Bloque I

Tema 2

El proceso de desarrollo del software. Paradigmas o modelos de desarrollo del Software

Contenido

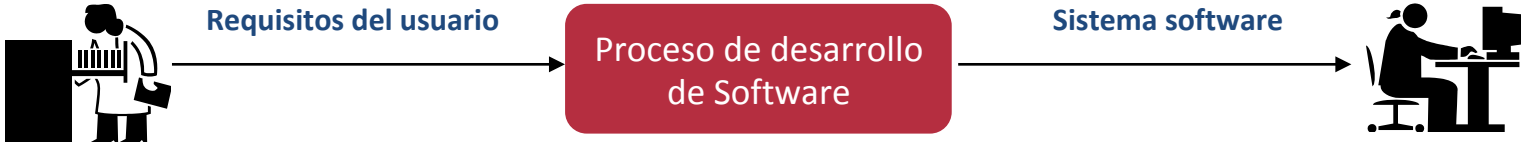
- El Proceso: Modelos de Desarrollo
- Paradigmas o Modelos de Desarrollo de Software
  - Modelos de Ciclos de Desarrollo
  - Modelos de Ciclos de Evolución
  - Modificadores de los Modelos
- Modelos de Procesos Ágiles
- Creación del Modelo del Ciclo de Vida

El Proceso: Modelos de Desarrollo

- Proceso
  - Conjunto ordenado de tareas, una serie de pasos que involucran actividades, restricciones y recursos, que producen una salida determinada
  - Proceso de software: conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software
  - Características
    - ✓ Tiene una serie de actividades principales
    - ✓ Utiliza recursos, está sujeto a restricciones y genera productos intermedios y finales
    - ✓ Compuesto por subprocesos que se encadenan de alguna forma
    - ✓ Cada actividad tiene sus criterios de entrada y salida, que permiten conocer cuando comienza y termina dicha actividad
    - ✓ Existen principios orientadores que explican las metas de cada actividad
  - Cuando implica la construcción de un producto, se suele llamar *ciclo de vida*
  - Aportan consistencia y estructura sobre el conjunto de actividades, lo que permite realizar la misma tarea correctamente de forma repetida
  - Existen diferentes modelos de proceso

El Proceso: Modelos de Desarrollo

- Desde el punto de vista del estándar ISO/IEC 12207 1995 un proceso es un conjunto de actividades y tareas relacionadas, que al ejecutarse de forma conjunta transforman una entrada en una salida



Procesos primarios del ciclo de vida del software

- Adquisición: Proceso global que sigue el adquiriente para obtener el producto
- Suministro: Proceso global que sigue el suministrador para proporcionar el producto
- Desarrollo: Proceso empleado por el suministrador para el diseño, construcción y pruebas del producto
- Operación: Proceso seguido por el operador en el "día a día" para el uso del producto
- Mantenimiento: Proceso empleado para mantener el producto, incluyendo tanto los cambios en el propio producto como en su entorno de operación producto

El Proceso: Modelos de Desarrollo

Procesos de soporte del ciclo de vida del software

El estándar 12207 identifica los procesos de soporte que pueden ser utilizados desde un proceso primario, o incluso desde otro proceso de soporte. Los procesos de soporte son:

- Documentación: Actividades empleadas para registrar información específica empleada por otros procesos
- Gestión de la Configuración: Actividades empleadas para mantener un registro de los productos generados en la ejecución de los procesos
- Aseguramiento de la Calidad: Actividades empleadas para garantizar de forma objetiva que el producto y los procesos asociados son conformes a los requisitos documentados y a las planificaciones
- Verificación: Actividades empleadas para verificar el producto
- Validación: Actividades empleadas para validar el producto
- Reuniones de Revisión: Reuniones empleadas por las dos partes para evaluar el estado del producto y de las actividades
- Auditorias: Actividades para determinar que el proyecto cumple con los requisitos, planes y contratos
- Resolución de Problemas: Actividades para analizar y resolver problemas relativas al proyecto, sea cual sea su fuente y naturaleza

El Proceso: Modelos de Desarrollo

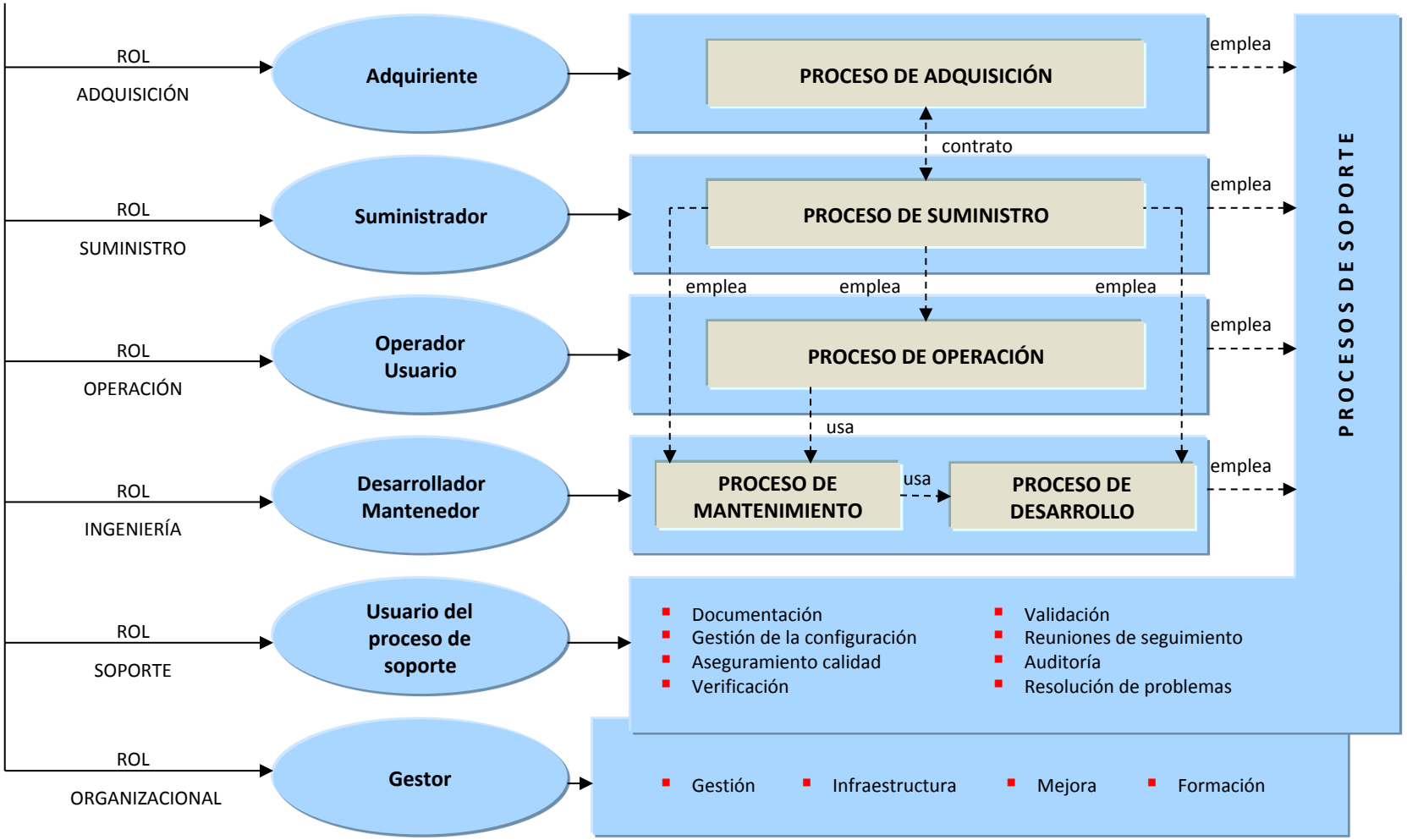
Procesos organizacionales

El estándar 12207 identifica los procesos que deben realizarse en el contexto de la organización que va a ejecutar el proyecto.
Normalmente estos procesos se aplican de forma común sobre múltiples proyectos. De hecho las organizaciones más maduras los identifican e institucionalizan

- Gestión: Describe las actividades de gestión de la organización, incluyendo también la gestión de proyectos.
- Infraestructura: Actividades necesarias para que puedan realizarse otros procesos del ciclo de vida. Incluye entre otros el capital y el personal
- Mejora: Actividades realizadas para mejorar la capacidad del resto de procesos
- Formación

El Proceso: Modelos de Desarrollo

VISIÓN GENERAL DE LOS PROCESOS, RELACIONES Y ROLES

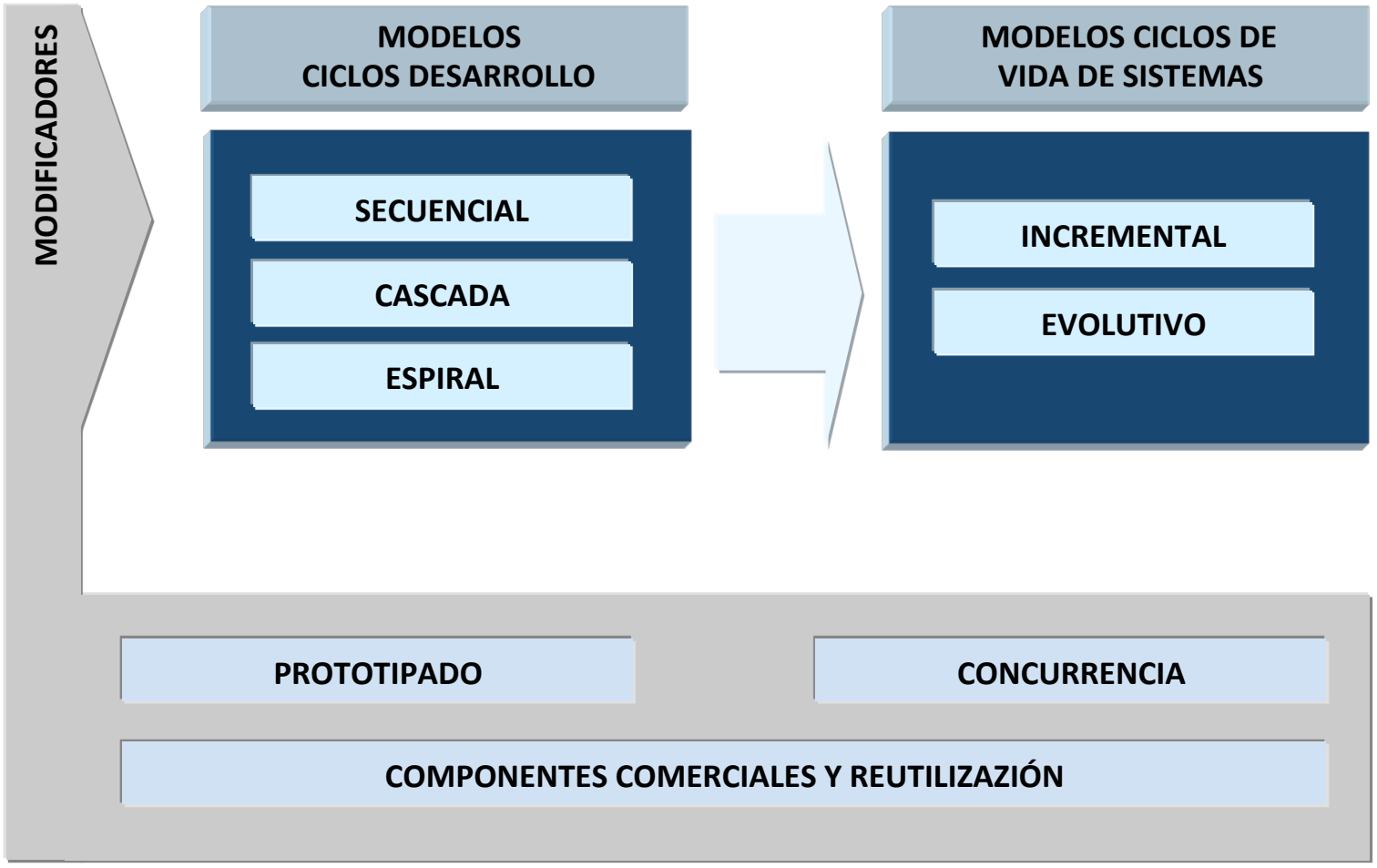


Paradigmas o modelos de desarrollo del Software

- Los conceptos básicos de partida son los definidos y normalizados en el estándar 12207:
  - **Ciclo de vida del software:** El periodo de tiempo comprendido desde la definición de los requisitos hasta el fin del su uso
  - **Procesos:** Actividades y tareas implicadas en el desarrollo, operación y mantenimiento de un sistema de software
- La aplicación de los procesos, tanto en el desarrollo como en el posterior mantenimiento y operación del software, se dibuja a través de unos "patrones fijos" que configuran el esquema de mapa de situación, relación y continuidad entre los diferentes procesos, actividades y tareas
- En la etapa de desarrollo los patrones básicos son:
  - Desarrollo en **cascada** (o variante secuencial)
  - Desarrollo en **espiral**
- Una vez desarrollada la primera versión, el ciclo de vida del sistema discurre en cada momento según uno de los siguientes patrones:
  - Desarrollo **incremental** del sistema
  - Desarrollo **evolutivo** del sistema
- Sobre estos patrones básicos, en las diferentes etapas del ciclo de vida pueden intervenir como modificadores los siguientes factores:
  - **Prototipado**
  - **Concurrencia**
  - **Componentes comerciales y reutilización**

  generando la riqueza de modelos y sub-modelos de patrones que algunos textos clasifican de forma lineal y agrupada como "modelos de ciclos de vida"

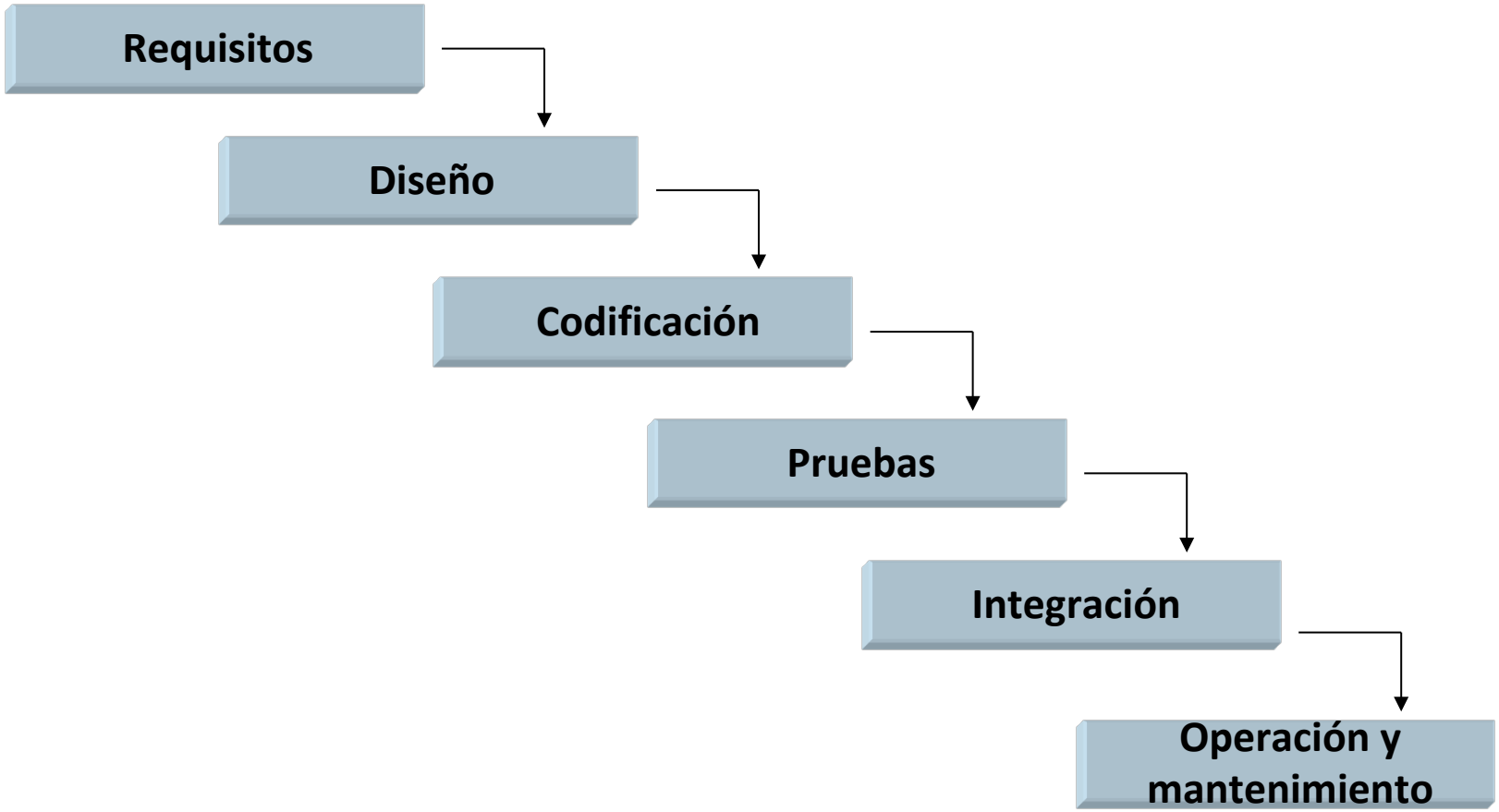
Paradigmas o modelos de desarrollo del Software



Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

Lineal o secuencial



Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

Lineal o secuencial

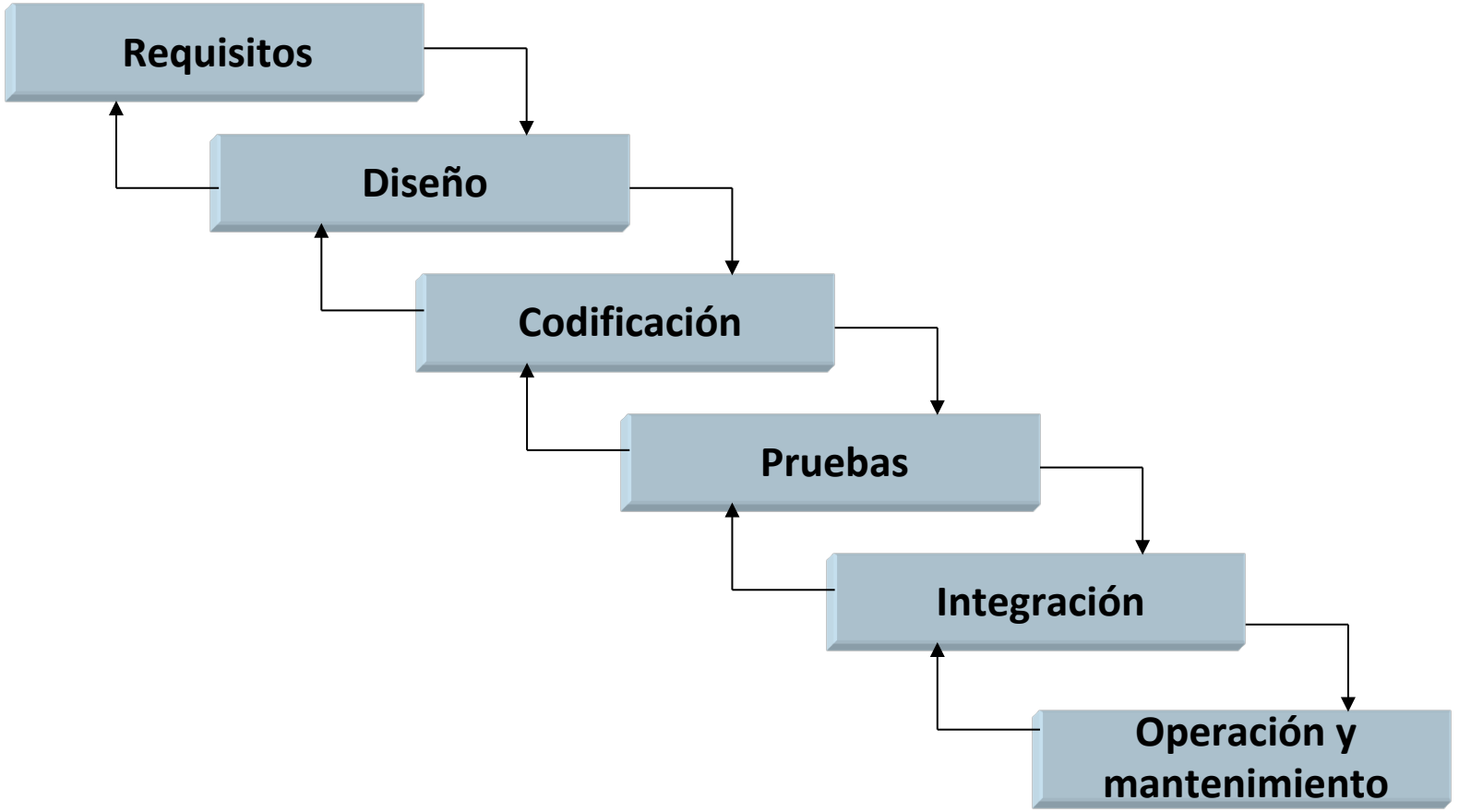
- Este modelo refleja un desarrollo marcado por la sucesión escalonada de las etapas que lo componen : requisitos, diseño, codificación, pruebas e integración
- Es necesario terminar por completo cada etapa para pasar a la siguiente
- Este modelo, identificado ya a principios de la década de los 50, resulta muy rígido porque cada fase requiere como elemento de entrada el resultado completo de la anterior
- Al aplicarlo en situaciones reales su rigidez genera problemas, porque muchas veces resulta difícil poder disponer de requisitos completos o del diseño pormenorizado del sistema en las fases iniciales, creando una barrera que impide avanzar
- Resulta apropiado para:
  - Desarrollar nuevas versiones de sistemas ya veteranos en los que el desconocimiento de las necesidades de los usuarios, o del entorno de operación no plantea riesgos
  - Sistemas pequeños, sin previsión de evolución a corto plazo

El modelo prácticamente idéntico, que evita esta rigidez es el de cascada, que se expone a continuación

Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

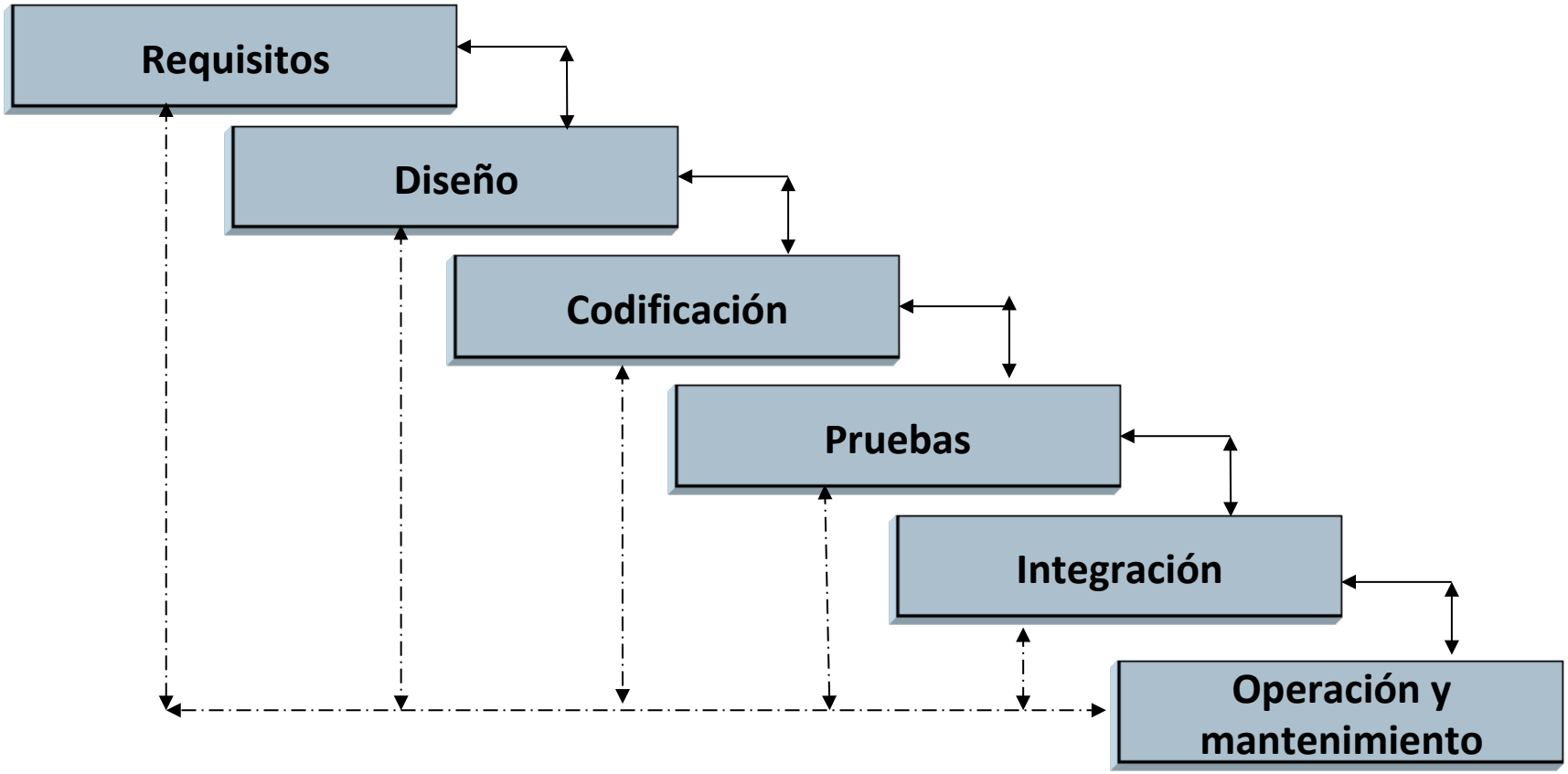
Cascada



Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

Cascada



Fuente: Standish Group Survey,

Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

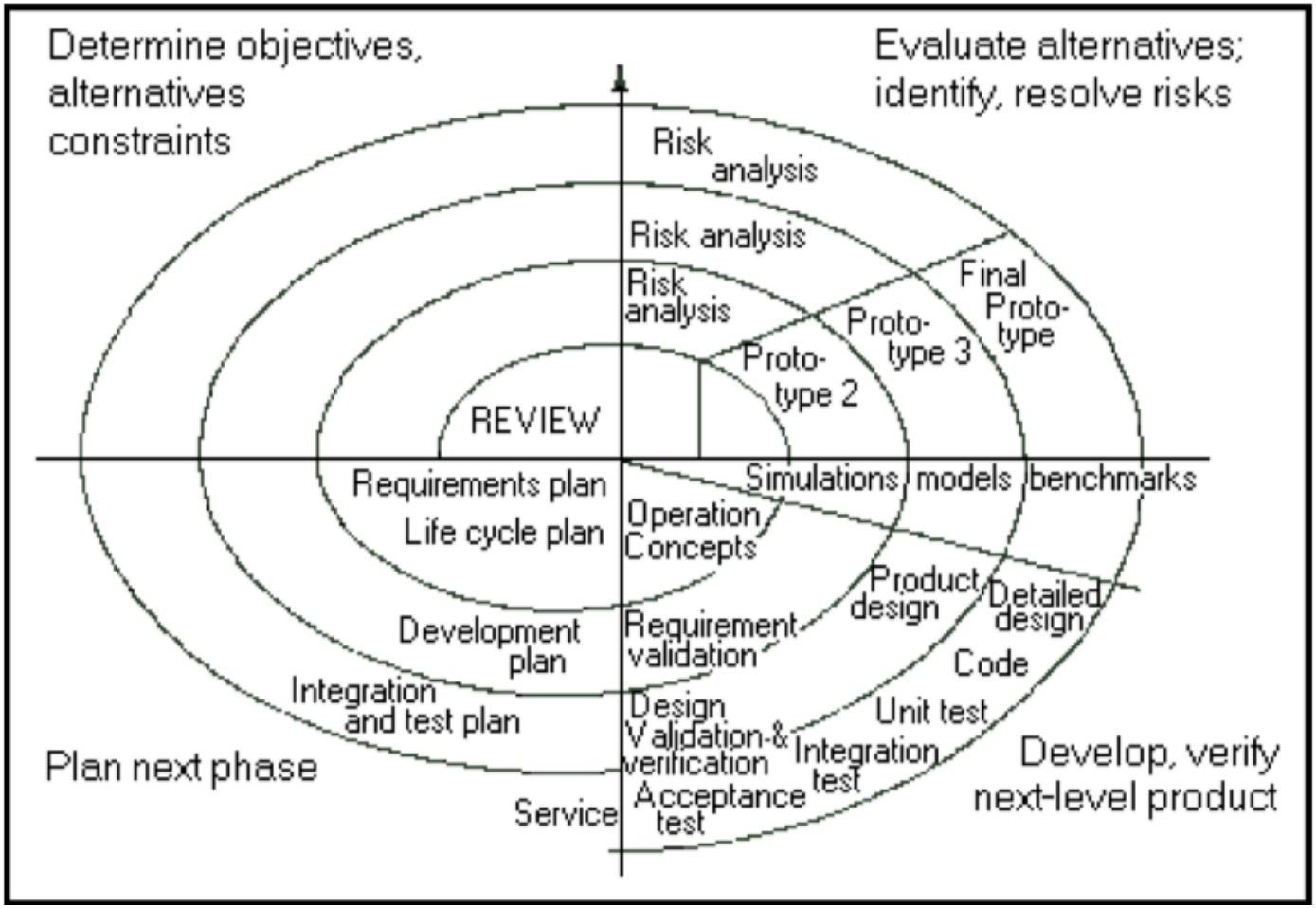
Cascada

- En 1970 Winston Royce definió flujos de retorno sobre el modelo secuencial, acuñando así el modelo en cascada
- El modelo en cascada refleja la necesidad impuesta por la realidad de retornar con frecuencia desde una fase hacia las anteriores con la información generada al avanzar el desarrollo
- Las representaciones más habituales de este modelo son las representadas en las dos figuras anteriores. La primera parece indicar que el retorno posible se da solamente entre una fase y la anterior, mientras que en la segunda se refleja mejor el hecho de que en cualquier fase puede surgir un retorno para modificar cualquiera de las anteriores.
  Este modelo, como el anterior, reconoce la importancia de disponer de unos requisitos y un diseño previo antes de comenzar con la codificación del sistema, pero al mismo tiempo se enfrenta al hecho de que en la realidad la dificultad que supone disponer de documentación elaborada de requisitos y diseño antes de empezar a codificar puede actuar como una barrera que bloquee el comienzo de la siguiente fase
- Por estas razones el modelo no se ha hecho muy popular, y los equipos que lo aplican pueden caer en la tentación de comenzar con el diseño o incluso con la codificación, sin tener un conocimiento suficiente de los requisitos
- Resulta apropiado para:
  - Desarrollar nuevas versiones de sistemas ya veteranos en los que el desconocimiento de las necesidades de los usuarios, o del entorno de operación no plantean riesgos
  - Sistemas pequeños, sin previsión de evolución a corto plazo
- Algunos textos llaman "cascada" al modelo lineal, y "cascada modificada" al modelo de cascada

Paradigmas o modelos de desarrollo del Software

Espiral

Modelos de ciclos de desarrollo



Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

Espiral

- Este modelo, definido por Boehm en 1988, presenta un desarrollo evolutivo, en contraste a la linealidad de los anteriores. También introduce como elemento distintivo la actividad de "análisis de riego" para guiar la evolución del proceso de desarrollo
- El ciclo de iteración de este modelo evolutivo se convierte en una espiral, que al representarse sobre ejes cartesianos muestra en cada cuadrante una clase particular de actividad: Planificación, Análisis de riesgo, Ingeniería y Evaluación, que se suceden de forma consecutiva a lo largo del ciclo de vida del desarrollo. La dimensión angular representa el avance relativo en el desarrollo de las actividades de cada cuadrante. En cada ciclo de la espiral se realiza una parte del desarrollo total, a través de los cuatro tipos de actividades
- En la planificación de cada vuelta se establece el contexto del desarrollo y se decide qué parte del mismo se abordará en el ciclo siguiente
- Las actividades de análisis de riesgo evalúan las alternativas posibles para la ejecución de la siguiente parte del desarrollo, seleccionando la más ventajosa y previendo los riesgos posibles
- Las actividades de ingeniería corresponden a las indicadas en los modelos lineales (secuencial y cascada): análisis, diseño, codificación, etc.

Paradigmas o modelos de desarrollo del Software

Espiral (cont.)

- Las actividades de evaluación analizan los resultados de la fase de ingeniería, tomando el resultado de la evaluación como punto de partida para el análisis de la siguiente fase
- Este modelo permite múltiples combinaciones ya que en la planificación de cada ciclo se determina el avance que se va a ejecutar durante la vuelta. Éste puede consistir en la obtención y validación de requisitos, o en el desarrollo del diseño, o el diseño junto con la codificación, o en la obtención de un subsistema completo (cascada de requisitos – diseño – codificación – pruebas – integración)
- En función de las combinaciones empleadas se podría argumentar que un desarrollo en espiral puede acabar siendo idéntico a otro modelo. Así por ejemplo si cada vuelta realizase exactamente una de las fases del modelo en cascada, al final se podría argumentar que se ha seguido una cascada. Si por el contrario en cada vuelta se desarrollara una parte del sistema global, se podría decir que se ha seguido no un modelo de ciclo de desarrollo, sino de ciclo de vida, y concretamente el modelo incremental
- Aunque a primera vista puede parecer cierto, en realidad no lo es. Si al comenzar el desarrollo se tiene decidido que se van a abordar las fases de una cascada de forma secuencial, indudablemente se va a seguir un modelo en cascada. Si se determina ir elaborando partes del sistema, se opta por un ciclo de vida incremental. Si sólo se determina dar un pequeño paso, y después de conseguido, evaluar el resultado y planificar el siguiente paso, y antes de cada ejecución se analizan los riesgos, en ese caso, el modelo seguido es un modelo en espiral

Paradigmas o modelos de desarrollo del Software

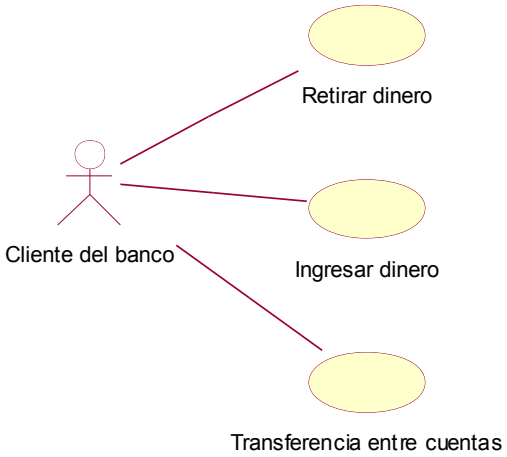
Proceso Unificado de Desarrollo

- Propuesto por los autores de UML (*lenguaje unificado de desarrollo)*
- Basado en *componentes* interconectados a través de *interfaces*
- Utiliza UML para desarrollar los esquemas y diagramas de un sistema software
- Principales aspectos definitorios
  - Dirigido por casos de uso
  - Centrado en la arquitectura
  - Iterativo e incremental

Paradigmas o modelos de desarrollo del Software

Proceso Unificado de Desarrollo (cont.)

- Utilidad de los Casos de Uso
  - Herramienta para especificar los requisitos de un sistema: representan los requisitos funcionales y juntos constituyen el ***modelo de casos de uso***, que describe la funcionalidad total del sistema
  - Guían el proceso de desarrollo (diseño, implementación y prueba)
    - ✓ basándose en el modelo de casos de uso, se crean modelos de diseño e implementación
    - ✓ se revisa cada modelo para que sean conformes al modelo de casos de uso
    - ✓ se prueba la implementación para garantizar que los componentes del modelo de implementación implementan correctamente los casos de uso
  - No sólo inician el proceso de desarrollo sino que éste sigue un hilo de trabajo que parte de los casos de uso



Paradigmas o modelos de desarrollo del Software

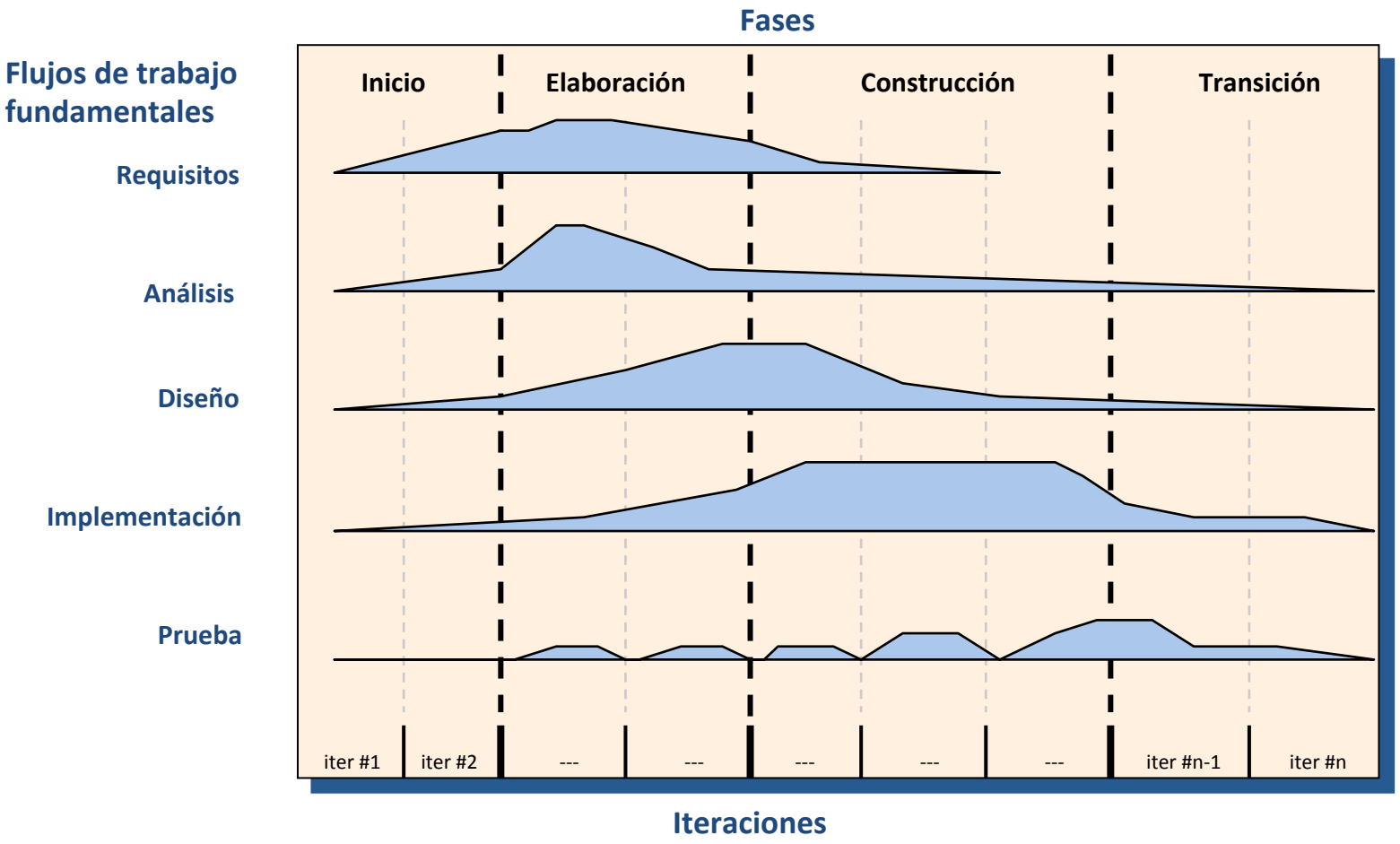
Proceso Unificado de Desarrollo (cont.)

- El proceso unificado se repite a lo largo de una serie de ciclos
  - Cada ciclo concluye con una *versión* del producto y consta de cuatro fases
    - ✓ Inicio: descripción del producto final a partir de una idea inicial y análisis de negocio para el producto
      - Principales funciones del sistema y usuarios más importantes (modelo de casos de uso)
      - Posible arquitectura del sistema
      - Plan del proyecto, coste, identificación y priorización de riesgos
    - ✓ Elaboración:
      - Se especifican en detalle los principales casos de uso
      - Se diseña la arquitectura del sistema: vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y modelo de despliegue
      - Al final se pueden planificar las actividades y estimar recursos necesarios para finalizar el proyecto
    - ✓ Construcción:
      - Se crea el producto añadiendo el software a la arquitectura
      - Al final se dispone de todos los casos de uso acordados para el desarrollo aunque puede incorporar defectos
    - ✓ Transición
      - Periodo durante el cual el producto se convierte en versión beta, en la que usuarios prueban el producto e informan de defectos y deficiencias
      - Se corrigen problemas e incorporan sugerencias
      - Incluye actividades como la formación del usuario, proporcionar una línea de ayuda y asistencia,...
  - Cada fase se divide a su vez en iteraciones

Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de desarrollo

Proceso Unificado de Desarrollo (cont.)

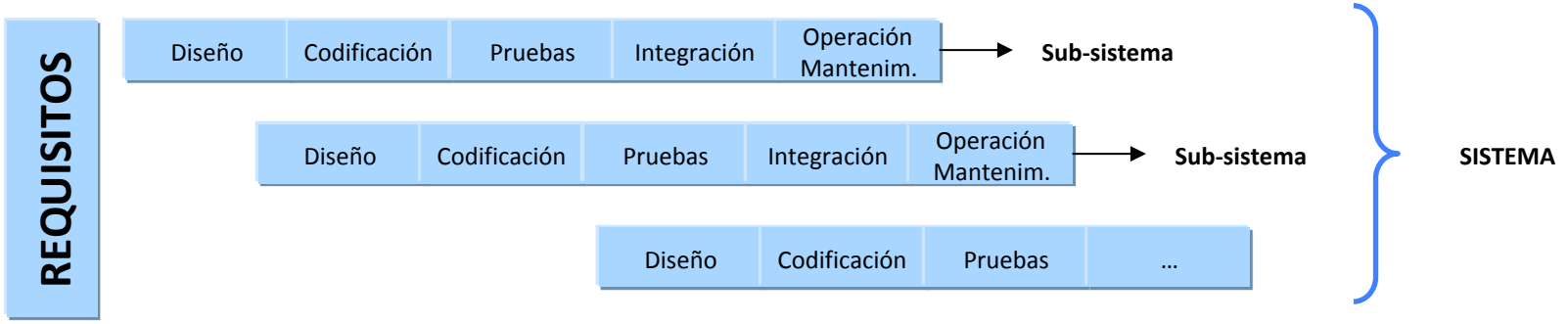


Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de evolución

Incremental

- El modelo incremental mitiga la rigidez del modelo en cascada, descomponiendo el desarrollo de un sistema en partes; para cada una de las cuales se aplica un ciclo de desarrollo (en cascada en la representación gráfica siguiente)
- Las ventajas que ofrece son:
  - El usuario dispone de pequeños subsistemas operativos que ayudan a perfilar mejor las necesidades reales del sistema en su conjunto
  - El modelo produce entregas parciales en periodos cortos de tiempo, comparados con el tiempo necesario para la construcción del sistema en su conjunto, y permite la incorporación de nuevos requisitos que pueden no estar disponibles o no ser conocidos al iniciar el desarrollo



Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de evolución

Incremental (cont.)

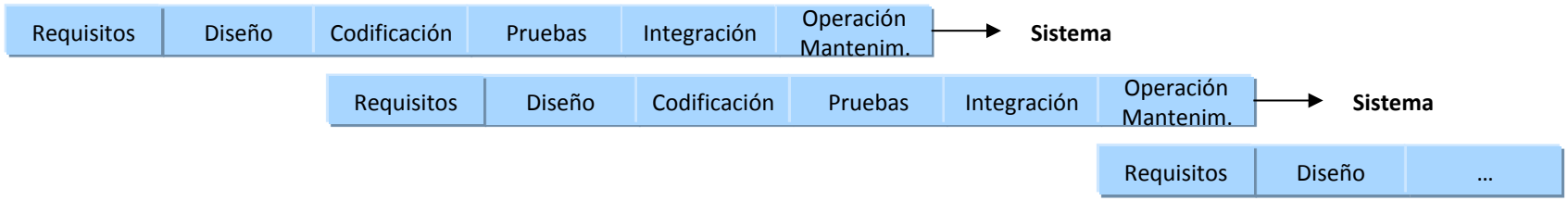
- Aunque en la representación gráfica de la figura anterior, los desarrollos de cada subsistema se solapan en el tiempo, en su aplicación real, el segundo y siguientes subsistemas pueden comenzar una vez concluido el anterior
- Resulta apropiado:
  - Desarrollo de sistemas en los que el cliente necesita disponer de parte de la funcionalidad antes de lo que costaría desarrollar el sistema completo
  - Desarrollo de sistemas en los que por razones del contexto interesa realizar la obtención de los requisitos de forma escalonada a través de subsistemas

Paradigmas o modelos de desarrollo del Software

Modelos de ciclos de evolución

Evolutivo

- Este modelo está compuesto por varios ciclos de desarrollo. Cada uno de ellos produce un sistema completo con el que se operará en el entorno de operación
- La información acumulada en el desarrollo de cada sistema, y durante su fase de operación sirve para mejorar o ampliar los requisitos y el diseño del siguiente
- En realidad es un ciclo de vida común a todos los sistemas desarrollados que se mejoran a través de versiones sucesivas



- Las circunstancias en las que este modelo puede resultar apropiado son:
  - Desconocimiento inicial de todas las necesidades operativas que serán precisas, generalmente por tratarse del desarrollo de un sistema que operará en un entorno nuevo sin experiencia previa
  - Necesidad de que el sistema entre en operación en tiempos inferiores a los que serían necesarios para diseñarlo y elaborarlo de forma exhaustiva
  - Necesidad de desarrollar sistemas en entornos cambiantes (sujetos a normas legislativas, mejora continua del producto para hacer frente a desarrollos de la competencia, etc.)
- Aunque en su concepción inicial contempla desarrollos internos en cascada, también podría plantearse, por ejemplo, un ciclo de vida evolutivo con desarrollos internos en espiral

Paradigmas o modelos de desarrollo del Software

Modificadores de los modelos

Prototipado

- El prototipado consiste en la construcción de modelos de prueba, que simulen el funcionamiento que se pretende conseguir en el sistema
- Los prototipos pueden ser:
  - Ligeros: dibujos de pantallas de interfaz con simulación de funcionamiento por enlaces a otros dibujos…
  - Operativos: Módulos de software con funcionamiento propio que se desarrollan sin cubrir las funcionalidades completas del sistema, normalmente en entornos RAD (rapid application development)

Esta forma de trabajo previo suele tener como principal objetivo la experimentación con un entorno similar al pretendido, para obtener retro-información del usuario o cliente que ayuda a los desarrolladores en la concreción de los requisitos.

Aunque ofrece muchas ventajas, deben conocerse los riesgos que implica el uso de prototipado:

- Como puede parecer que se ha desarrollado un interfaz de usuario sofisticado y elaborado, el cliente puede llegar a pensar que ya se ha realizado el grueso del trabajo
- Si se trata de un prototipo operativo, puede empezar a crecer al margen de la planificación, más allá de los objetivos previstos, desbordando agendas y recursos
- Si se trata de un prototipo ligero desarrollado fuera del departamento de desarrollo (ej. Marketing), puede mostrar al cliente funcionalidades no implementables
- El prototipo puede llegar a ofrecer funcionalidades superiores a lo conseguible, por estar construido en un entorno diferente al de desarrollo, o no incluir toda la funcionalidad del sistema

Paradigmas o modelos de desarrollo del Software

Modificadores de los modelos

Concurrencia

- Consiste en el solapamiento de un proceso sobre otro
- Resulta bastante frecuente que aunque se haya planteado un desarrollo en cascada, se comience con una fase sin haber terminado por completo la anterior; y así por ejemplo quizá el equipo que ha llevado a cabo el diseño detallado de determinados módulos, quizá comienza ya su codificación, mientras otros equipos aún tienen en su planificación tareas de diseño pendientes
- La concurrencia puede aportar beneficios sobre la planificación de un proyecto de software, o por el contrario ser origen o consecuencia de problemas
- Los factores que deben tenerse en cuenta para analizar cómo ayuda o perjudica al rendimiento son:
  - Índice de concurrencia. Se produce en un grado reducido, generando un escaso flujo de modificaciones; o por el contrario se da de forma intensiva generando situaciones problemáticas en la planificación o en la distribución del trabajo
  - Gestión de la concurrencia. La concurrencia puede producirse en un proyecto de forma planificada o inducida por las circunstancias. En ambos casos resulta muy importante la labor de gestión del proyecto para tratarla de forma adecuada con el mayor beneficio, o el menor perjuicio a los planes y la calidad del proyecto

Paradigmas o modelos de desarrollo del Software

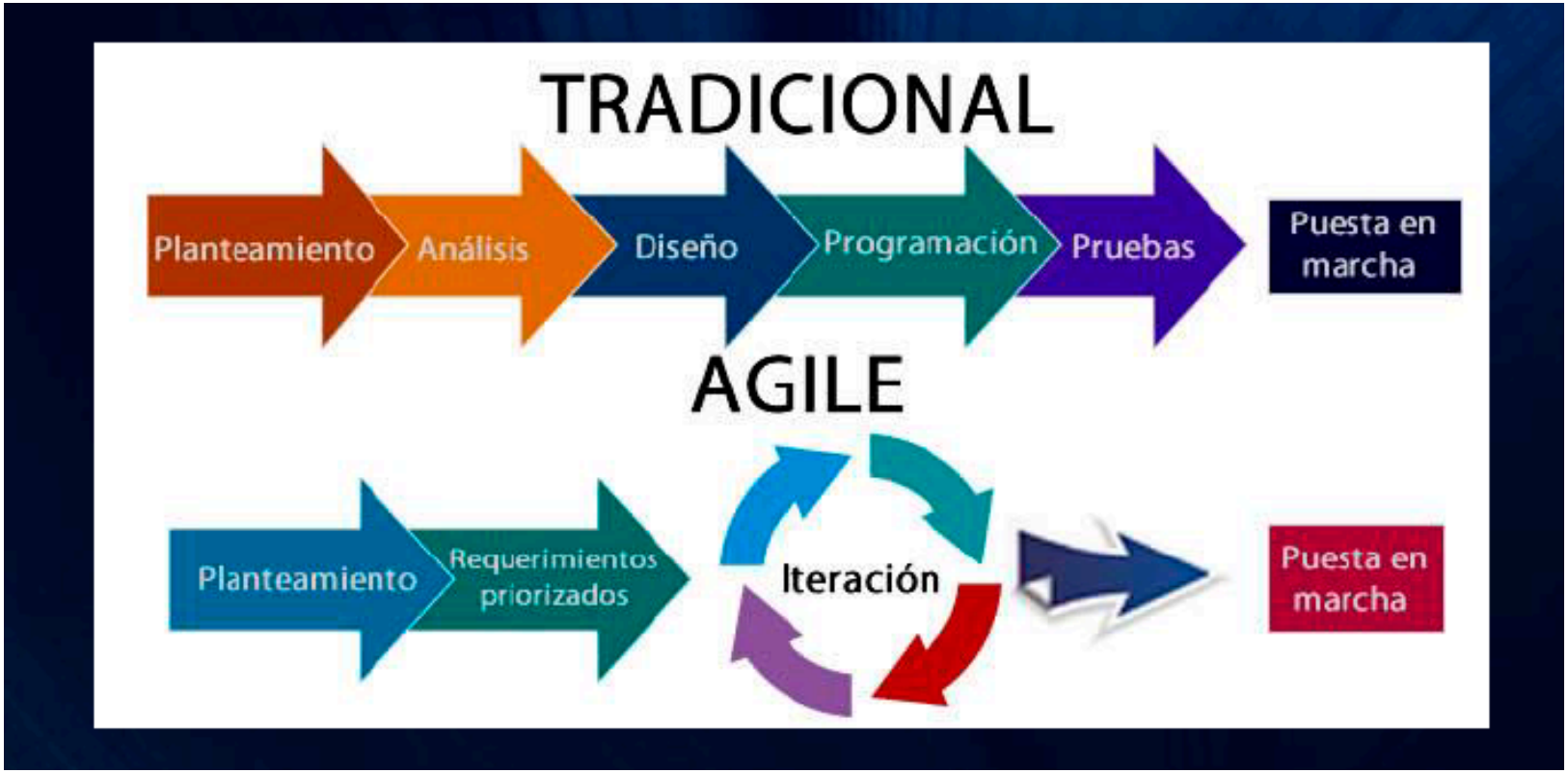
Modificadores de los modelos

Componentes comerciales y reutilización

- Resulta muy habitual integrar en el desarrollo de un sistema partes "pre-construidas": que pueden ser componentes comerciales, o la reutilización de componentes o marcos ya desarrollados para otros sistemas
- Esta tendencia surge desde tres situaciones:
  - Presión competitiva para reducir agendas y costes
  - Incremento de la complejidad y estandarización de los entornos de operación
  - Aparición de las líneas de producción en las que se desarrollan múltiples sistemas de software re-utilizando partes de diseño y componentes
- El uso de componentes o partes ya desarrolladas tienen implicaciones en el ciclo de desarrollo, diferentes según las circunstancias. Así por ejemplo, si gran parte del sistema consta de componentes ya desarrollados y probados, el periodo de pruebas se acortará sustancialmente
- Si un proyecto va a delegar funcionalidades críticas en un componente comercial, que no ha empleado previamente la organización desarrolladora, es posible que incorpore en su ciclo de desarrollo una fase de pruebas de ese componente, antes del diseño, para obtener la certeza previa de que el componente se comporta como se espera

Modelos de Procesos Ágiles

Se caracterizan por ser la negación de las características de los ciclos de vida convencionales



Modelos de Procesos Ágiles

- Se basan en el denominado "Manifiesto Ágil de Software"
- Este manifiesto surge en el año 2001 cuando 16 notables desarrolladores, escritores y consultores firman dicho manifiesto, que esta centrado en 4 valores

Individuos e Interacciones	**sobre**	**Procesos y Herramientas**
Software que funciona	**sobre**	**Documentación exhaustiva**
Colaboración con el Cliente	**sobre**	**Negociación de Contratos**
Responder ante el cambio	**sobre**	**Seguimiento de un Plan**

Modelos de Procesos Ágiles

El manifiesto Ágil está basado en 12 principios, que son:

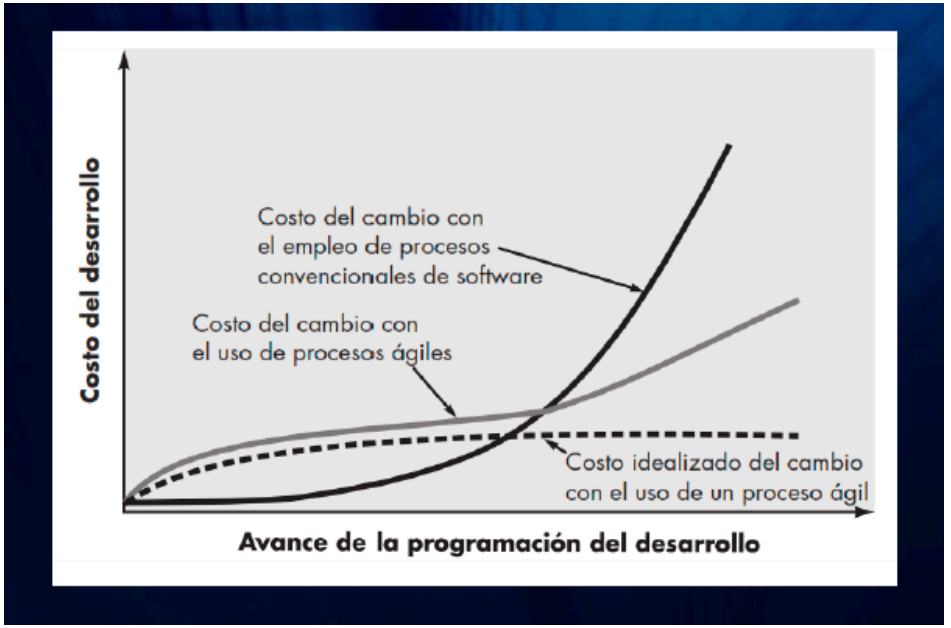
- Satisfacer al cliente a través de la entrega de valor
- Aceptamos que los requisitos cambien
- Entregamos software funcional frecuente
- La gente del negocio y los desarrolladores trabajamos juntos diariamente
- Los proyectos se hacen en entornos de individuos motivados
- Las comunicaciones cara a cara
- El software funcionando es la medida principal de progreso
- Promover un paso sostenido
- Excelencia técnica y buen diseño
- La simplicidad es esencial
- Equipos auto-organizados
- Inspeccionar y adaptar

Modelos de Procesos Ágiles

El proceso ágil es candidato a aplicarse en determinadas situaciones clave como son:

- Cuando resulta difícil predecir cuales requisitos del software persistirán y cuales cambiaran (cambios en las prioridades de los clientes)
- Es conveniente que el diseño y la construcción de software estén intercalados
- El análisis, diseño y construcción de software no son predecibles respecto a la planificación

Debido a que uno de sus elementos primordiales son los factores humanos ya que "*El desarrollo ágil se centra en los talentos y habilidades de los individuos, puesto que el proceso se ajusta a personas y a equipos específicos*"



Modelos de Procesos Ágiles

La Ingeniería del Software Ágil combina una filosofía y un conjunto de directrices de desarrollo, como son:

- Busca la satisfacción del cliente y la entrega temprana del software incremental
- Equipos de proyectos pequeños y con alta motivación
- Métodos informales
- Mismos productos de trabajo
- Simplicidad general del desarrollo

Es importante en la actualidad debido a:

- En estos momentos los sistemas basados en computadoras y los productos de software están acelerados y en un cambio continuo
- La ingeniería del software ágil representa una alternativa a la ingeniería convencional para ciertas clases de software y para ciertos tipos de proyectos

Modelos de Procesos Ágiles: Modelo de Desarrollo SCRUM

- SCRUM es un proceso ágil que nos permite centrarnos en ofrecer el mas alto valor de negocio en el menor tiempo
- Nos permite rápidamente y en repetidas ocasiones inspeccionar software real de trabajo (cada dos semanas o un mes)
- El negocio fija las prioridades. Los equipos se auto-organizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad
- Cada dos semanas o un mes, cualquiera puede ver el software real funcionando y decidir si liberarlo o seguir mejorándolo en otro sprint
- Ken Schwaber and Mike Cohn, fundaron conjuntamente la Scrum Alliance en 2002, inicialmente dentro de la Agile Alliance

Ha sido utilizado por

- Microsoft
- Yahoo
- Google
- Electronic Arts
- High Moon Studios
- Lockheed Martin
- Philips
- Siemens
- Nokia
- Capital One
- BBC
- Intuit
- Etc.

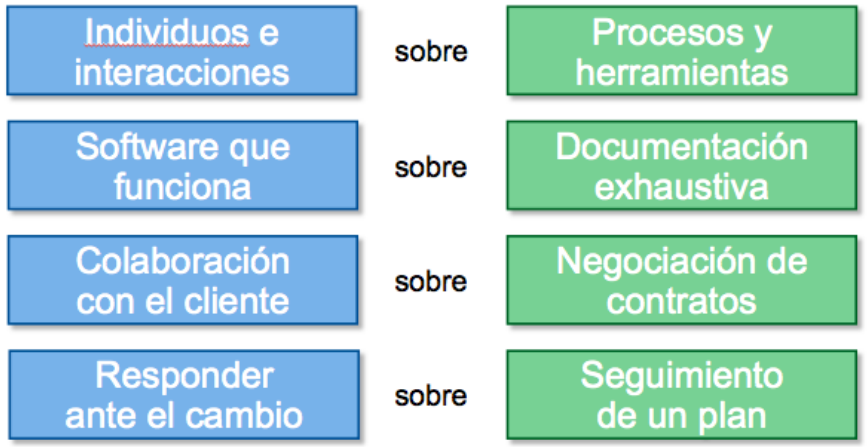
Ha sido utilizado para

- Software comercial
- Desarrollos internos
- Desarrollos bajo Contrato
- Aplicaciones Financieras
- Aplicaciones certificadas ISO 9001
- Sistemas Embebidos
- Joint Strike Fighter
- Desarrollo de video juegos
- Software de control satelital
- Sitios Web
- Teléfonos portátiles
- Algunas de las más grandes aplicaciones en uso

Modelos de Procesos Ágiles: Modelo de Desarrollo SCRUM

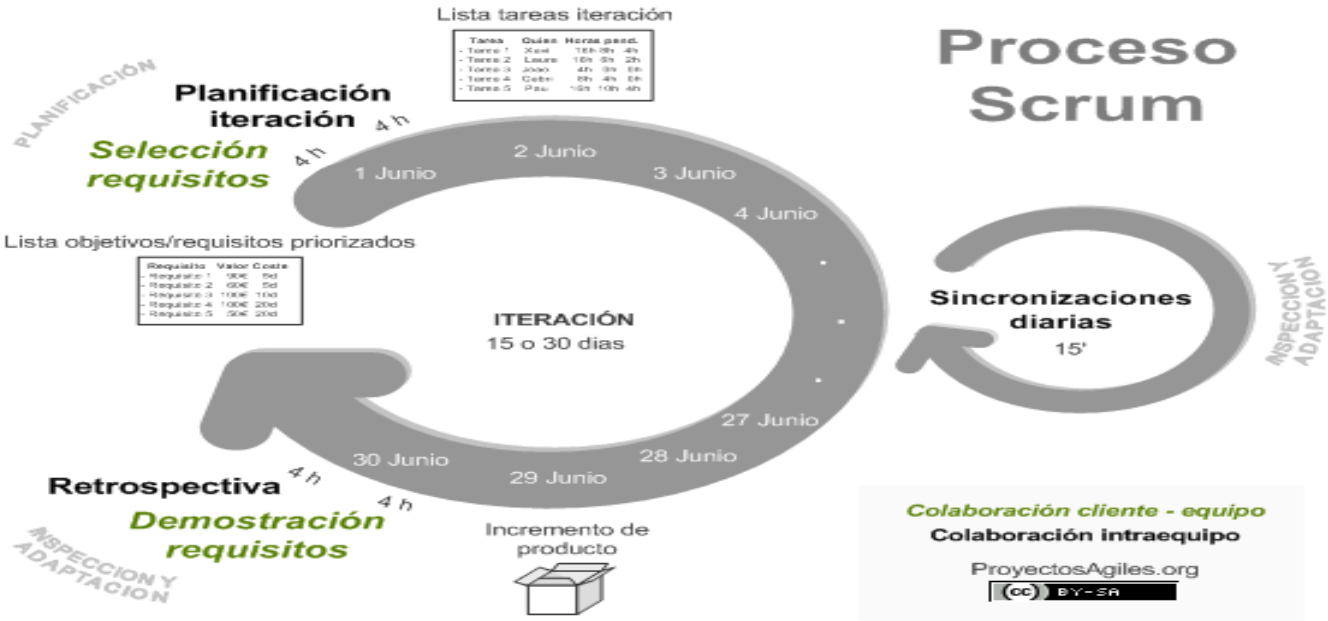
Características

- Equipos auto-organizados
- El producto avanza en una serie de "Sprints" de dos semanas a un mes de duración
- Los requisitos son capturados como elementos de una lista de "Product Backlog"
- No hay prácticas de ingeniería prescritas
- Utiliza normas generativas para crear un entorno ágil para la entrega de proyectos
- Uno de los "procesos ágiles"
- Cumple el manifiesto Ágil



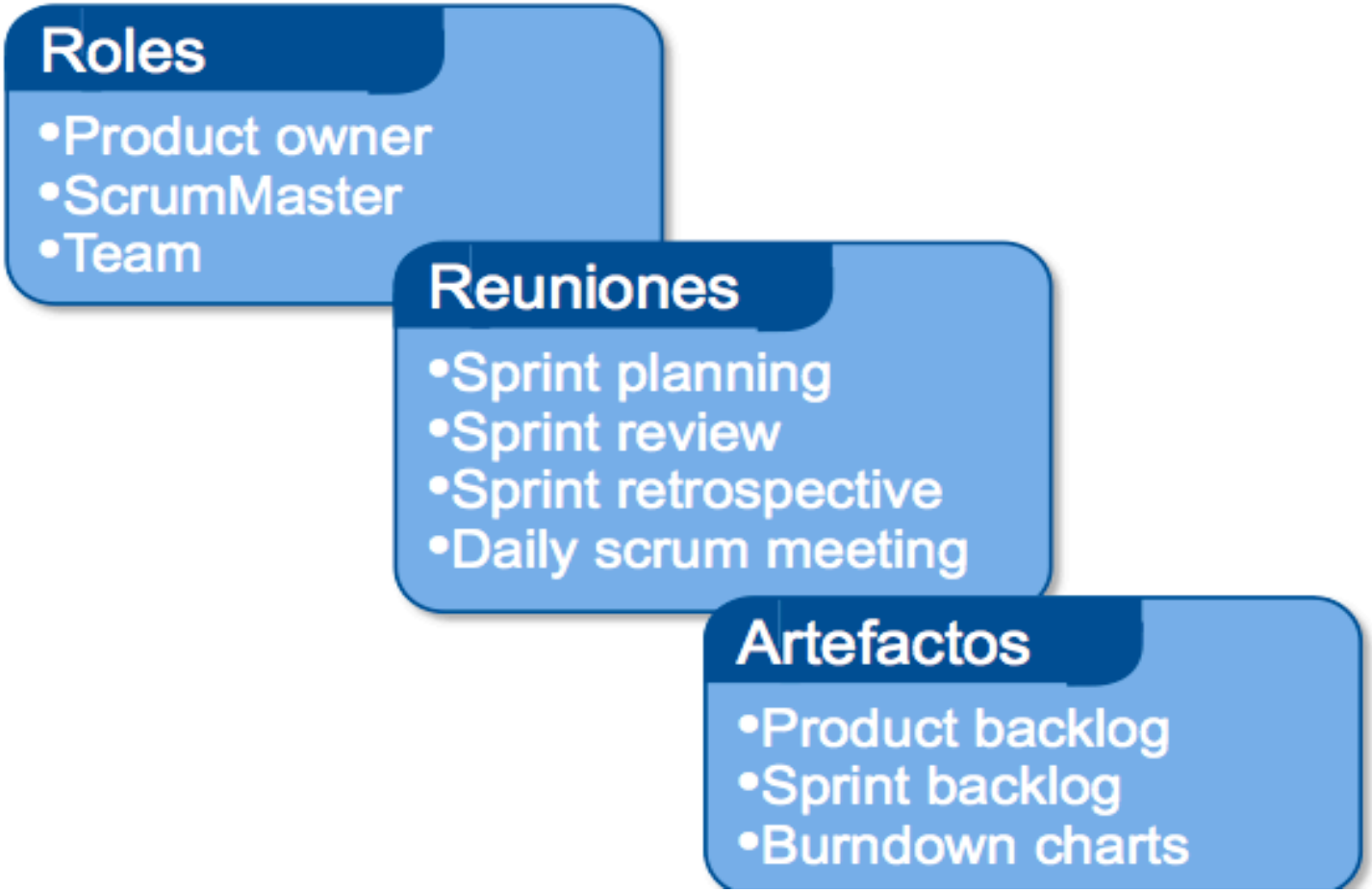
Modelos de Procesos Ágiles: Modelo de Desarrollo SCRUM

- Sprints
  - En Scrum los proyectos avanzan en una serie de "Sprints" análogo a las iteraciones en XP
  - La duración típica es 2–4 semanas o a lo sumo un mes de calendario
  - La duración constante conduce a un mejor ritmo
  - El producto es diseñado, codificado y testeado durante el Sprint



Modelos de Procesos Ágiles: Modelo de Desarrollo SCRUM

- SCRUM FrameWork



Creación del Modelo del Ciclo de Vida

- Al iniciar el proyecto, el responsable de la arquitectura de procesos debe realizar los siguientes pasos:
  - Análisis de las circunstancias ambientales del proyecto
  - Diseño del modelo específico de ciclo de vida para el proyecto (sobre las bases de los diseños más apropiados, para el desarrollo y la evolución del sistema de software)
  - Mapeo de las actividades sobre el modelo
  - Desarrollo del plan para la gestión del ciclo de vida del proyecto
- Debe considerar aspectos como:
  - Posibilidad de descomposición del sistema en subsistemas de software, con agendas y entregas diferenciadas
  - Estabilidad esperada de los requisitos
  - Novedad del proceso o procesos gestionados por el sistema en el entorno del cliente.
  - Criticidad de las agendas y presupuestos
  - Grado de complejidad del interfaz de operación, criticidad de la usabilidad
  - Grado de conocimiento y familiaridad con el entorno de desarrollo, componentes externos empleados, etc.