabilidad de un grupo de aplicaciones
    - **Ciclo 1. Objetivos del ciclo de vida de la aplicación**: objetivos, prototipos, planes, especificaciones de cada aplicación y arquitectura viable
    - **Ciclo 2. Arquitectura del ciclo de vida de la aplicación**: establecimiento de una arquitectura detallada y verificación de su viabilidad
    - **Ciclo 3. Capacidad de operación inicial**: consecución de la capacidad para cada etapa crítica del proyecto

Modelos en espiral (v)

- **Modelo *win-win*** [Boehm et al., 1998]

Modelos en espiral (vi)

- Ventajas
  - Refleja de forma más realista la idiosincrasia del desarrollo de *software*
  - Toma lo mejor y evita lo peor de los demás modelos, según la situación en cada momento
  - Las opciones de reutilización se tienen en cuenta desde el primer momento
  - Proporciona una preparación para la evolución, crecimiento y cambio
  - Proporciona un mecanismo para incorporar objetivos de calidad en el desarrollo
  - Se centra en la eliminación de errores y opciones no atractivas desde el principio
  - Determina el nivel de esfuerzo de cada fase en cada proyecto
  - Se sigue el mismo procedimiento para el desarrollo que para el mantenimiento, con lo que se evitan los problemas de las "mejoras rutinarias" de alto riesgo
  - Permite una gran flexibilidad
  - Se adapta bien al diseño y programación orientado a objetos

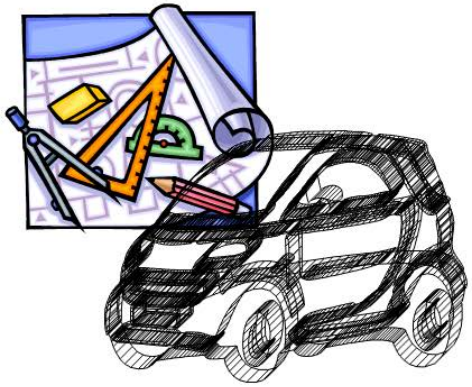
Modelos en espiral (vii)

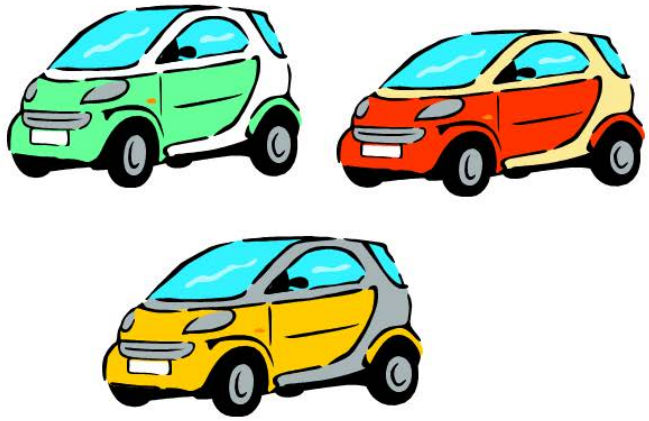
- Inconvenientes
  - No ha sido desarrollado en el mundo de la contratación comercial sino en el de desarrollo interno
    - Puede resultar difícil convencer a grandes clientes de que el enfoque evolutivo es controlable
  - Necesita experiencia en la evaluación de riesgos, expertos, que no siempre están disponibles
  - Necesita una elaboración adicional de los pasos del modelo

Modelos en espiral (viii)

- Diferencias con otros modelos [Wolff, 1989]
  - Existe un reconocimiento explícito de las diferentes alternativas para alcanzar los objetivos del proyecto
  - El modelo se centra en identificar los riesgos de cada alternativa, así como las formas de solventarlos
  - La división de los proyectos en ciclos, cada uno con un acuerdo al final, implica que existe un acuerdo para los cambios a realizar o para la finalización del mismo, en función de lo aprendido a lo largo del proyecto
  - Es un método que se adapta a cualquier tipo de actividad, alguna de las cuales no existen en otros paradigmas, como puede ser la consulta a asesores externos

4. Modelos para sistemas orientados a objetos





Modelos para sistemas orientados a objetos

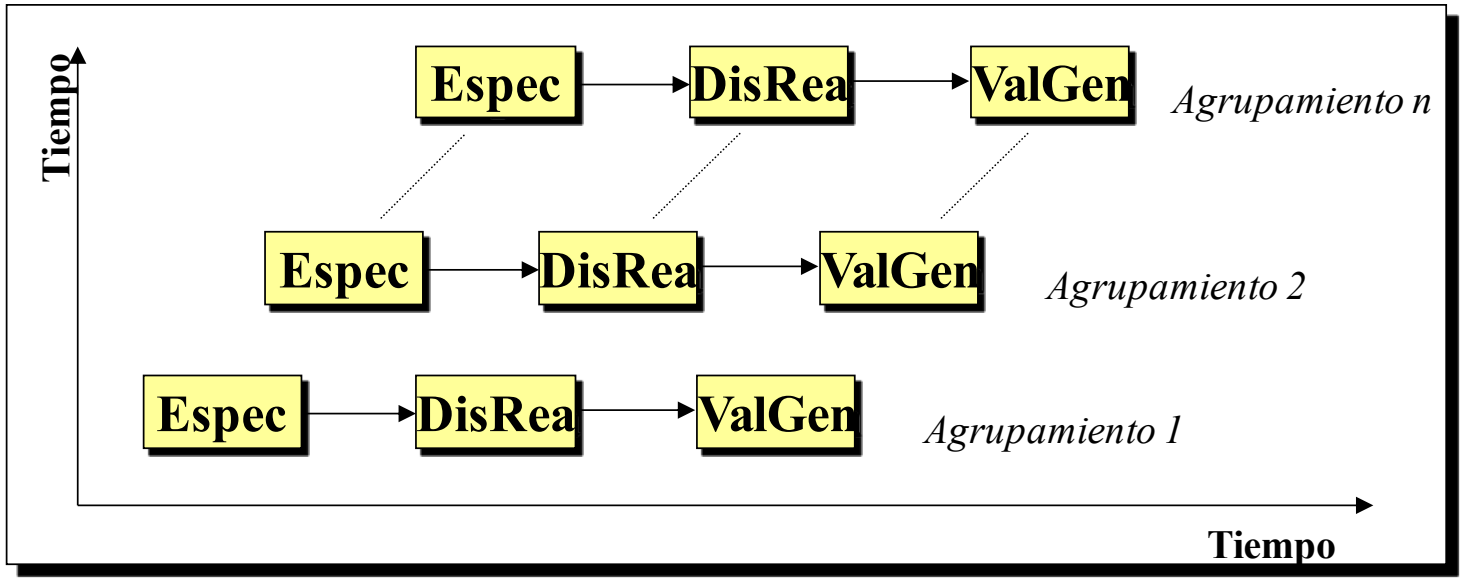
- Surgieron por la dificultad de adaptar los modelos tradicionales al desarrollo de sistemas orientados a objetos
- Características principales
  - Eliminación de las fronteras entre fases
    - En el desarrollo de *software* orientado a objetos están más difuminados los límites entre fases
  - Uso de componentes reutilizables
    - Las características del *software* orientado a objetos (abstracción, herencia, encapsulamiento, ocultación de la información) favorecen la reutilización
  - Alto grado de iteratividad y solapamiento
  - Desarrollo incremental
    - División del sistema en partes que se van desarrollando e integrando gradualmente por lo que el sistema puede ponerse en producción antes de estar totalmente terminado

Modelo de agrupamiento (i)

- Propuesto por **Bertrand Meyer** [Meyer, 1990]
- Concepto clave: **AGRUPAMIENTO (*cluster*)** [Meyer, 1999]
  - Unidad organizativa básica
  - Grupo de clases relacionadas o, recursivamente, *clusters* relacionados
  - Unidad natural para el desarrollo por parte de un único desarrollador
  - Evita el efecto todo-nada propio del modelo en cascada
- Tiene un componente secuencial y un componente concurrente
  - Existencia de diferentes subciclos de vida (uno para cada *cluster*) que pueden solaparse en el tiempo
  - Cada subciclo de vida que gobierna el desarrollo de un *cluster* está formado por
    - Especificación, Diseño, Implementación, Verificación/Validación y Generalización

Modelo de agrupamiento (ii)

- Enfoque ascendente
- La ocultación de la información posibilita la forma del modelo de *clusters* de ingeniería concurrente



Distribución temporal de las fases de cada agrupamiento

El proceso unificado (i)

- Definido por **Rational Software Corporation** [Jacobson et al., 2000]
  - Evolución del proceso Objectory de Rational
  - Utilización de UML [Booch et al., 1999] como lenguaje de modelado
  - Basado en componentes
- Características
  - **Conducido por casos de uso**
    - Los casos de uso se implementan para asegurar que toda la funcionalidad se realiza en el sistema y verificar y probar el mismo. Como los casos de uso contienen las descripciones de las funciones, afectan a todas las fases y vistas
  - **Centrado en la arquitectura**
    - La arquitectura se describe mediante diferentes vistas del sistema. Es importante establecer una arquitectura básica pronto, realizar prototipos, evaluarla y finalmente refinarla durante el curso del proyecto
  - **Iterativo e incremental**
    - Resulta práctico dividir los grandes proyectos en mini proyectos, cada uno de los cuales es una iteración que resulta en un incremento

El proceso unificado (ii)

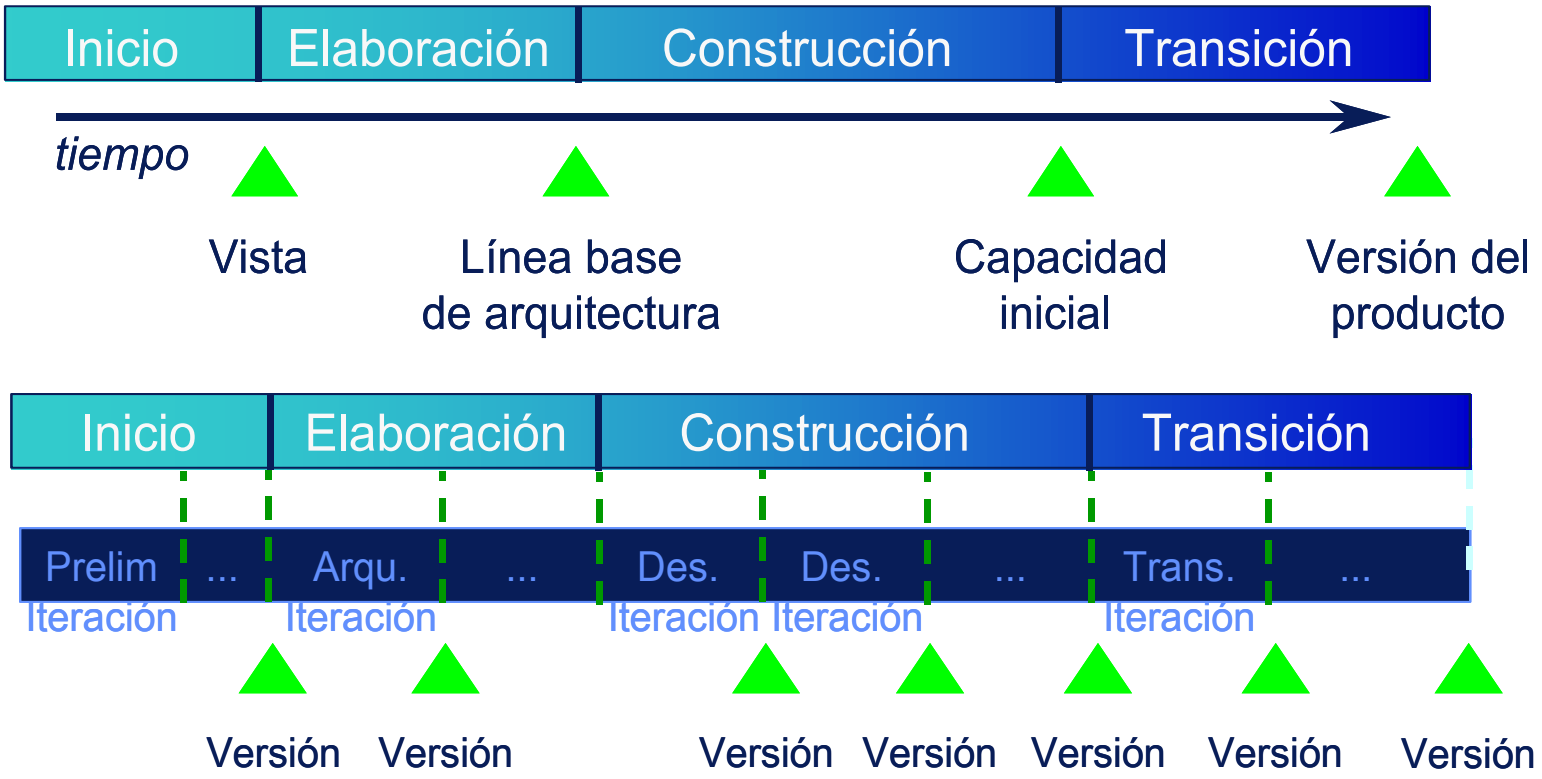
- El Proceso Unificado se repite a lo largo de una serie de ciclos
- Cada ciclo consta de cuatro **fases**
  - **Inicio**: se define el alcance del proyecto y se desarrollan los casos de negocio
  - **Elaboración**: se planifica el proyecto, se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura del sistema
  - **Construcción**: se construye el producto
  - **Transición**: el producto se convierte en versión beta. Se corrigen problemas y se incorporan mejoras sugeridas en la revisión

El proceso unificado (iii)

- Dentro de cada fase se puede, a su vez, descomponer el trabajo en **iteraciones** con sus incrementos resultantes
- Cada fase termina con un **hito**, cada uno de los cuales se caracteriza por la disponibilidad de un conjunto de componentes de *software*
  - Objetivos de los hitos
    - Toma de decisiones para continuar con la siguiente fase
    - Controlar el progreso del proyecto
    - Proporcionar información para la estimación de tiempo y recursos de proyectos sucesivos

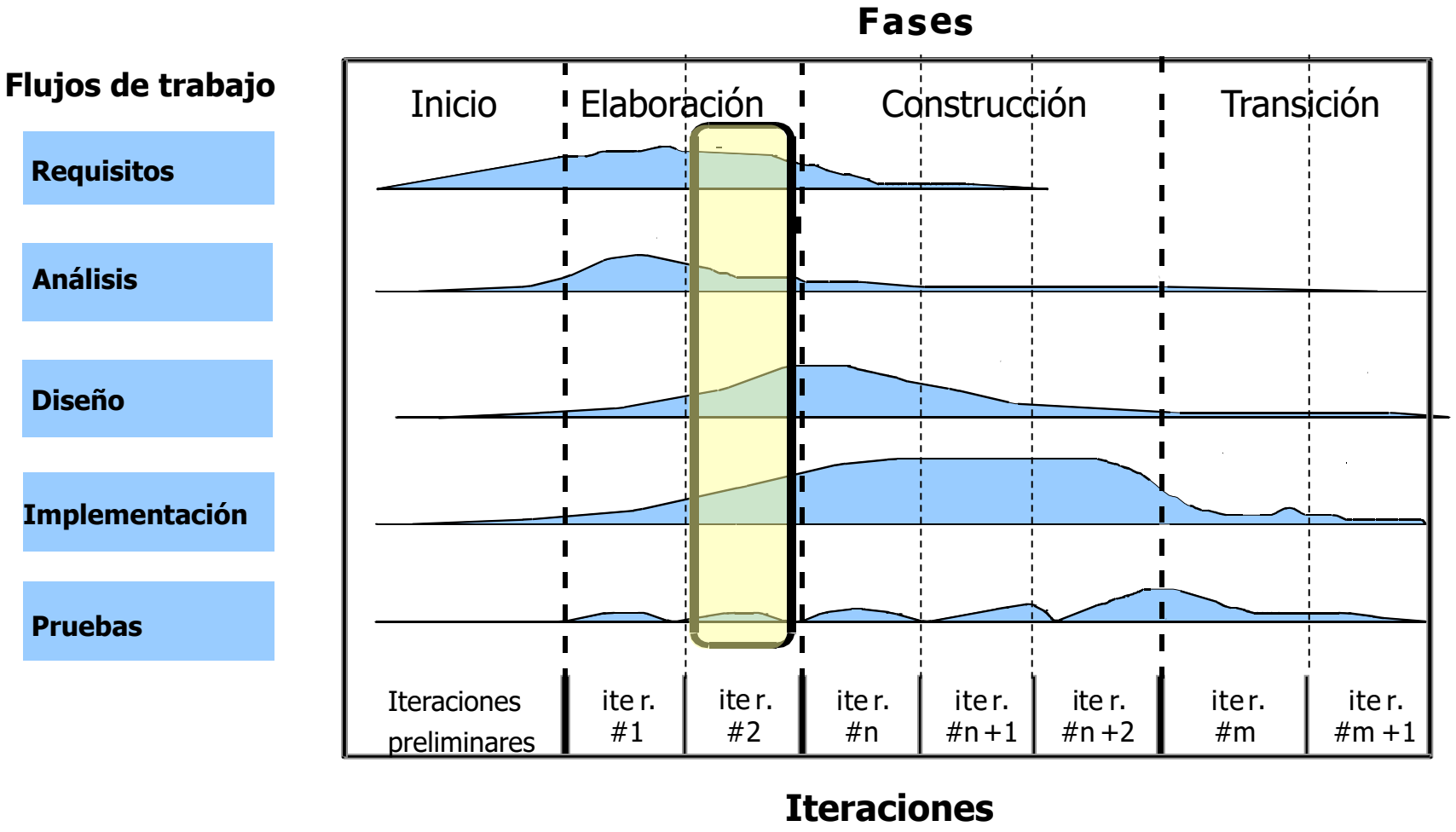
El proceso unificado (iv)

- Cada ciclo concluye con una versión del producto para los clientes



El proceso unificado (v)

- Las iteraciones discurren a lo largo de los flujos de trabajo



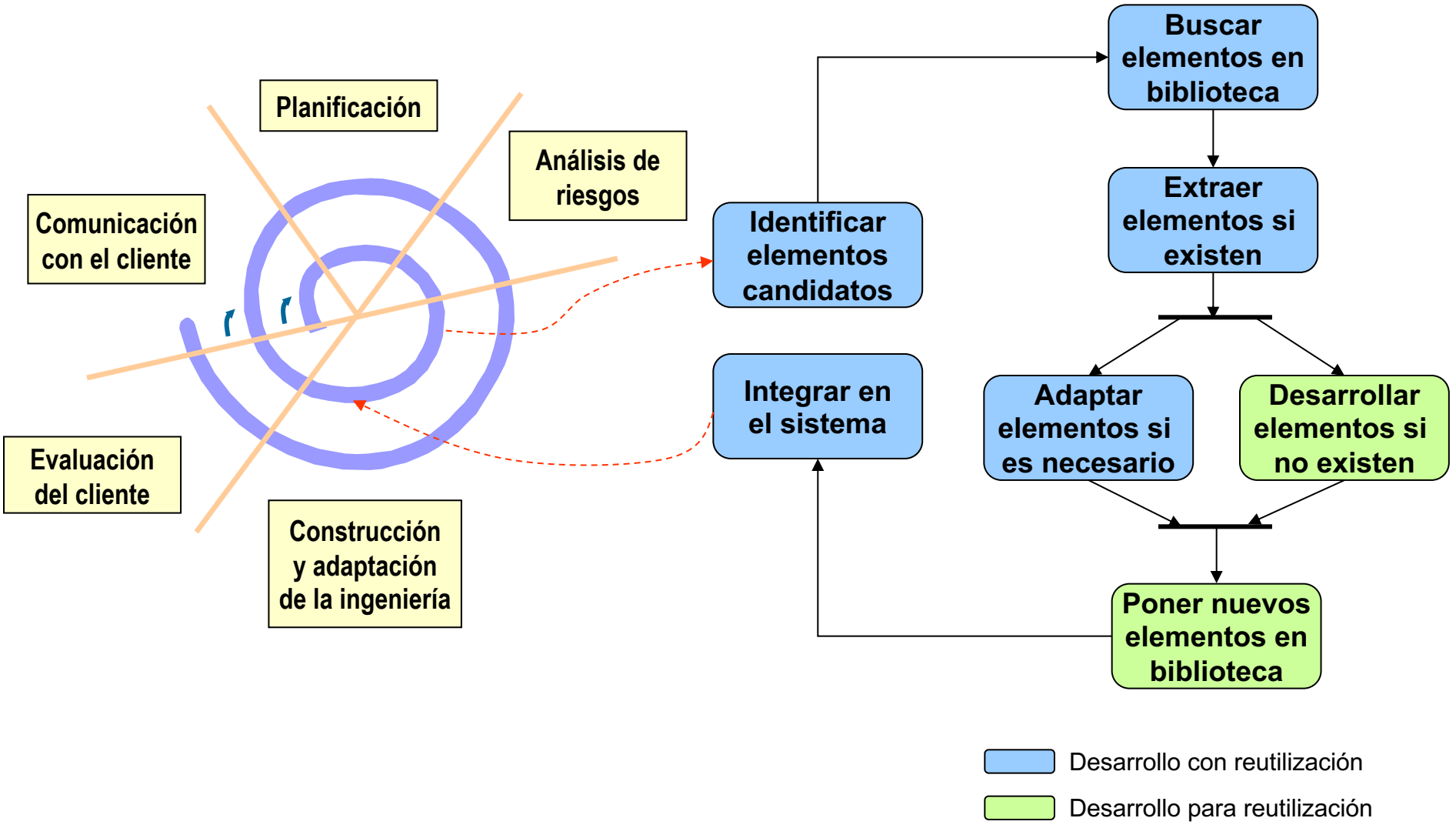
https://unsplash.com/photos/UOk1ghQ7juY

5. Modelos basados en reutilización

Modelos basados en reutilización (i)

- Enfoque de desarrollo que trata de maximizar la reutilización de *software* existente [Sommerville, 2002]
- Se basa en la existencia de un número significativo de elementos reutilizables
- El proceso de desarrollo del sistema se centra en la integración de estos elementos en un sistema, en lugar de desarrollarlo desde cero
- Incorpora muchas características del modelo en espiral
  - Es evolutivo por naturaleza [Nierstrasz et al., 1992] y existe un enfoque iterativo para la creación de *software*
- El proceso *software* tiende a estructurarse en dos subprocesos distintos y separados [Karlsson, 1995]
  - El desarrollo para reutilización
    - Construcción de elementos reutilizables dentro de un dominio concreto
  - El desarrollo con reutilización
    - Construcción de aplicaciones utilizando elementos reutilizables

Modelos basados en reutilización (ii)

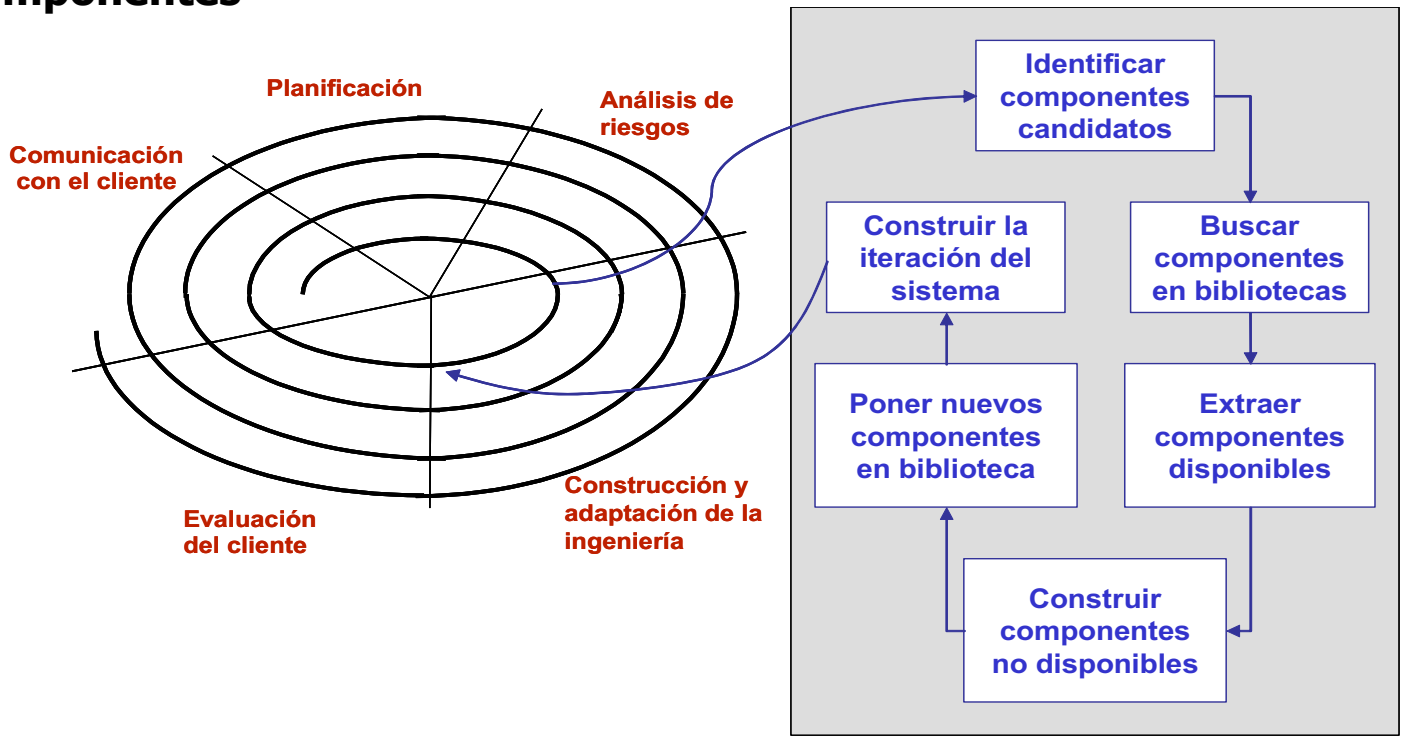


Modelos basados en reutilización (iii)

- Las **unidades *software* reutilizables** pueden ser de diferente tamaño
  - **Sistemas de aplicaciones**: se reutiliza la totalidad del sistema
    - Sin ningún cambio (reutilización de productos COTS)
    - Desarrollo de familias de aplicaciones para plataformas diferentes o necesidades específicas
  - **Componentes**: la reutilización va desde subsistemas hasta objetos simples
  - **Funciones**: componentes de *software* que implementan una sola función
- **Familias de aplicaciones** o líneas de productos: conjunto relacionado de aplicaciones que tiene una arquitectura común de dominio específico. Existen varios tipos de especialización
  - **De la plataforma**: varias versiones de la aplicación se desarrollan para diferente plataforma
  - **De la configuración**: se crean diferentes versiones para manejar diversos dispositivos periféricos
  - **De la funcionalidad**: diferentes versiones para clientes con distintos requisitos

Modelos basados en reutilización (iv)

- **Desarrollo basado en componentes** (i)
  - Configura aplicaciones a partir de componentes de *software* preparados [Pressman, 2002]
  - Enfoque iterativo y evolutivo
  - Se enmarca en un contexto más amplio: **Ingeniería del *Software* basada en componentes**



Modelos basados en reutilización (v)

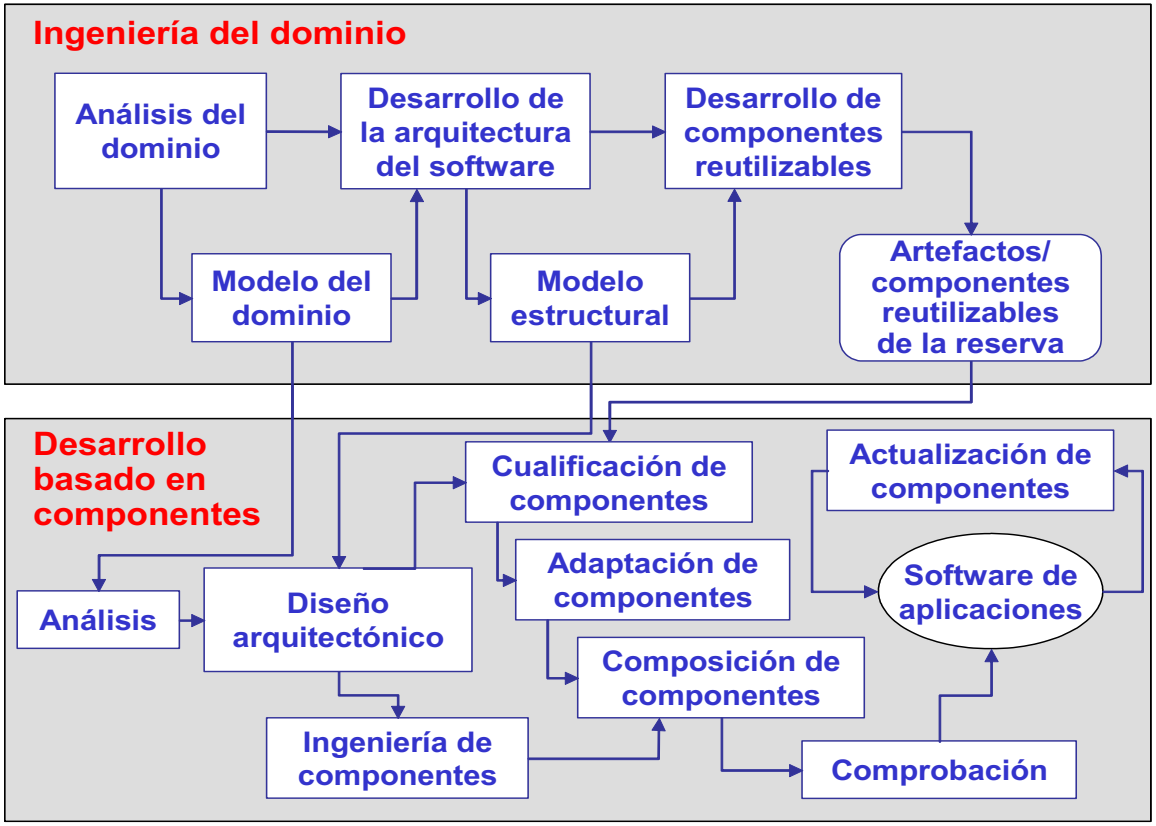
- **Desarrollo basado en componentes** (ii)
  - Un componente es una unidad ejecutable e independiente
  - Los componentes publican su interfaz y todas las interacciones son a través de ella
    - Una interfaz que se suministra define los servicios que ofrece el componente
    - Una interfaz que se solicita especifica qué servicios deben estar disponibles
  - Para el desarrollo con reutilización
    - Debe ser posible encontrar los componentes reutilizables apropiados
    - Se debe confiar en que los componentes que se utilizan se comportan conforme a lo especificado y son fiables
    - Los componentes deben tener documentación asociada para ayudar a comprenderlos y adaptarlos a una nueva aplicación

Modelos basados en reutilización (vi)

- **Ingeniería del *Software* basada en componentes** (i)
  - Ingeniería del dominio
  - Desarrollo basado en componentes

El objetivo de la **ingeniería del dominio** es identificar, construir, catalogar y diseminar un conjunto de componentes de software que tienen aplicación en el *software* actual y futuro dentro de un dominio de aplicación particular

[Presman, 2001]



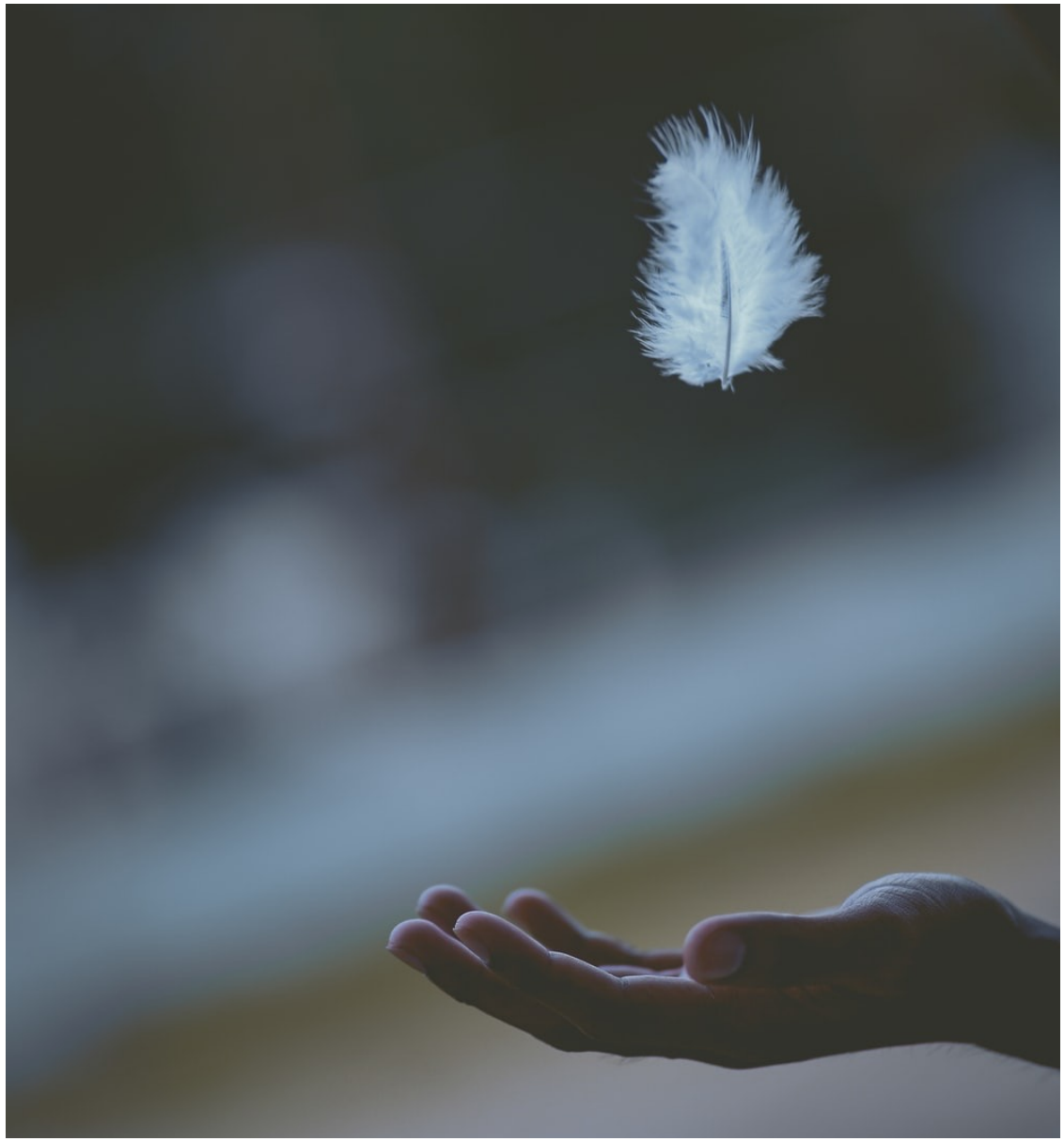
Ingeniería del *software* basada en componentes

Modelos basados en reutilización (vii)

- **Ingeniería del *Software* basada en componentes** (ii)
  - **Actividades de la ingeniería del dominio**
    - **Análisis del dominio**
      - Definir el dominio a investigar
      - Categorizar los elementos extraídos del dominio
      - Recoger una muestra representativa de las aplicaciones del dominio
      - Analizar cada aplicación de la muestra
      - Desarrollar un modelo de análisis para los objetos
      - Definir un lenguaje del dominio: hace posible la especificación y construcción posterior de aplicaciones dentro del dominio
    - **Modelo del dominio**: resultado de las actividades anteriores
    - **Modelado estructural**: enfoque de ingeniería basado en ***tramas*** que opera efectuando la suposición consistente de que todo dominio de aplicación posee tramas repetidas (de función, de datos y de comportamiento) que tienen un potencial de reutilización
      - Todo dominio de aplicación se puede caracterizar por un modelo estructural
      - Un modelo estructural es un ***estilo arquitectónico reutilizable***

Modelos basados en reutilización (viii)

- **Ingeniería del *Software* basada en componentes** (iii)
  - **Actividades del desarrollo basado en componentes**
    - **Cualificación de componentes**: asegura que un componente candidato llevará a cabo la función necesaria, encajará en el estilo arquitectónico del sistema y tendrá la calidad requerida
    - **Adaptación de componentes**: elimina conflictos de integración
      - Enmascaramiento de caja blanca, gris o negra
    - **Composición de componentes**: ensambla componentes cualificados, adaptados y diseñados para la arquitectura establecida
    - **Ingeniería de componentes**: diseño de componentes para su reutilización
    - **Actualización de componentes**: el *software* actual se reemplaza a medida que se dispone de nuevas versiones de componentes



https://unsplash.com/photos/FL6rma2iePU

6. Procesos ágiles

Procesos ágiles

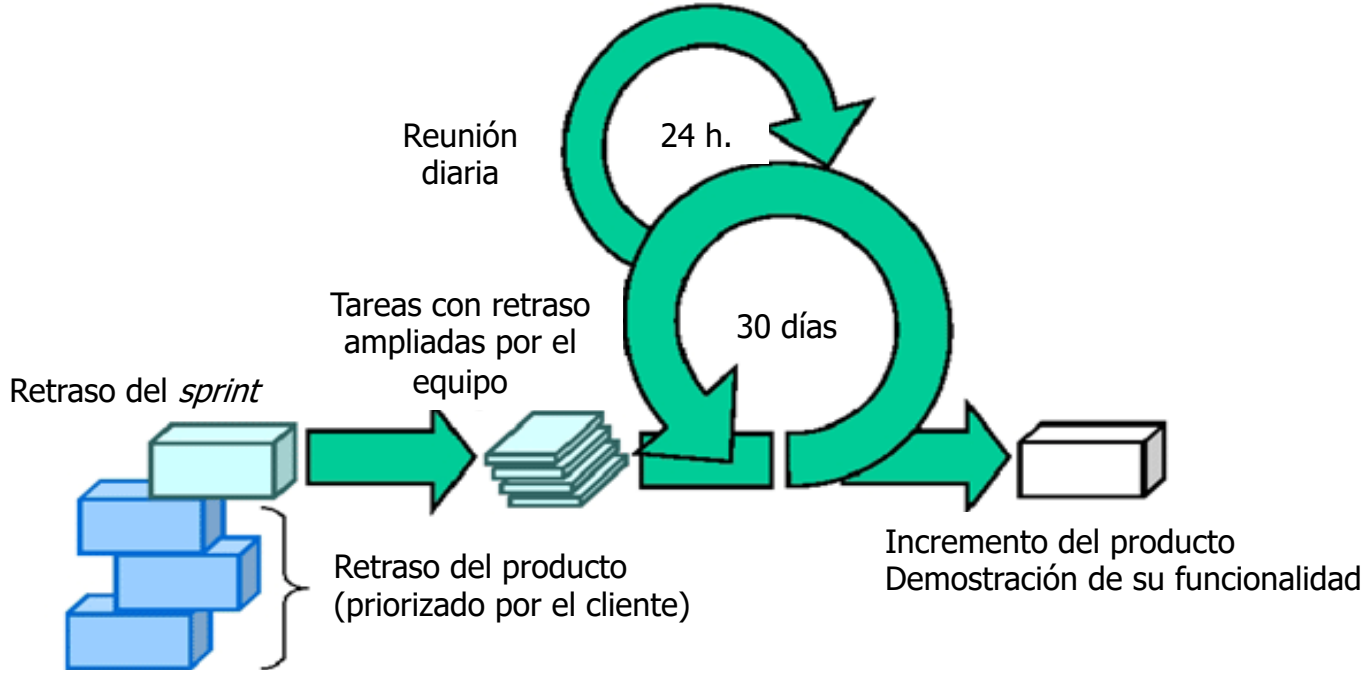
- Constituyen un nuevo enfoque en el desarrollo de *software* cuyas principales características son
  - Menor énfasis en el análisis, diseño y documentación
  - Equipos pequeños
  - Desarrollo incremental
  - Programación (planificación temporal) en cajas de tiempo
  - Supervivencia en un entorno caótico
- Diversos enfoques
  - XP (*eXtreme Programming*) [Beck, 1999]
  - *Scrum* [Schwaber, 1995]
  - *Crystal* [Cockburn, 1999]
  - Proceso *Software* Adaptativo [Highsmith, 2000]
  - Proceso Unificado Ágil (http://www.ambysoft.com/unifiedprocess/agileUP.html)

Scrum (i)

- Propuesto por Jeff Sutherland y desarrollado por Shwaber y Beedle
- Actividades estructurales
  - Requisitos
  - Análisis
  - Diseño
  - Evolución
  - Entrega
- Dentro de cada actividad las tareas se organizan con un patrón de proceso denominado ***sprint***
- El trabajo del *sprint* se adapta al problema y se define y modifica en tiempo real por el *equipo Scrum*
- Uso de patrones de proceso de demostrada eficacia en proyectos críticos, con plazos cortos y requisitos cambiantes

Scrum (ii)

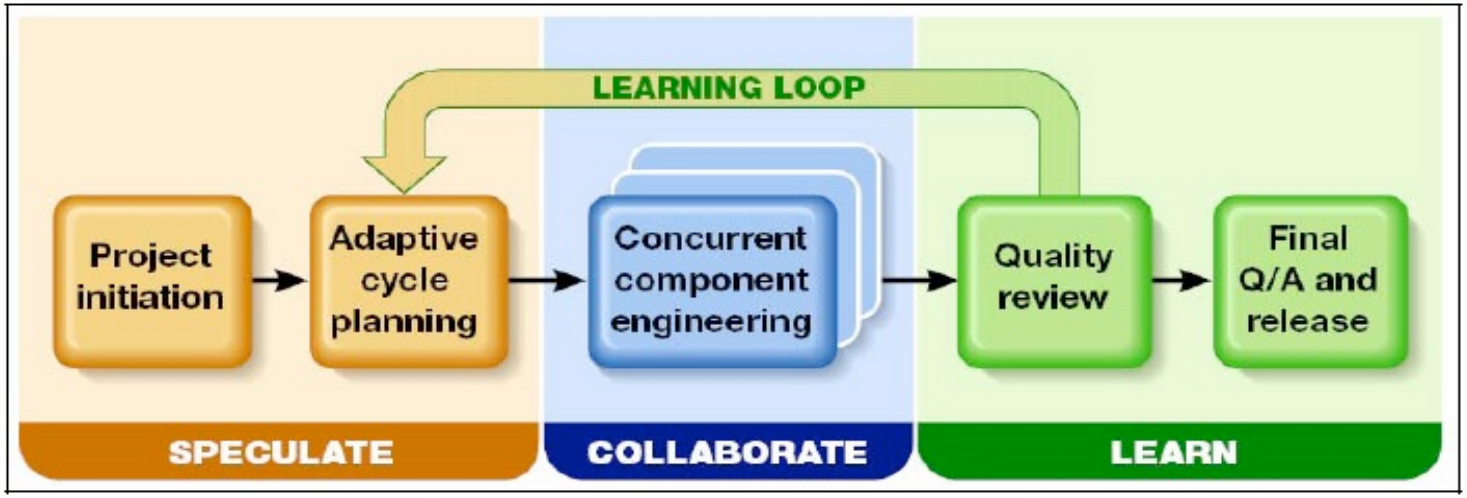
- Acciones de los patrones de proceso
  - ***Retraso***: priorización de requisitos
  - ***Sprints***: unidades de trabajo requeridas para alcanzar un requisito
  - ***Reuniones Scrum***: reuniones breves dirigidas por el *maestro Scrum*
  - ***Demostraciones preliminares***: entrega de un incremento al cliente

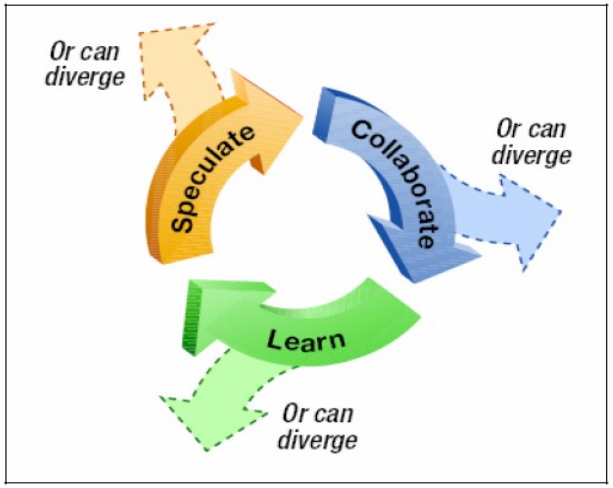


Desarrollo de software adaptativo (i)

- Modelo ágil y adaptativo basado en la colaboración y orientado al desarrollo de sistemas complejos
- **Fases del ciclo de vida**
  - **Especulación**
    - Inicio del proyecto
    - Planificación del ciclo adaptativo: enunciado, restricciones y requisitos básicos
      - Plan de lanzamiento: definición de un conjunto de ciclos (incrementos)
  - **Colaboración**
    - Construir la funcionalidad definida en la fase anterior
    - Uso de técnicas JAD (*Joint Application Development*) y trabajo colaborativo
  - **Aprendizaje**
    - Revisión de calidad al final de cada ciclo
    - Aprendizaje
      - Grupos enfocados
      - Revisiones técnicas formales
      - Postmortem

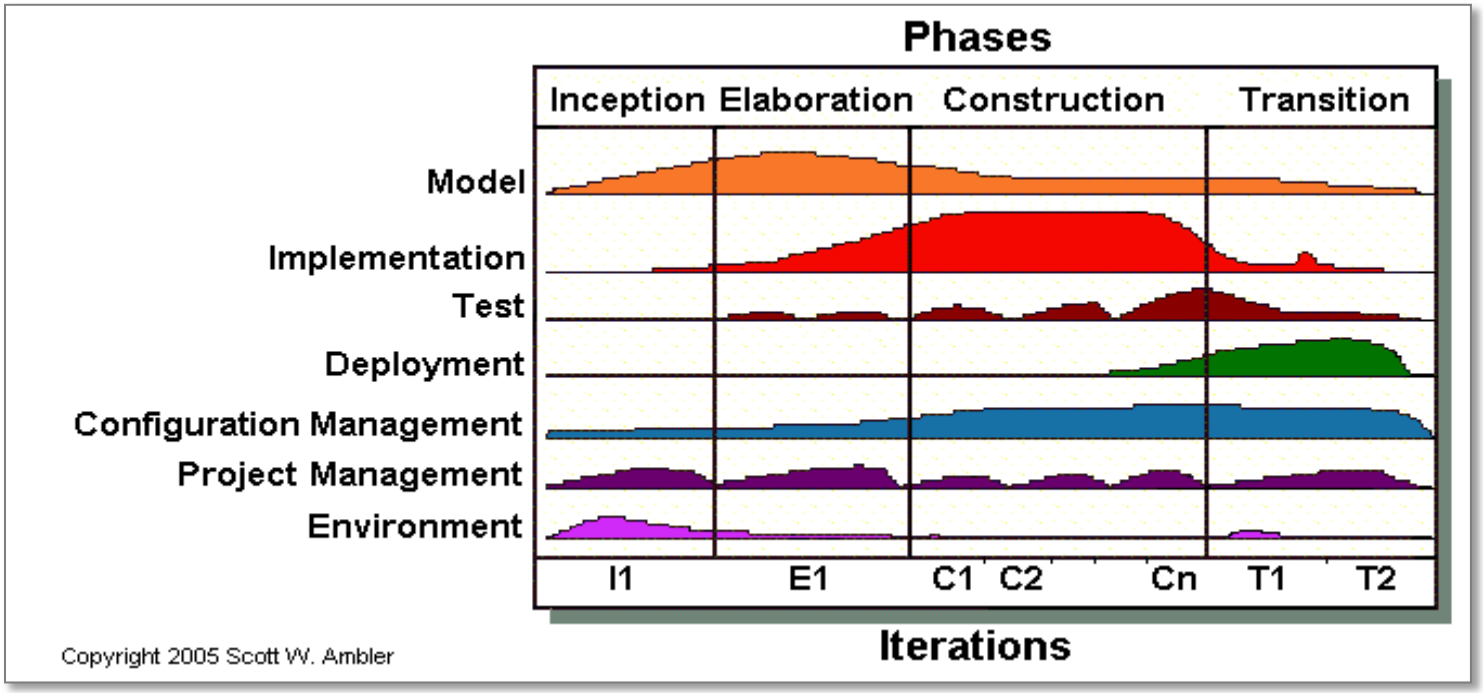
Desarrollo de software adaptativo (ii)





Proceso Unificado Ágil (i)

- Versión simplificada del Proceso Unificado (PU) desarrollada por Scott W. Ambler [1994]
- Adaptación de PU a los principios de los procesos ágiles

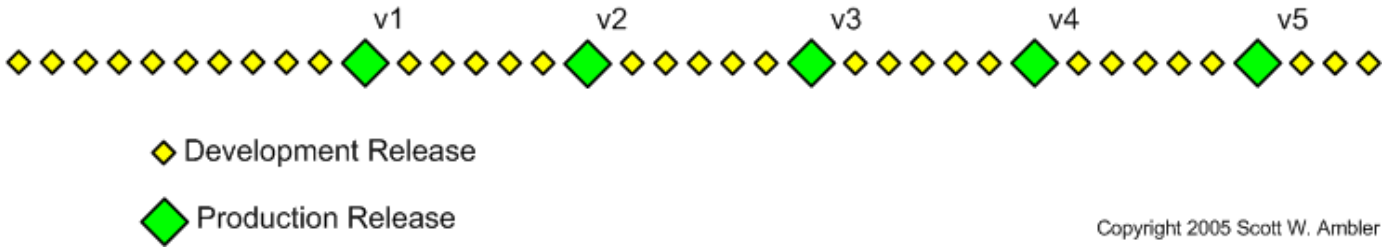


Proceso Unificado Ágil (ii)

- **Disciplinas**
  - **Modelo**: estudio del dominio del problema e identificación de una solución viable
  - **Puesta en práctica**: transformación de los modelos en código ejecutable y realización un nivel básico de la prueba
  - **Prueba**: realización de una evaluación objetiva para asegurar calidad
  - **Despliegue**: planificación y entrega del sistema a los usuarios finales
  - **Gestión de la configuración**: control de cambios y gestión de versiones
  - **Gestión del proyecto:** gestión del riesgo, de recursos, de costes, organización de tareas, seguimiento, etc.
  - **Ambiente**: actividades para asegurar que el proceso, las normas (estándares y directrices), y las herramientas apropiados (*hardware*, *software*, etc.) están disponibles

Proceso Unificado Ágil (iii)

- Lanzamiento de versiones incrementales a lo largo del tiempo
  - Versiones de desarrollo
    - Se lanzan al final de cada iteración
  - Versiones de producción
    - Son versiones de desarrollo que han pasado procesos de aseguramiento de la calidad, prueba y despliegue
    - Se lanzan con menos frecuencia que las de desarrollo



https://unsplash.com/photos/ZArDeAtxi0Q

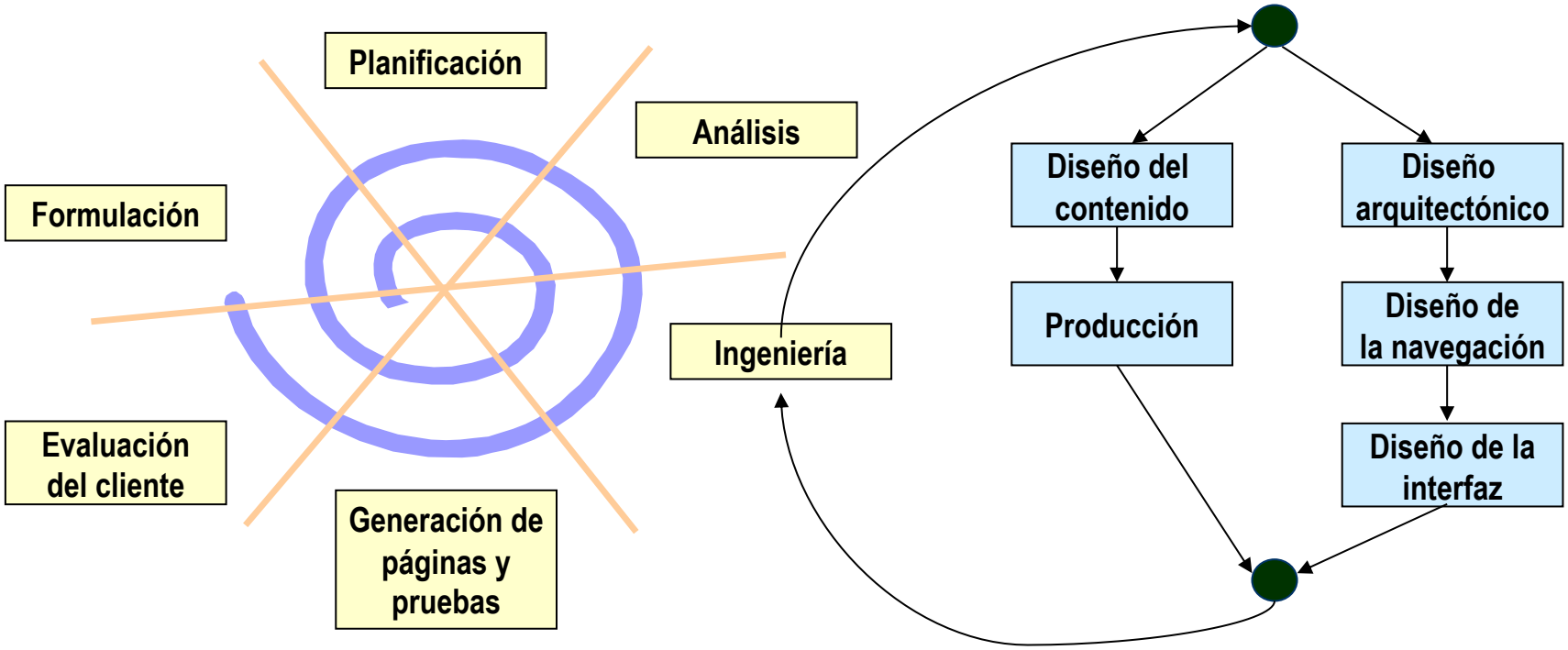
7. Modelos para la ingeniería web

Modelos para la Ingeniería Web

- Las **características** de sistemas y aplicaciones basados en Web influyen enormemente en el proceso de Ingeniería Web (IWeb)
  - Intensivas de red
  - Controladas por contenido
  - Evolución continua
  - Inmediatez
  - Estética…
- El ciclo de desarrollo de una aplicación web consta de las siguientes fases de ingeniería
  - Definición y análisis de los sistemas web
  - Diseño de los sistemas web
    - Diseño arquitectónico
    - Diseño de la navegación
    - Diseño de la interfaz
  - Pruebas de las aplicaciones web

Modelos de Pressman (i)

- Pressman propuso inicialmente el modelo IWEB [Pressman, 2002] tomando como base su modelo de ciclo de vida en espiral



Modelo de proceso de IWEB [Pressman, 2002]

Modelos de Pressman (ii)

- Etapas del modelo IWEB
  - **Formulación**: identificación de metas y objetivos
  - **Planificación**: estimación de costes, evaluación de riesgos y planificación temporal del proyecto
  - **Análisis**: establecimiento de requisitos
  - **Ingeniería**: dos grupos de tareas paralelas,
    - Técnicas (diseño arquitectónico, de navegación y de interfaz)
    - No técnicas (diseño del contenido y producción)
  - **Generación de páginas y pruebas**
    - El contenido se fusiona con los diseños arquitectónico, de navegación y de interfaz para elaborar páginas web ejecutables en HTML, JSP...
    - Integración con el *software* intermedio (*middleware*) de componentes
  - **Evaluación con el cliente**: revisión de cada incremento y solicitud de cambios

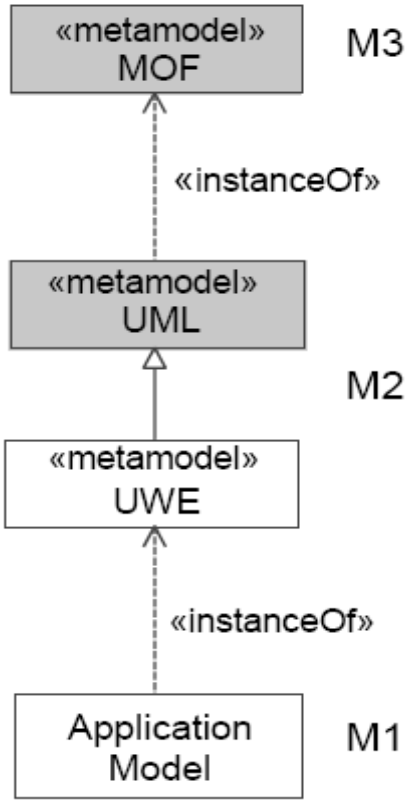
Modelos de Pressman (iii)

- Pressman propuso posteriormente una modificación del modelo IWEB [Pressman, 2006] con las siguientes actividades
  - **Comunicación con el cliente**
    - Análisis de negocio
    - Formulación
  - **Planificación**: definición de tareas y calendario para el desarrollo de un incremento
  - **Modelado**: las actividades de análisis y diseño convencionales se adaptan y se funden con las específicas de las aplicaciones Web
  - **Construcción**: construcción y prueba de un incremento
  - **Despliegue**
    - Configuración de la aplicación para su ambiente operativo
    - Entrega a los usuarios
    - Evaluación

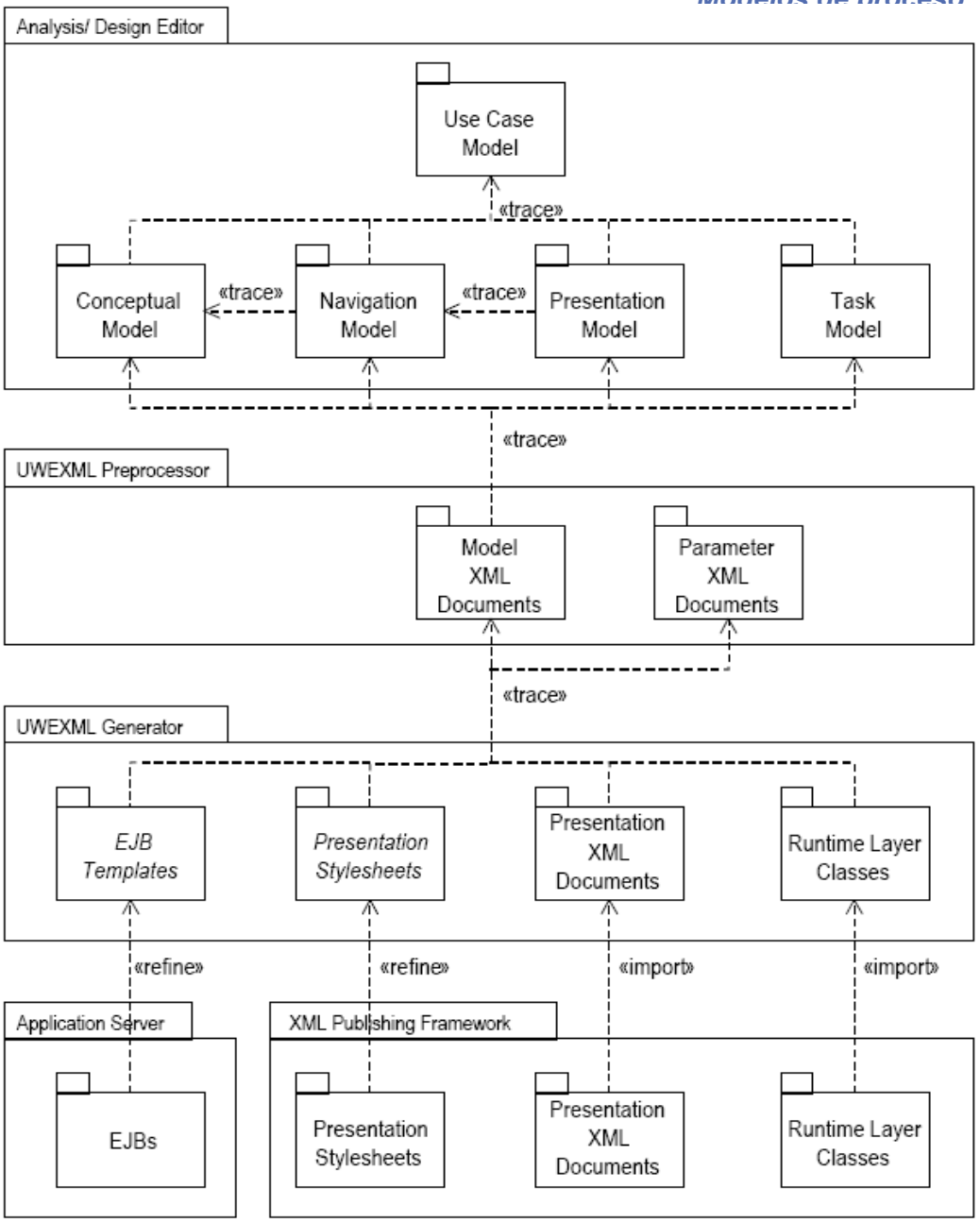
  Las actividades se realizan siguiendo un flujo de proceso incremental

UWE (UML-based Web Engineering) (i)

- Características [Koch, 2001], [Hennicker y Koch, 2000]
  - **Desarrollo iterativo e incremental:** basado en el Proceso unificado
  - **Uso de UML:** perfil UML propio
  - **Centrado en la sistematización y automatización**
    - Proceso sistemático de diseño
    - Generación semiautomática de aplicaciones web a través de un *framework* de publicación XML (UWEXML)
  - **UWE comprende**
    - Una **notación**
    - Un **método**
    - Un **metamodelo**
    - Un **proceso de desarrollo**
    - Una **herramienta CASE**



UWE (UML-based Web Engineering) (ii)



Vista general del proceso UWE

https://unsplash.com/photos/7tkDoo2L_Eg

8. Aportaciones principales del tema

Aportaciones principales (i)

- Existen muchas propuestas de modelos de proceso para sistematizar el desarrollo *software*
- Se pueden establecer varias categorías de modelos aunque no existe una clara separación entre ellas pudiendo haber modelos de proceso que pertenezcan a más de una categoría
- Los modelos tradicionales, que incluyen el modelo clásico y algunas variantes del mismo, se caracterizan por el establecimiento de fases secuenciales
- La rigidez de los modelos tradicionales y su dificultad para llevarlos a cabo en proyectos reales ha dado lugar a la propuesta de otros modelos más flexibles
- Los modelos evolutivos consideran que el *software* puede cambiar a lo largo del tiempo por lo que permiten desarrollar versiones de forma iterativa e incremental

Aportaciones principales (ii)

- Se han propuesto modelos específicos para sistemas orientados a objetos que incorporan algunas características de los modelos evolutivos
- Las particularidades del *software* orientado a objetos favorecen la reutilización de elementos. Esto ha dado lugar a la propuesta de modelos centrados en la reutilización sistemática del *software*
- Los procesos ágiles surgen como respuesta a la excesiva formalización del proceso y en contraposición a los modelos "pesados" como el Proceso Unificado. Prestan menos atención al análisis, diseño y documentación y más a la programación
- Los modelos más recientes son los dedicados al desarrollo de sistemas web. Son modelos que se adaptan a las peculiaridades de este tipo de aplicaciones: Controladas por contenido, evolución continua, inmediatez, etc.

https://unsplash.com/photos/hhq1Lxtuwd8

9. Cuestiones y ejercicios

Cuestiones y ejercicios

- Considerar los modelos de proceso presentados, ¿cuáles permiten una mayor capacidad de reacción ante requisitos cambiantes?
- Explicar cómo el modelo en cascada y el modelo de prototipos pueden integrarse en el modelo en espiral
- Comentar ventajas e inconvenientes de los procesos ágiles respecto a otros modelos de proceso más formales
- Identificar actividades de los modelos de proceso para la ingeniería web que no se encuentran presentes en otros modelos
- ¿Qué tipo de modelo de proceso sería más conveniente para una aplicación en la que se va a probar una tecnología nueva y los requisitos no están claramente definidos? Razonar la respuesta
- Una empresa recibe el encargo de desarrollar aplicaciones de gestión de matrícula para varias universidades. Parte de la funcionalidad es común a todas las aplicaciones y parte es específica de cada una. Además, se requiere desarrollar versiones para diferentes plataformas. ¿Qué modelo de proceso deberían elegir? Razonar la respuesta

https://unsplash.com/photos/YLSwjSy7stw

10. Lecturas complementarias

Lecturas complementarias

- B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah y R. Madachy, "Using the WinWin spiral model: a case study," *Computer,* vol. 31, no. 7, pp. 33-44, 1998. doi: 10.1109/2.689675. Disponible en: https://goo.gl/Cr3CVY
  - En este artículo se presenta la aplicación práctica del modelo de ciclo de vida en espiral WinWin, una extensión del ciclo de vida definido por Boehm, al que se le ha añadido las actividades de la *Teoría W* al comienzo de cada ciclo
- I. Gutiérrez y N. Medinilla, "Contra el arraigo de la cascada," en *Actas de las IV Jornadas de Ingeniería del Software y Bases de Datos, JISBD'99 (24-26 de noviembre de 1999, Cáceres - España)*, P. Botella, J. Hernández y F. Saltor, Eds. pp. 393-404, 1999
  - Trabajo crítico con el modelo de ciclo de vida en cascada, realizado desde la perspectiva de la complejidad de la incertidumbre en los proyectos *software*
- B. Henderson-Sellers y J. M. Edwards, "The object-oriented systems life cycle," *Communications of the ACM,* vol. 33, no. 9, pp. 142-159, 1990. doi: 10.1145/83880.84529
  - En este artículo se describe el modelo de ciclo de vida fuente para desarrollos orientados a objetos

https://unsplash.com/photos/8muUTAmcWU4

11. Referencias

Referencias (i)

[Ambler, 1994] Ambler, S. W. "*In search of a generic SDLC for object systems*". Object Magazine, 4(6): 76-78, 1994

[Beck, 1999] Beck, K. "*Embracing Change with Extrem Programming*", IEEE Computer 32, pp. 70-77, 1999

[Beck, 2000] Beck, K. "*Extreme Programming Explained. Embrace Change*". Addison-Wesley, 2000

[Birrel y Ould, 1985] Birrel, N. D., Ould, M.A. "*A practical Handbook for Software Development*". Cambridge University Press, 1985

[Boehm, 1986] Boehm, B. W. "*A Spiral Model of Software Development and Enhancement*". ACM Software Engineering Notes, 11(4):22-42. 1986

[Boehm, 1988] Boehm, B. W. "*A Spiral Model of Software Development and Enhancement*". Computer , 21(5): 61-72 , 1988

[Boehm et al., 1998] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J., Madachy, R. "*A Stakeholder Win-Win Approach to Software Engineering Education*". Annals of Software Engineering , 6, 295-321, 1998

[Booch et al., 1999] Booch, G., Rumbaugh, J., Jacobson, I. "*El Lenguaje Unificado de Modelado*". Addison Wesley, 1999

[Butler, 1994] Butler, J. "*Rapid Application Development in Action*". Managing System Development, Applied Computer Research, 14(5):6-8. May, 1994

[Cockburn, 1999] Cockburn, A. "Software Development as a Cooperative Game", Humans and Tecnology inc., 1999

Referencias (ii)

[Curtis et al., 1987] Curtis, B., **Krasner, H., Shen, V., Iscoe, N.** *"On Building Software Process Models under the Lamppost"*. En Proceedings of the 9th International Conference on Software Engineering, pp. 96-103, IEEE CS Press, 1987

[GMD, 1992] German Ministry of Defense. "*V-Model: Software Lifecycle Process Model*". Bundesminister des Innern, Koordinierungs-und Beratungstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, 1992

[Graham, 1996] Graham, I. "*Métodos orientados a objetos*", Adison-Wesley, 1996.

[Hennicker y Koch, 2000] Hennicker, R. y Koch, N. "*A UML-based Methodology for Hypermedia Design*". En *Proceedings of the Unified Modeling Language Conference (UML'2000)*. A. Evans y S. Kent (Eds.). Lecture Notes in Computer Science LNCS Vol. 1939. Páginas 410-424. Springer-Verlag, 2000

[Henderson-Sellers , 1993] Henderson-Sellers, B., Edwards, J. M. "*The fountain Model for object-oriented systems development*", Object Magazine, julio/agosto, pp 71-79, 1993

[Henderson-Sellers y Edwards , 1990] Henderson-Sellers, B., Edwards, J. M. "*The object-oriented systems life cycle*", Communications of the ACM, 33(9): 143-159, 1990

[Highsmith, 2000] Highsmith, J., "*Adaptive Software Development: A Colaborative Approach to Managing Complex Systems*", Dorset House, 2000

[Jacobson et al., 2000] Jacobson, I., Booch, G., Rumbaugh, J. *"El Proceso Unificado de Desarrollo"*, Addison Wesley, 2000

Referencias (iii)

[Karlsson, 1995] Karlsson, E.-A. (Ed). "*Software Reuse. A Holistic Approach*". Wiley Series in Software Based Systems. John Wiley and Sons Ltd., 1995

[Kerr y Hunter, 1994] Kerr, J., Hunter, R. "*Inside RAD*", McGraw-Hill, 1994

[Koch, 2001] Koch, N. "*Software Engineering for Adaptive Hypermedia Applications. Reference Model, Modeling Techniques and Development Process*". PhD. Thesis, Ludwig-Maximilians-Universität München, 2001

[Kruchten, 1991] Kruchten, P. "*Un processus de développement de logiciel itératif et centré sur lárchitecture*", Proceedings of the 4th International Conference on Software Engineering. Toulouse, Paris, 1991

[Martin, 1981] Martin, J. "*Rapid Application Development*", Prentice Hall, 1991.

[McCracken y Jackson, 1981] McCracken, D. D., Jackson, M. A. "*A Minority Dissenting Opinion*". En W. W. Cotterman, J. D. Couger, N. L. Enger, F. Harold (Eds.). Systems Analysis and Design: A Foundation for the 1980s, pp. 551-553, New York: Elsevier, 1981

[McDermid y Rook, 1993] McDermid, J., Rook, P. "*Software Development Process Models*". En McDermid, J. (Ed.) The Software Engineer's Reference Book. CRC Press, Páginas 15-28. 1993

[Meyer, 1990] Meyer, B. "*La Nueva Cultura del Desarrollo de Software*", Systems, pp. 12-13. Septiembre, 1990

Referencias (iv)

[Meyer, 1999] Meyer, B. "*Construcción de software orientado a objetos*", Prentice Hall, 1999

[Mills et al., 1987] Mills, H. D., Dyer, M., Linger, R. "*Cleanroom Software Engineering*", IEEE Software, 4(5): 19-25. September 1987

[Muller, 1997] Muller, P. A. "*Modelado de objetos con UML*". Eyrolles-Ediciones Gestión 2000, 1997

[Nierstrasz et al., 1992] Nierstrasz,O., Gibbs, S.J., Tsichritzis, D. "*Component-Oriented Software Development*". CACM, 35(9): 160-165, 1992.

[Pfleeger, 2002] Pfleeger, S. L. "*Ingeniería del Software. Teoría y Práctica*". Prentice Hall, 2002

[Pressman, 2002] Pressman, R. S. "*Ingeniería del Software, un enfoque práctico*", 5ª Edición. Mc Graw Hill, 2002.

[Pressman, 2006] Pressman, R. S. "*Ingeniería del Software, un enfoque práctico*", 6ª Edición. Mc Graw Hill, 2006.

[Pressman, 2010] Pressman, R. S. "*Ingeniería del Software: Un Enfoque Práctico*". 7ª Edición. McGraw-Hill. 2010

[Royce, 1970] Royce, W. W. "*Managing the Development of Large Software Systems: Concepts and Techniques*", In Proceedings WESCON. August, 1970

[Rumbaugh, 1992] Rumbaugh, J. "*Over the waterfall and into the whirlpool*", JOOP, mayo, pp 23-26, 1992

Referencias (v)

[Sommerville, 2002] Sommerville, I. "*Software Engineering*", 6th ed., Addison Wesley, 2001

[Sommerville, 2005] Sommerville, I. "*Software Engineering*", 7th ed., Addison Wesley, 2005

[Schwaber, 1995] Schwaber, K. "*SCRUM Development Process".* OOPSLA'95 Workshop on Business Object Design and Implementation, 10 Dec 1995

[Turk et al., 2002] Turk, D., France, R. y Rumpe, B. "*Limitations of Agile Software Processes".* En Proceedings of 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering, XP2002. (Alghero, Sardinia, Italy, April 2002), pp. 43-46, 2002

[Ward y Kroll, 1999] Ward, S. , Kroll, P. "*Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process*", Rational Software & Context Integration white paper, 1999

[Wolff, 1989] Wolff, J. G. "*The Management of Risk System Development: 'Project SP' and the 'New Spiral Model"*. Software Engineering Journal, May 1989

INGENIERÍA DE SOFTWARE I

Tema 3: Modelos de proceso

Grado en Ingeniería Informática
Fecha de última modificación: 6-2-2024

Dr. Francisco José García-Peñalvo / fgarcia@usal.es
Dra. Alicia García-Holgado / aliciagh@usal.es
Dra. Andrea Vázquez-Ingelmo

Departamento de Informática y Automática
Universidad de Salamanca