



DOCENCIA VIRTUAL

Finalidad:

Prestación del servicio Público de educación superior (art. 1 LOU)

Responsable:

Universitat Politècnica de València.

Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:

<http://www.upv.es/contenidos/DPD/>

Propiedad intelectual:

Uso exclusivo en el entorno de aula virtual.

Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



EL PROCESO DEL SOFTWARE

Tema 2

Ingeniería del Software

ETS Ingeniería Informática

DSIC – UPV

Objetivos

- Definir el término “Proceso del Software”
- Presentar los principales modelos de proceso de desarrollo propuestos a lo largo de la historia
- Introducir la noción de metodología

Contenidos

1. Introducción. El Proceso Software

2. Ciclos de Vida

- Clásico o en Cascada
- Clásico con Prototipado
- Programación Automática
- Incremental
- Espiral

3. Metodologías




Anexo:

RUP
Metodologías Ágiles

Bibliografía básica

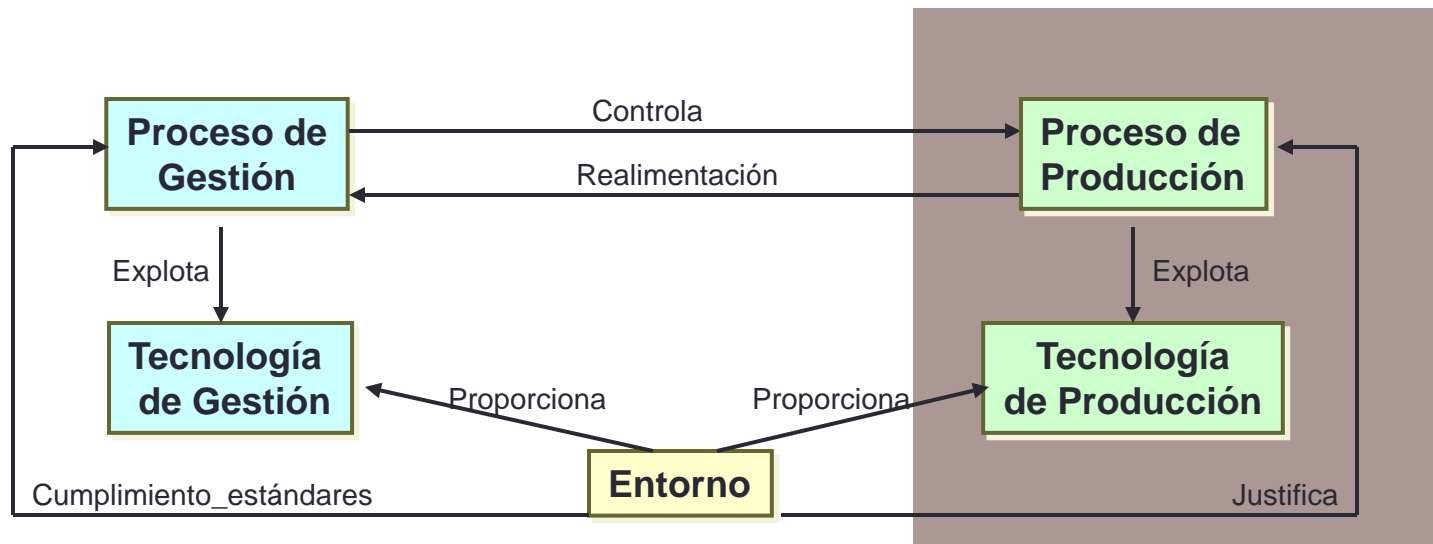
- 📖 Sommerville, I., Ingeniería del Software (9ª ed.), Addison-Wesley, 2011.
- 📖 Pressman, R., Ingeniería del Software. Un enfoque práctico (7ª ed.), McGraw-Hill, 2010.
- 📖 Royce, W.W., Managing the Development of Large Software Systems: Concepts and Techniques. Proc WESCON, 1970
- 📖 Agresti, W.W. Tutorial: New Paradigms for Software Development. IEEE Computer Society Press, 1986
- 📖 Balzer, R., Cheatman, T.E. and Green, C., Software Technology in the 1990's: Using a New Paradigm, IEEE Computer, Nov. 1983, pp. 39-45
- 📖 McDermid, J. And Rook, P., Software Development Process Models. Software Engineer's Reference Book, CRC Press, 1993
- 📖 Boehm, B.W.. A Spiral Model of Software Development and Enhancement, IEEE Computer, pages 61-72, May 1988.

Bibliografía metodologías

-  Krutchen, P., *The Rational Unified Process- An Introduction*. Addison – Wesley, 1998
-  Jacobson, G. Booch and J. Rumbaugh., *The Unified Software Development Process*, Addison-Wesley, 1999
-  Beck, K., *Extreme Programming Explained: Embrace Change*. The XP Series. Addison-Wesley, 2000

El Proceso del Software

- Establece un marco para el desarrollo de software
- En general el término “Proceso Software” se asocia al *proceso de producción....* pero incluye también el *proceso de gestión*

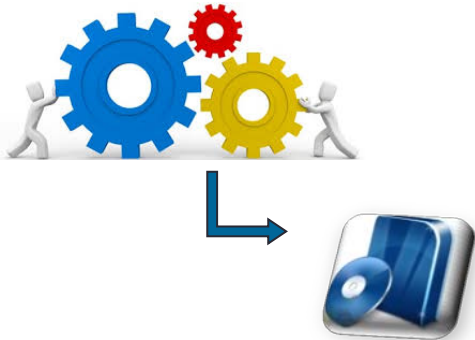


El Proceso de Desarrollo

- Conjunto de actividades cuya meta es el desarrollo o evolución de software
- Conocido también como **Ciclo de Vida**

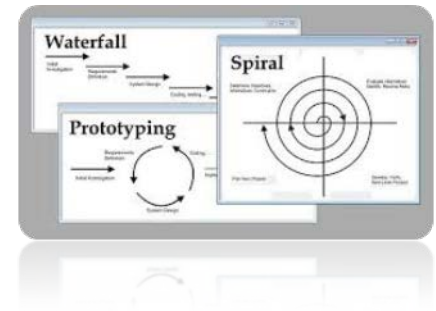
- **Actividades** genéricas que aparecen siempre:

- Especificación
- Desarrollo
- Validación
- Evolución

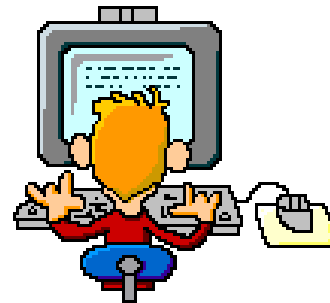
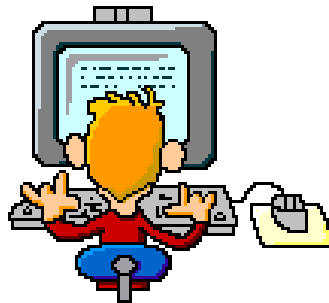
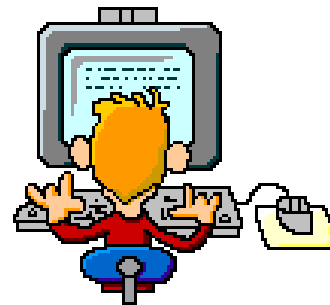
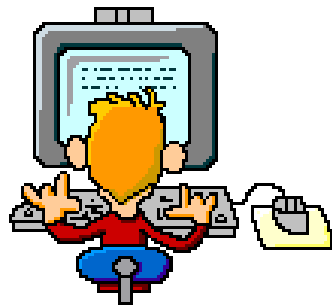
- 
- The illustration shows two white 3D figures interacting with three interlocking gears: a large blue one, a medium yellow one, and a small red one. A blue arrow points from the gears down to a blue CD icon. A red diagonal line is drawn across the slide, separating the generic activities on the left from the specific ones on the right.
- Análisis
 - Diseño
 - Implementación
 - Pruebas
 - Mantenimiento

Modelos de Ciclo de Vida

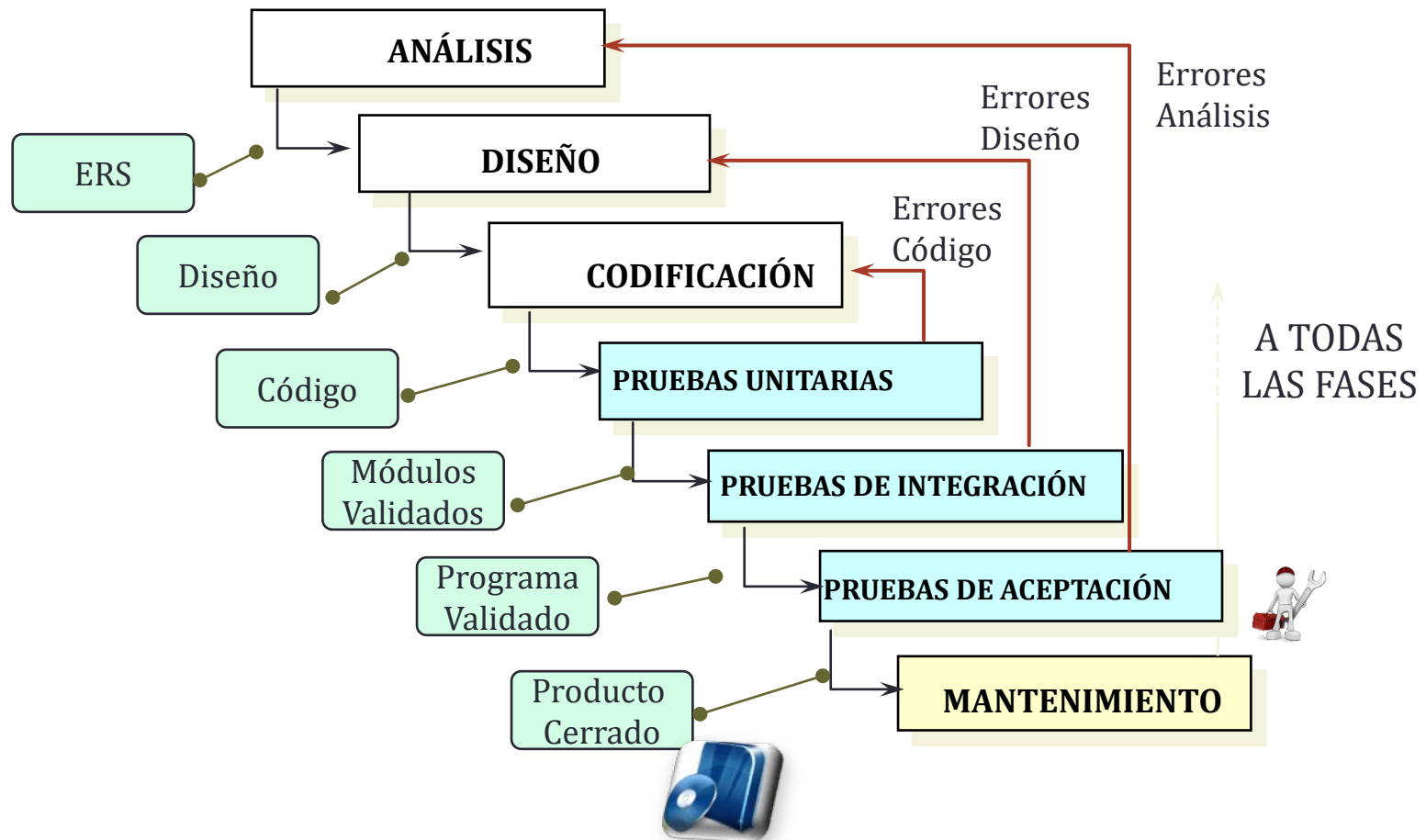
- Codificar y corregir (*code-and-fix*)
- Clásico o en cascada
- Clásico con prototipado
- Programación Automática
- Modelos evolutivos:
 - Incremental
 - En espiral



Codificar y Corregir

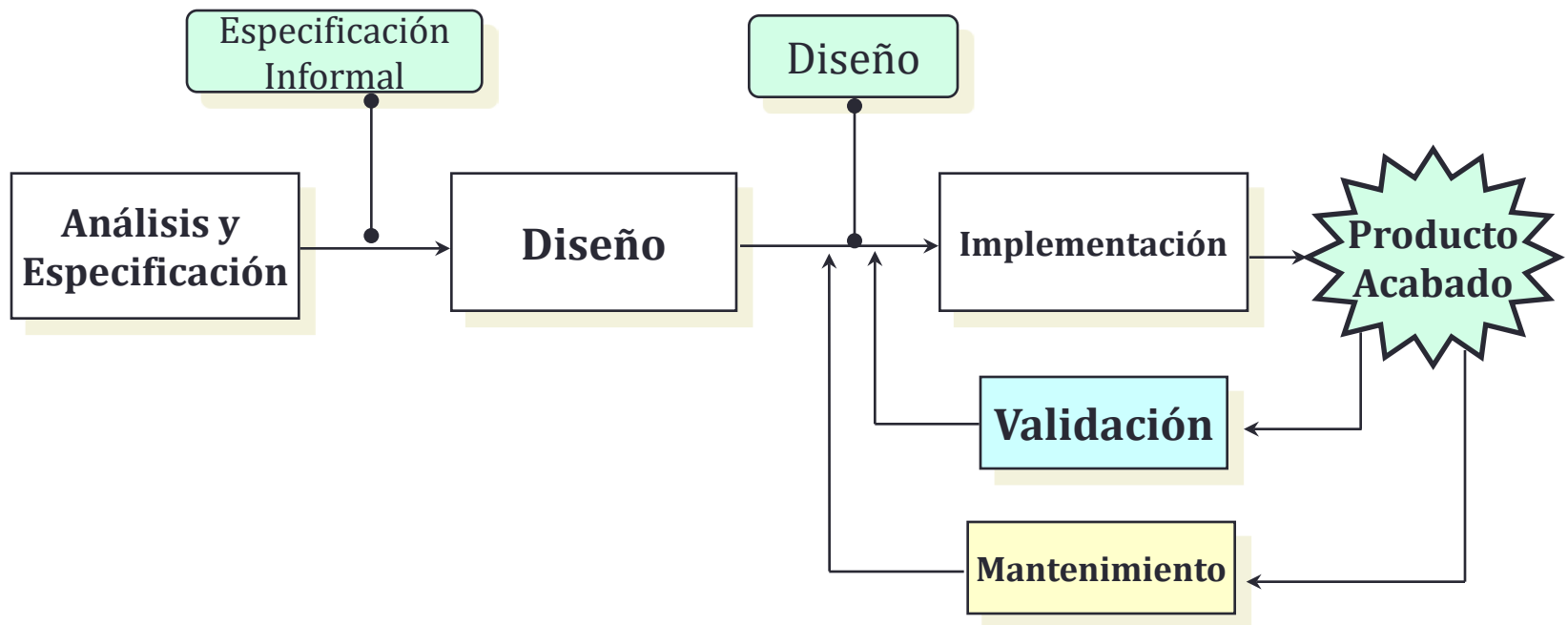


Modelo Clásico o en Cascada



Modelo Clásico o en Cascada

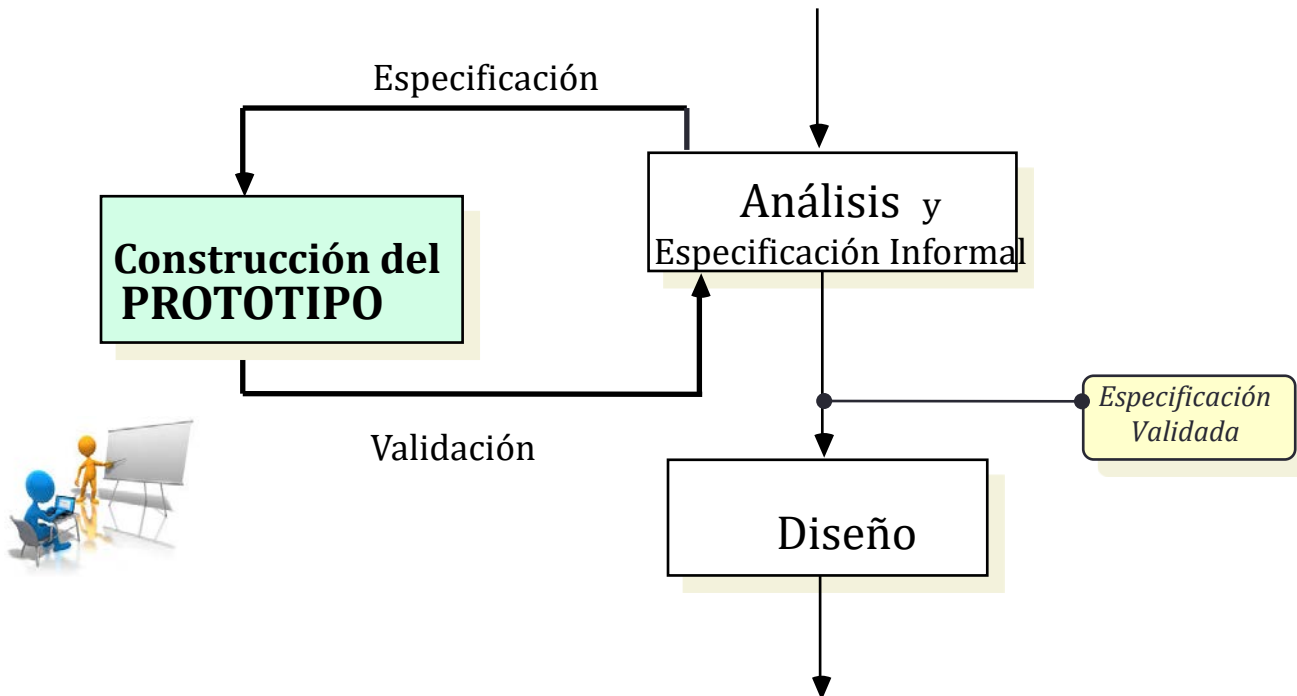
- En la práctica el modelo tiende a *deformarse*, y todo el peso de la validación y mantenimiento recae, en su mayor parte, sobre el *código fuente*.



Modelo Clásico con Prototipado

- **Prototipo:** Primera versión de un producto, en el que se han incorporado sólo algunas características del sistema final, o no se ha realizado completamente
- **Clases de Prototipos:**
 - Vertical: desarrolla completamente alguna de las facetas del programa.
 - Horizontal: desarrolla en parte todas las facetas

Modelo Clásico con Prototipado



- Ayuda a los clientes a establecer claramente los requisitos
- Ayuda a los desarrolladores a mejorar sus productos

Modelo Clásico con Prototipado

- Crítica:

- ☺ Reduce el riesgo de parcheado sobre el producto final (Aunque no elimina el mantenimiento sobre el código)
- ☺ Ayuda a entender los requisitos tanto a clientes como a desarrolladores
- ☹ El cliente ve una versión de lo que será el programa final, sin asumir que no es robusta ni completa
- ☹ Supone una inversión adicional que puede no ser rentable; además, el tiempo invertido puede hacer que el producto pierda oportunidad
- ☹ Es frecuente arrastrar malas decisiones que sólo eran apropiadas para la obtención rápida del prototipo

Modelo de Programación Automática

(R. Balzer, 1983)

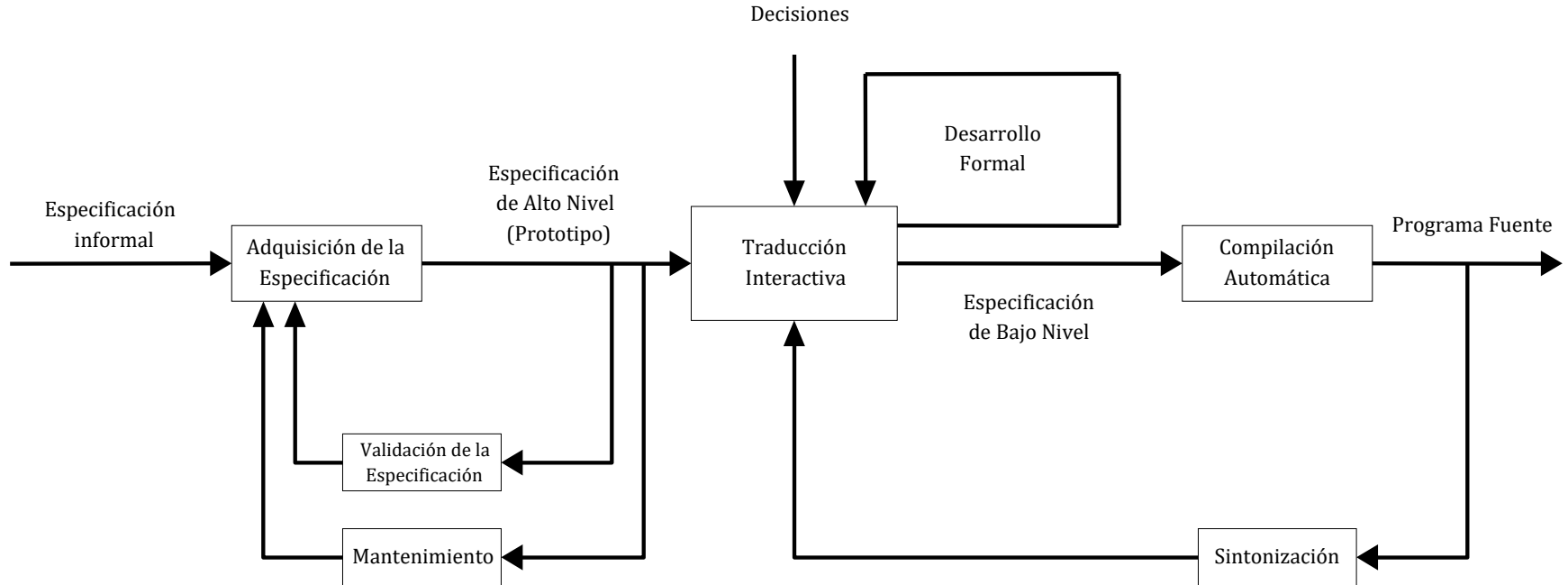
- Objetivo

Introducir la automatización en el proceso de construcción del software

- Características básicas:

- ✓ Uso de lenguajes formales de especificación
- ✓ La especificación es un prototipo del producto
- ✓ Los requisitos se perfilan animando la especificación
- ✓ El programa se deriva (semi)automáticamente

Modelo de Programación Automática



Modelo de Programación Automática

- **Comparación**

CLÁSICO Prototipado

- Especificación informal
- Prototipado no usual
- El prototipo se crea manualmente
- El prototipo se desecha
- Implementación manual
- El código ha de probarse
- Mantenimiento sobre el código

P. AUTOMÁTICA

- Especificación formal
- Prototipado estándar
- La especificación es el prototipo
- Evoluciona hacia el producto final
- Implementación automática
- Sin pruebas
- Mantenimiento sobre la especificación

Modelo de Programación Automática

- Crítica

- ☺ Ayuda a reducir errores humanos

- ☺ Reduce el coste de desarrollo

- ☹ Dificultad de uso de los lenguajes formales

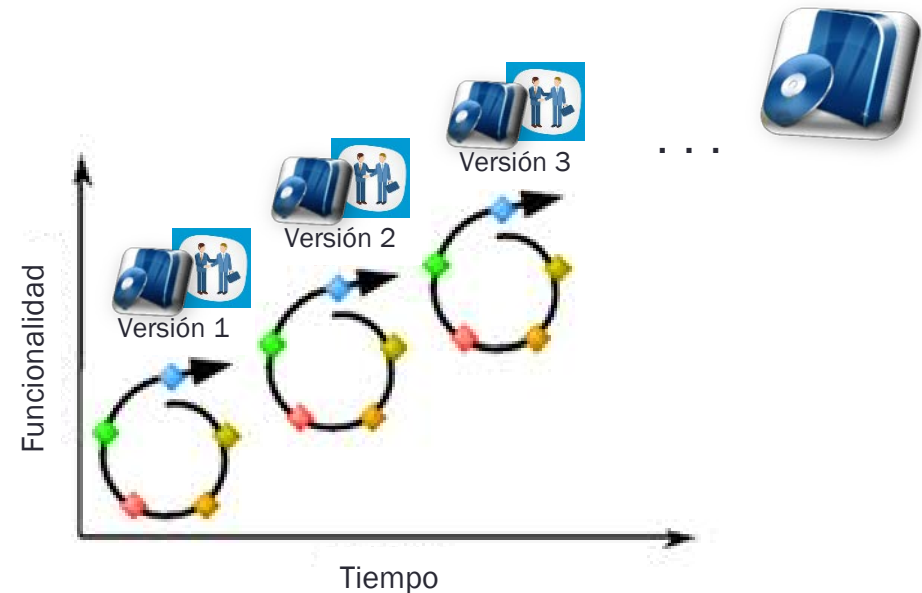
➡ *Es el antecesor de MDE/MDA*

Desarrollo Evolutivo

- Adaptable a requisitos cambiantes
- Se elaboran versiones cada vez más completas del software

➔ Modelo incremental

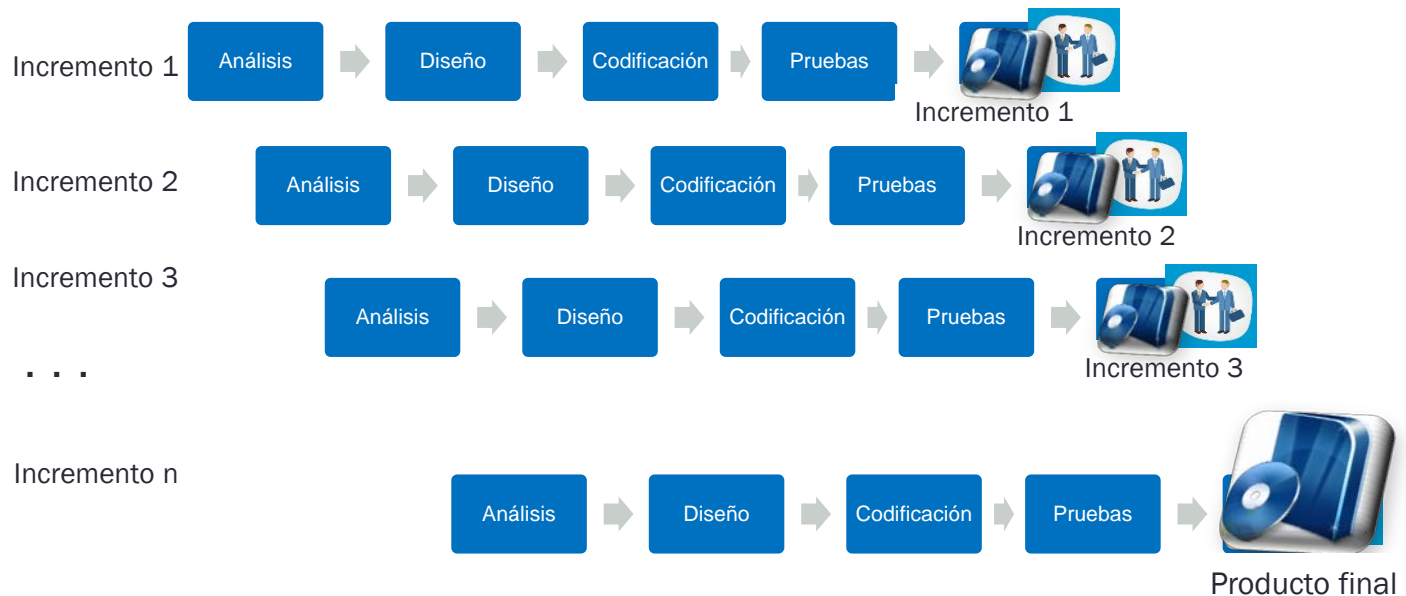
➔ Modelo en espiral



Modelo Incremental

(McDermind, 1983)

- Secuencia de aplicaciones del ciclo clásico
- Cada iteración produce un incremento del producto
- Finaliza cuando se entrega el producto final



Modelo Incremental

- Crítica

- 😊 Útil cuando no se dispone de personal para una implementación completa
- 😊 Cada entrega puede ser evaluada por el usuario ➔ alta interactividad
- 😞 Dificultad de determinar el incremento requerido en cada iteración

Modelo En Espiral

(B. Boehm, 1988)

- Enfoque:
 - Iterativo.
 - Interactivo.
 - Evolutivo
- Introduce el análisis de riesgos en el proceso de desarrollo

Modelo En Espiral



Planificación

Recopilación de requisitos
y planificación inicial

Planificación basada en
los comentarios del cliente

Evaluación
del cliente



Evaluación del cliente

Análisis de riesgo



Análisis de riesgo basado
en los requerimientos iniciales

Análisis de riesgo basado
en la reacción del cliente

**Decisión de
seguir o no**

Hacia el sistema final

Versión inicial del software

Versión del siguiente nivel

Sistema de Ingeniería

Ingeniería



Modelo En Espiral

- Crítica

- 😊 Cada vez se obtienen versiones más completas del producto.

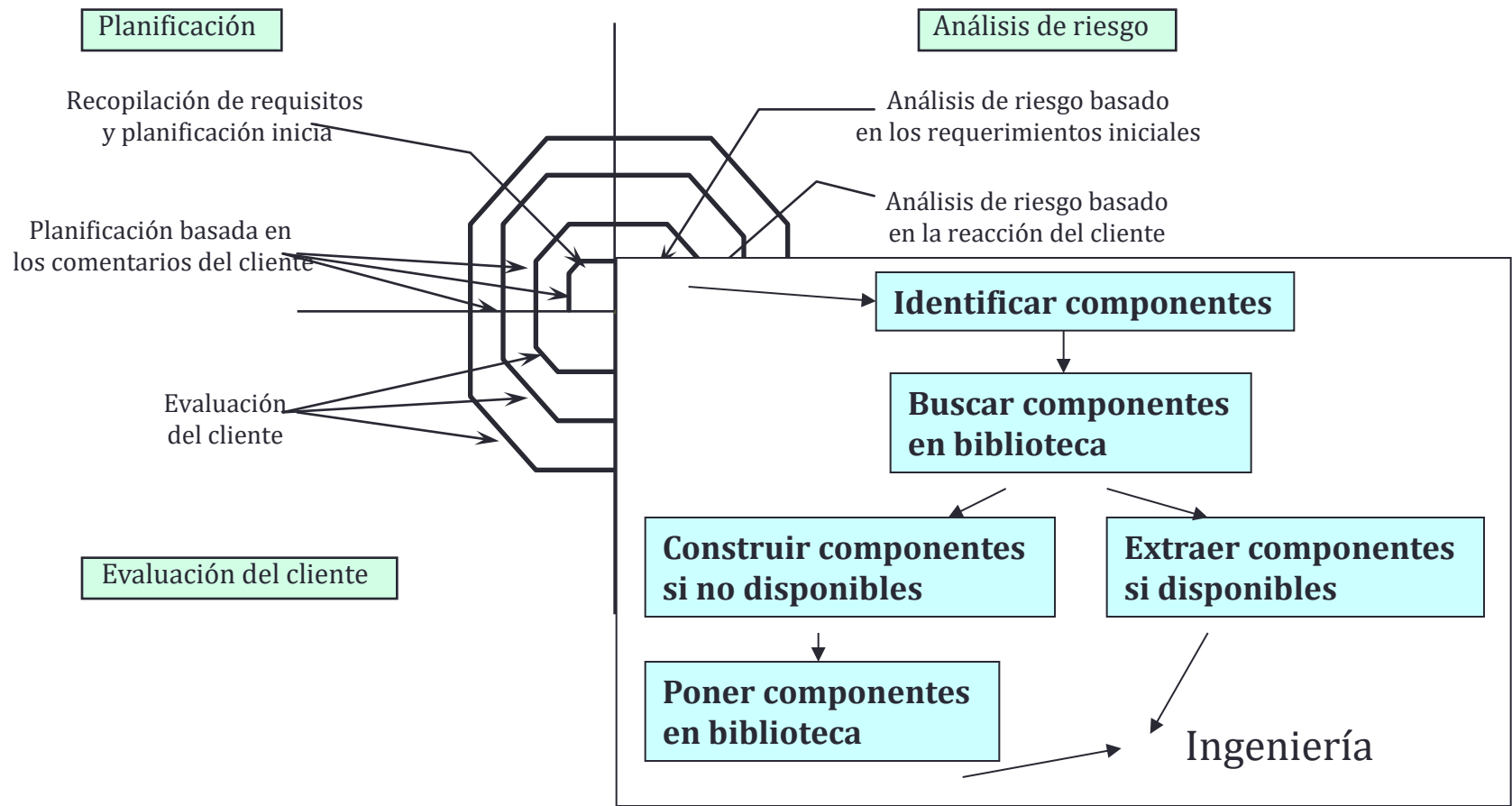
- 😊 Cada versión es evaluada por el usuario ➔ alta interactividad

- 😞 Dificultad en evaluar los riesgos

- 😞 Asegurar que se avanza hacia el producto final

Modelo de Ensamblaje de Componentes

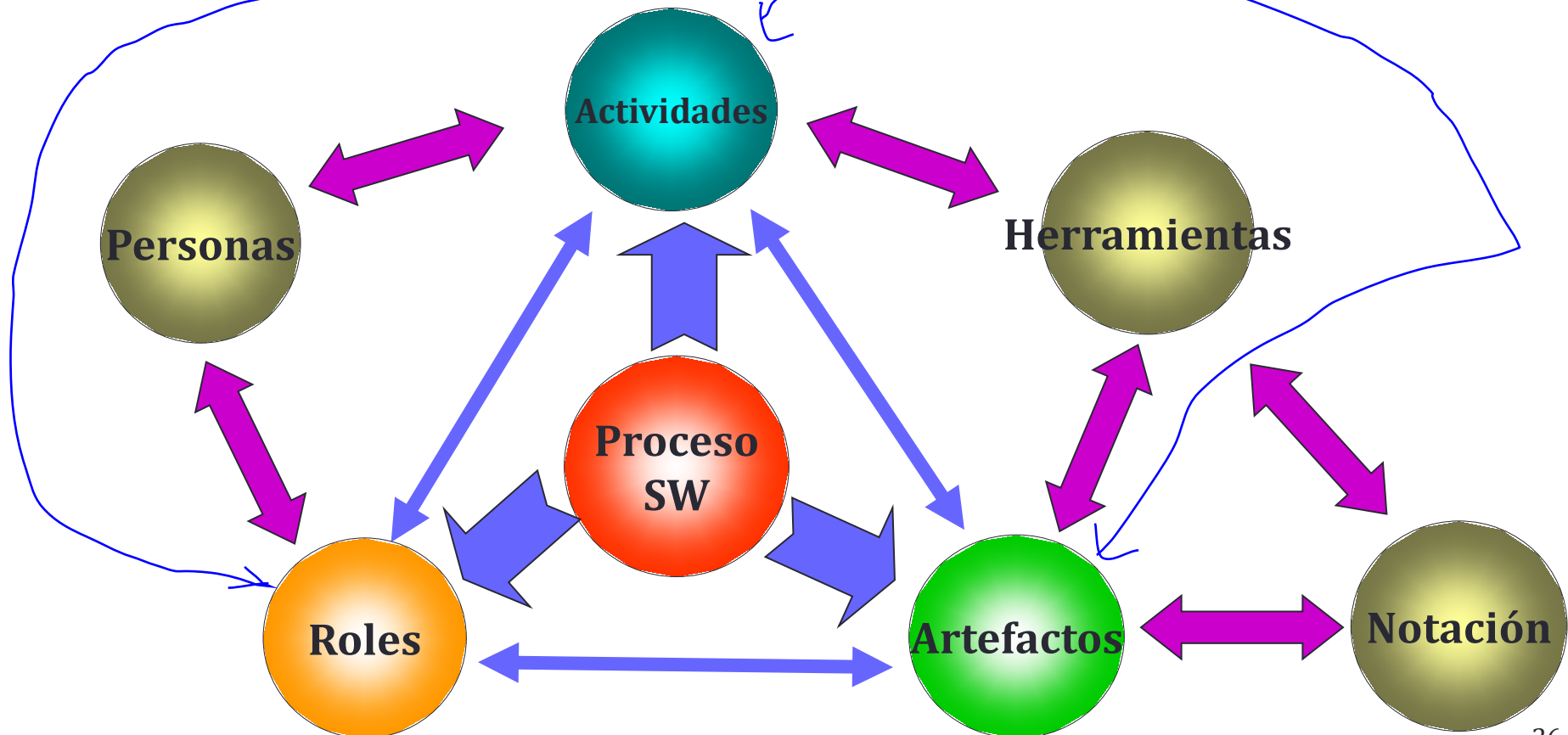
- El modelo en espiral se puede adaptar la fase de ingeniería a nuevas propuestas.



Metodología

en qué punto del proceso se ponen en marcha las actividades.

- En un proyecto de desarrollo de software, la metodología define: **Quién** / **Qué** / **Cómo** / **Cuándo**



Metodología

- Define un proceso explícito de desarrollo de software

(su objetivo es la formalización de las actividades relacionadas con la elaboración de sistemas informáticos)

- Dicho proceso debe ser:
 - Reproducible
 - Definido
 - Medible en cuanto a rendimiento
 - Optimizable
 - ...

Metodología

No existe una metodología de software universal.

Metodologías estructuradas

Metodologías orientadas a objetos

RUP

Metodologías Tradicionales vs. *Metodologías Ágiles*

RUP

XP

ANEXO:

Metodologías

- (RUP) Rational Unified Process
- Metodologías Ágiles

Proceso Unificado de Rational (RUP)



Proceso de Desarrollo de Software
(Rational – IBM)

Utiliza UML, como lenguaje de modelado

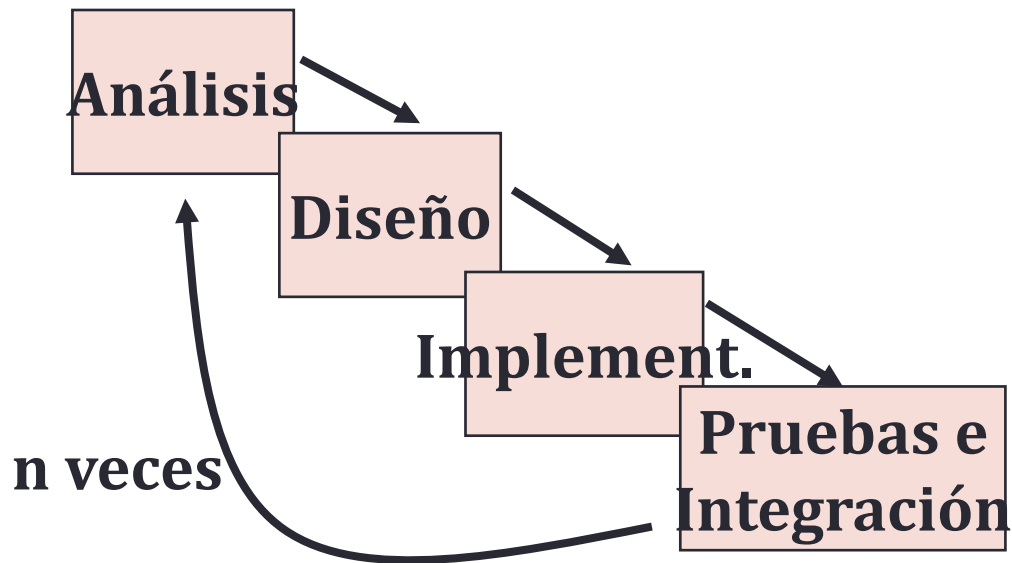
Características:

- *Proceso Dirigido por los Casos de Uso*: desde la especificación hasta el mantenimiento
- *Proceso Iterativo e Incremental*: las iteraciones en función de la importancia de los casos de uso y el estudio de riesgos.
- *Proceso Centrado en la Arquitectura*: reutilizable y como guía hasta la solución

RUP

- Iterativo e Incremental

Las actividades se realizan en una mini-cascada con un alcance limitado por los objetivos de la iteración

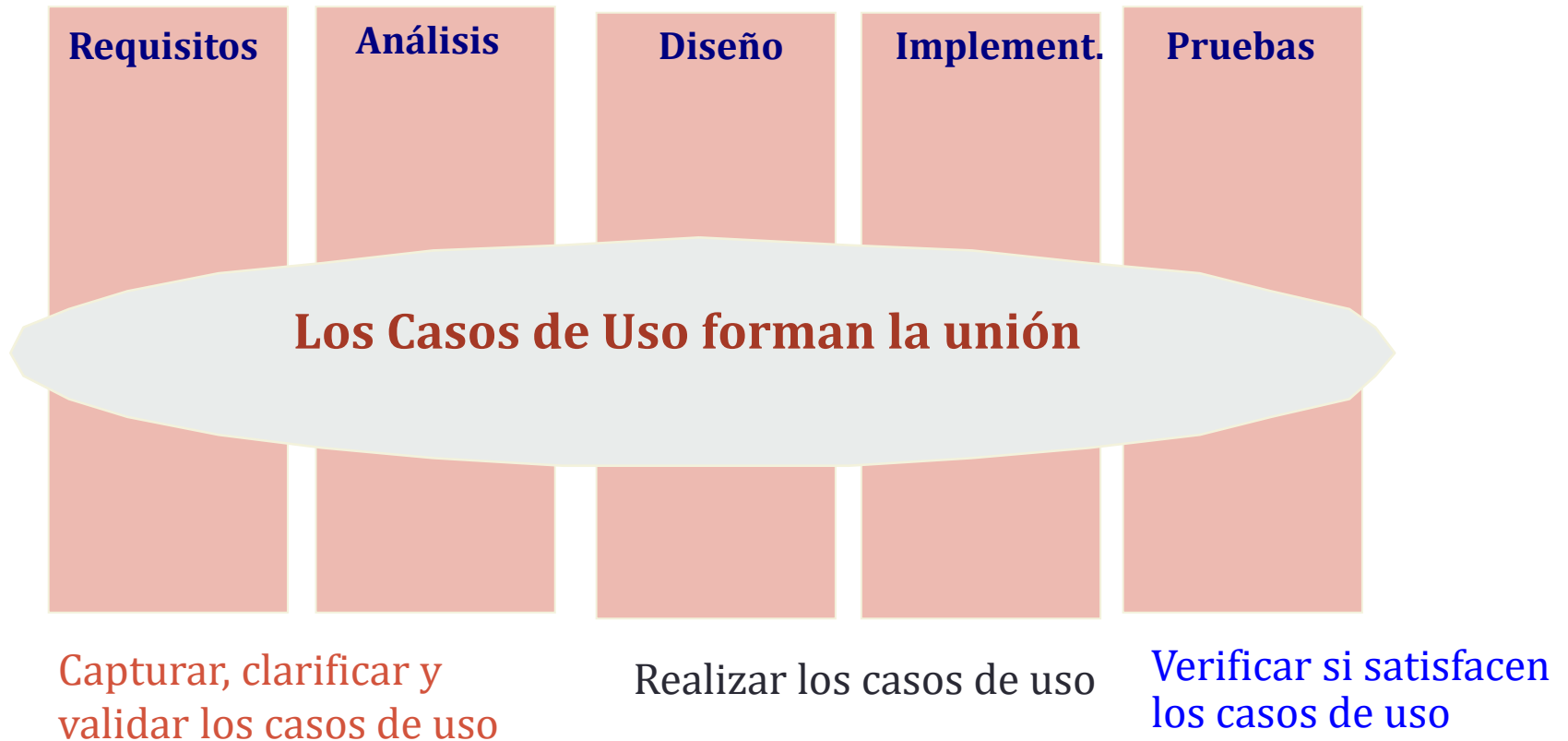


ACTIVIDADES DE LA ITERACIÓN

- Planificar la iteración (riesgos)
- Análisis de **Casos de Uso** y Escenarios
- Diseño Opciones Arquitectónicas
- Implementación
- Pruebas
- Integración
- Evaluación de la entrega
- Preparación de la entrega

RUP

- Dirigido por los casos de uso



RUP

Vista dinámica

Vista estática

Eje Horizontal: Organización a lo largo del tiempo



Fases

Flujos de control de proceso

Modelado de negocios

Requisitos

Análisis y diseño

Implementación

Prueba

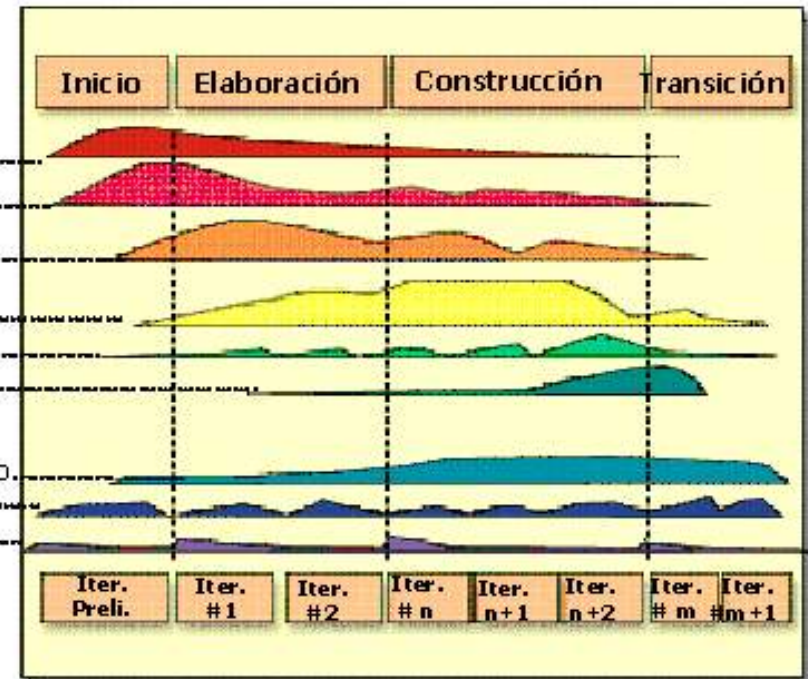
Despliegue

Flujos de control de apoyo

Configuración y manejo del cambio

Administración del proyecto

Entorno

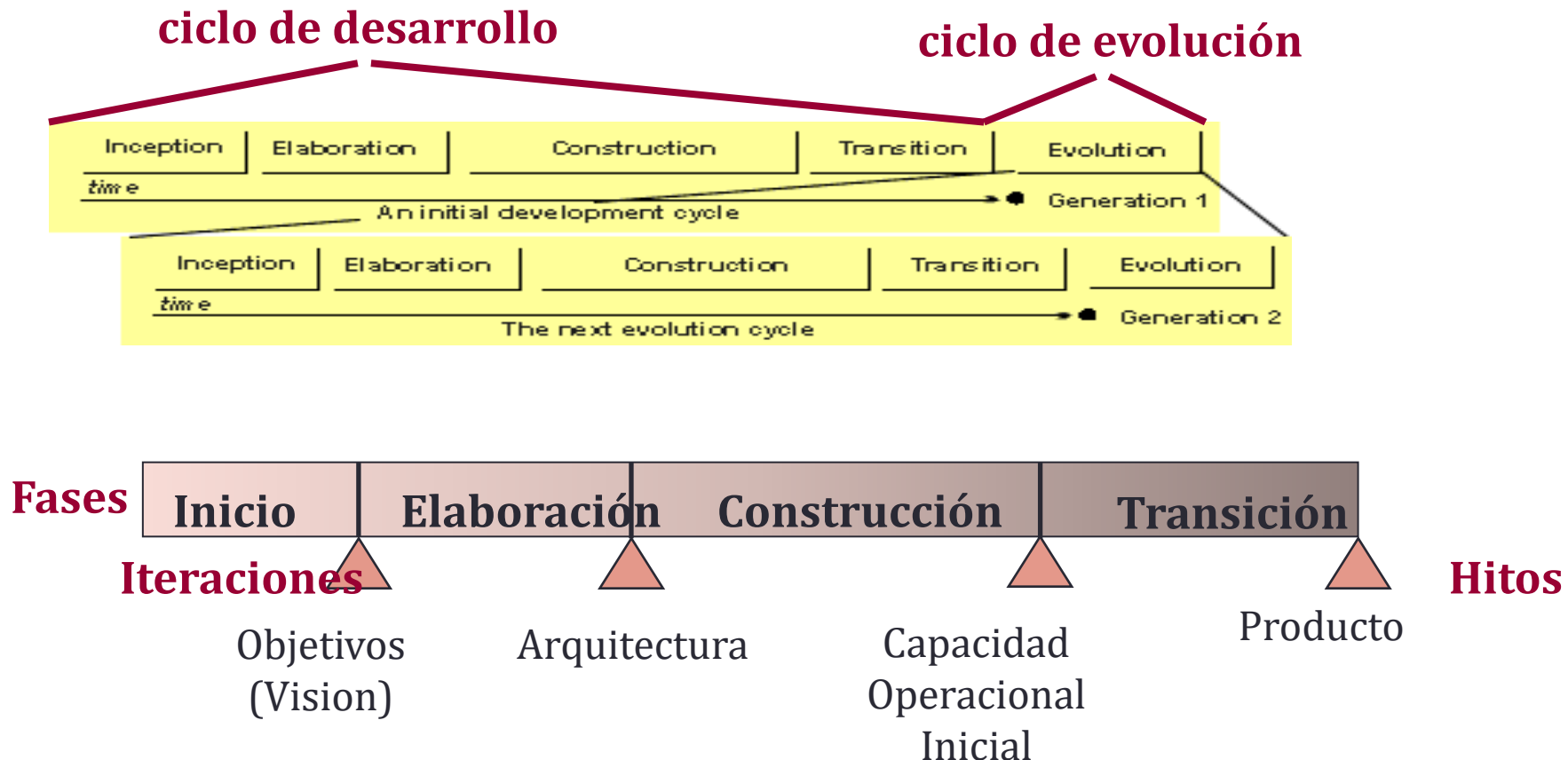


Iteraciones

RUP

Vista dinámica

- Ciclos, Fases, Iteraciones e Hitos



RUP

Vista dinámica

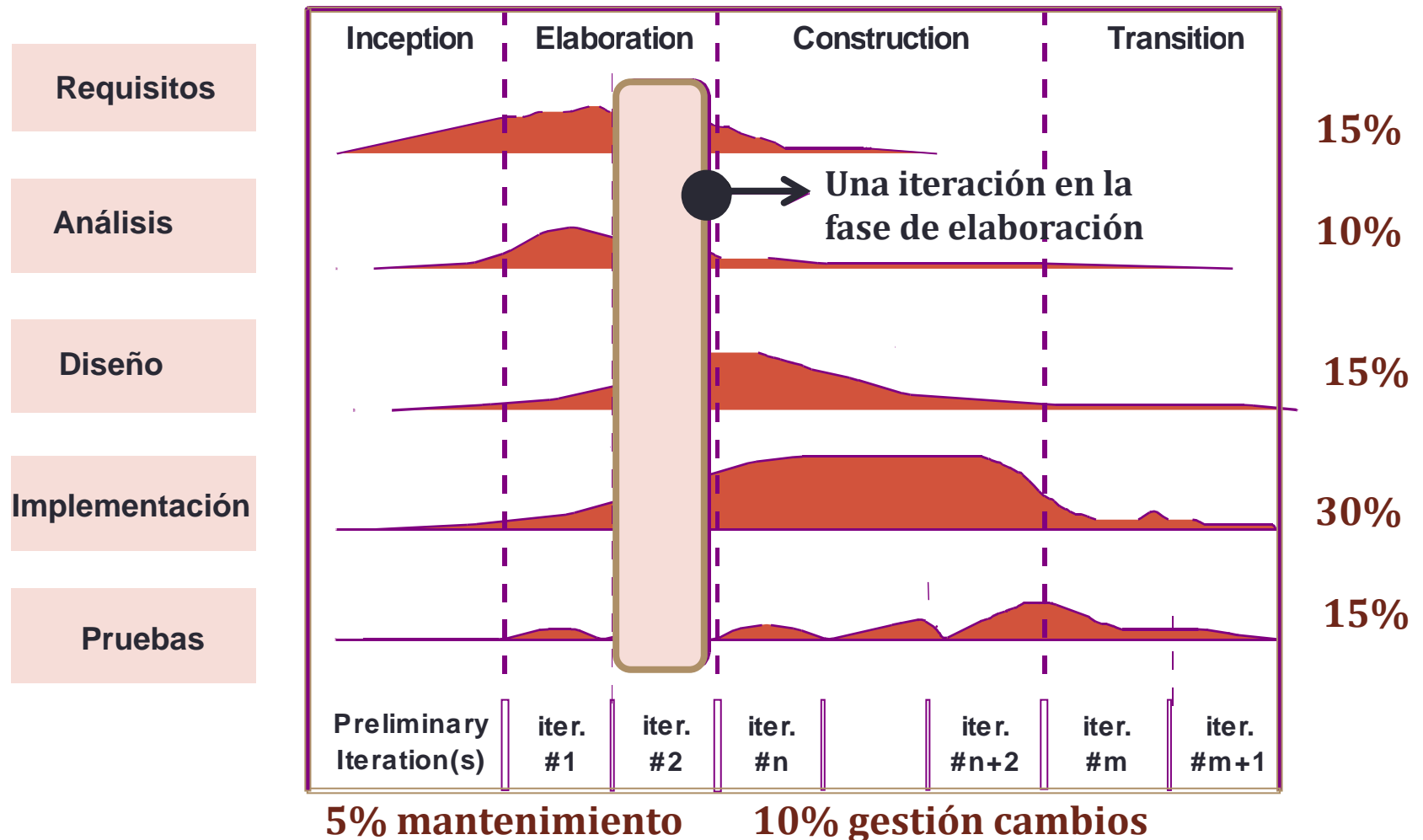
- Fases
 - *Inicio (Estudio de Oportunidad)*
 - Se define el ámbito y objetivos del proyecto
 - Se define la funcionalidad y capacidades del producto
 - *Elaboración*
 - El dominio del problema y la funcionalidad deseada se estudian en profundidad
 - Se define una arquitectura básica
 - Se planifica el proyecto considerando los recursos disponibles

RUP

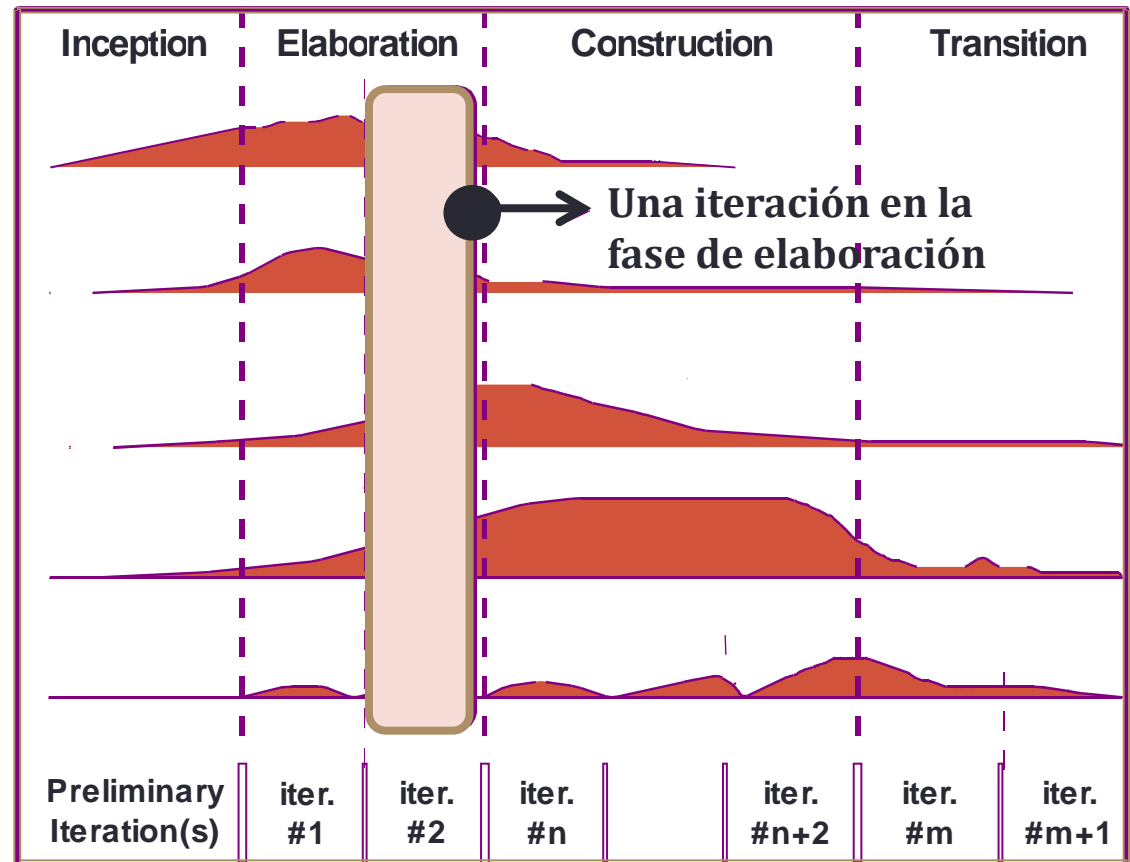
Vista dinámica

- *Construcción*
 - En cada iteración se realizan tareas de análisis, diseño e implementación
 - Se refina la arquitectura
 - Una parte importante del trabajo se dedica a programación y pruebas
 - Se documenta tanto el sistema construido como la utilización del mismo
 - Esta fase proporciona un producto construido y una documentación
- *Transición*
 - Se entrega al usuario para su uso real
 - Se realizan tareas de marketing, empaquetado, instalación, configuración, entrenamiento, soporte, mantenimiento, ...
 - Los manuales de usuario, instalación, ... se completan y refinan

RUP - Distribución de esfuerzos respecto actividades



RUP - *Distribución de esfuerzos respecto fases*



Esfuerzo:	5%	10%	20%	65%
Duración:	10%	10%	30%	50%

RUP

Vista estática

- Flujos de Trabajo (Workflows)

<i>Flujo de Trabajo</i>	<i>Descripción</i>
Modelado del Negocio	Los procesos del negocio se modelan utilizando casos de uso del negocio
Requisitos	Se definen los actores que interactúan en el sistema y se desarrollan casos de uso para modelar los requisitos del sistema
Análisis y diseño	Se crea y documenta un modelo de diseño utilizando modelos arquitectónicos, modelos de componentes, modelos de objetos y modelos de interacción.
Implementación	Se implementan y estructuran en subsistemas los componentes del sistema. La generación automática de código de los modelos de diseño ayuda a acelerar este proceso.
Pruebas	Las pruebas son un proceso iterativo que se llevan a cabo conjuntamente con la implementación. Cuando finaliza la implementación se realizan las pruebas del sistema.
Despliegue	Se crea una <i>release</i> (versión) del producto, se distribuye a los usuarios y se instala en su lugar de trabajo.

RUP

Vista estática

- Flujos de Trabajo (Workflows)

<i>Flujo de Trabajo</i>	<i>Descripción</i>
Configuración y gestión de cambios	Gestiona los cambios en el sistema
Gestión del proyecto	Gestiona el desarrollo del sistema
Entorno	Desarrollo de herramientas software apropiadas para los equipos de desarrollo de software.

Metodologías Ágiles

Las Metodologías Ágiles valoran:

- Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas
- Desarrollar software que funciona más que conseguir una buena documentación \Rightarrow Minimalismo respecto del modelado y la documentación del sistema
- La colaboración con el cliente más que la negociación de un contrato
- Responder a los cambios más que seguir estrictamente una planificación

<http://www.agilealliance.com>

Metodologías Ágiles

Principios de las Metodologías Ágiles (1/2)

- 1.- La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software utilizable.
- 2.- Dar la bienvenida a los cambios. Los procesos ágiles aplican los cambios para que el cliente sea competitivo.
- 3.- Entregar el software desarrollado frecuentemente con el menor intervalo de tiempo posible entre una entrega y la siguiente
- 4.- La gente de negocios y los desarrolladores trabajan juntos a través de un proyecto
- 5.- Construir proyecto empujados por motivaciones personales. Dar el entorno que necesitan las personas y confiar en ellos.

Metodologías Ágiles

Principios de las Metodologías Ágiles (2/2)

- 6.- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo
- 7.- Desarrollar software es la primera medida de progreso.
- 8.- Los procesos ágiles promueven un desarrollo llevadero. Los patrocinadores, desarrolladores y usuarios son capaces de mantener una paz constante
- 9.- La atención continua a la calidad técnica y al buen diseño incrementa la agilidad
- 10.- La simplicidad es esencial
- 11.- Las mejores arquitecturas, requisitos y diseños surgen de la propia organización del equipo
- 12.- En intervalos regulares, el equipo reflexiona en como llegar a ser más efectivo, sincronizar y ajustar su comportamiento.

Metodologías Ágiles

- Comparativa

Metodología Ágil	Metodología No Ágil
El cliente es parte del equipo de desarrollo (además <i>in-situ</i>)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura	La arquitectura es esencial

Metodologías Ágiles

- Comparativa

Metodología Ágil	Metodología No Ágil
Heurísticas	Rigurosas
Tolerante a los cambios	Resistente a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado

Principales Metodologías Ágiles

- ⇒ Extreme Programming (XP) <http://www.extremeprogramming.org>
- ⇒ SCRUM <http://www.controlchaos.com>
- ⇒ Crystal Methods <http://alistair.cockburn.us/Crystal+methodologies>
- ⇒ Adaptive Development Software (ADS) <http://www.adaptivesd.com>
- ⇒ Feature-Driven Development (FDD)
<http://www.featuredrivendevelopment.com>
- ⇒ Lean Development (LD) <http://www.poppendieck.com>

Extreme Programming (XP)



Kent Beck, Ward Cunningham y Ron Jeffries

www.extremeprogramming.org

www.xprogramming.com

- Diseñado para entornos dinámicos
- Pensado para equipos pequeños (≤ 10 programadores)
- Orientado fuertemente hacia la codificación
- Énfasis en la comunicación informal, verbal
- Otros valores: simplicidad, realimentación y coraje

Historias, Iteraciones, Versiones, Tareas y Casos de Prueba

- ✓ El cliente selecciona la **siguiente versión** a construir, eligiendo las **características funcionales** que considera más valiosas (llamadas **Historias**) de un conjunto posible de historias, siendo informado de los *costes* y del *tiempo* que costará su implementación.
- ✓ Los programadores **convierten las historias** en **tareas a realizar** y a continuación convierten las **tareas** en un **conjunto de casos de prueba** para demostrar que las tareas se han finalizado.
- ✓ Trabajando con un compañero el programador **ejecuta los casos de prueba** y **evoluciona el diseño** intentando mantener su simplicidad.

XP

Prácticas

El juego de la Planificación

Pruebas

Propiedad Colectiva

Entregas Pequeñas

Metáfora

Semanas de 40 horas

Refactorización

Diseño Sencillo

El Cliente siempre con el Desarrollador

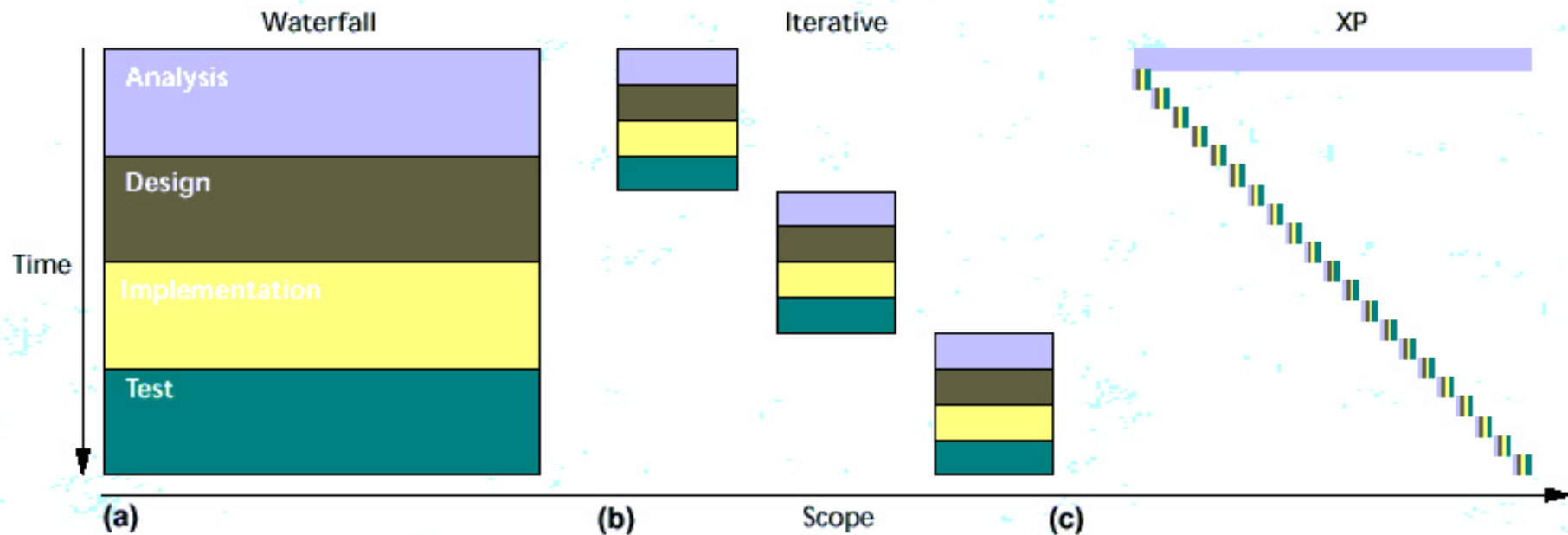
Programación en Parejas

Integración Continua

Estándares de Codificación

XP

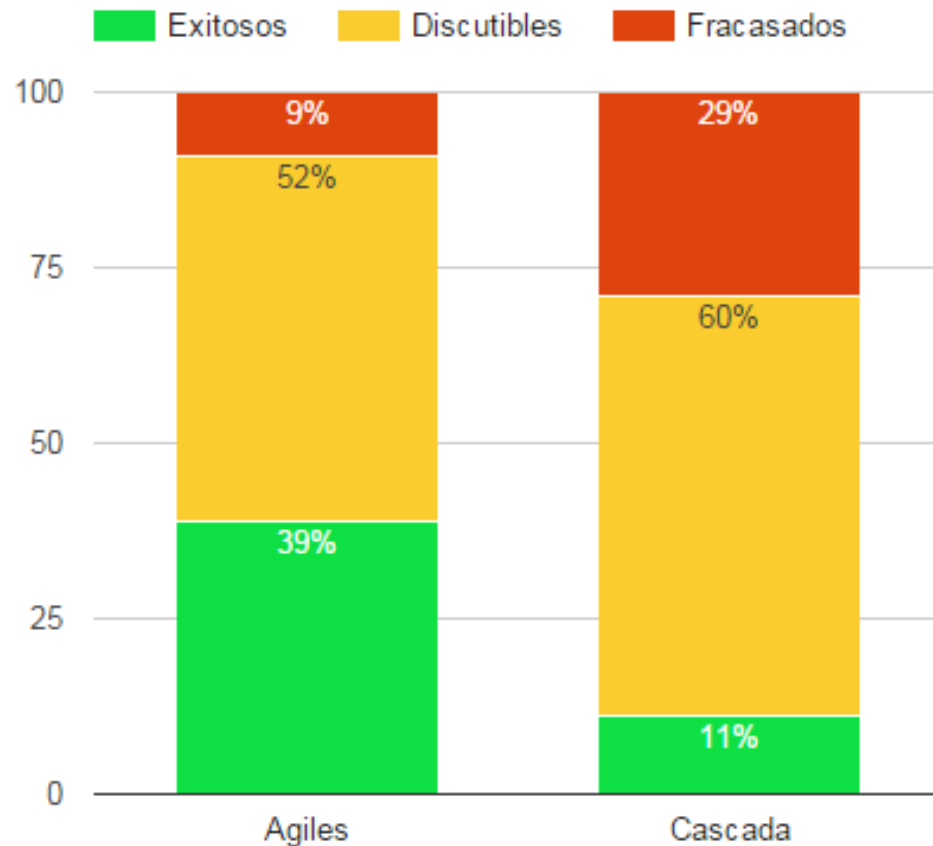
Comparativa



Agiles vs. Cascada

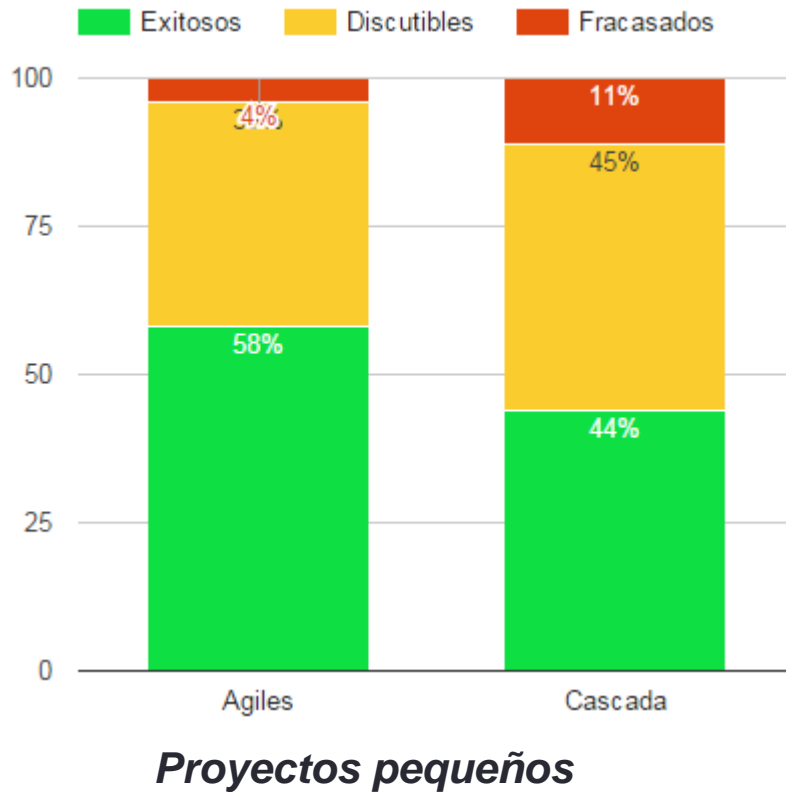
Comparativa

Comparativa del **éxito** de los proyectos en función de la **metodología** seguida para su desarrollo 2011-2015

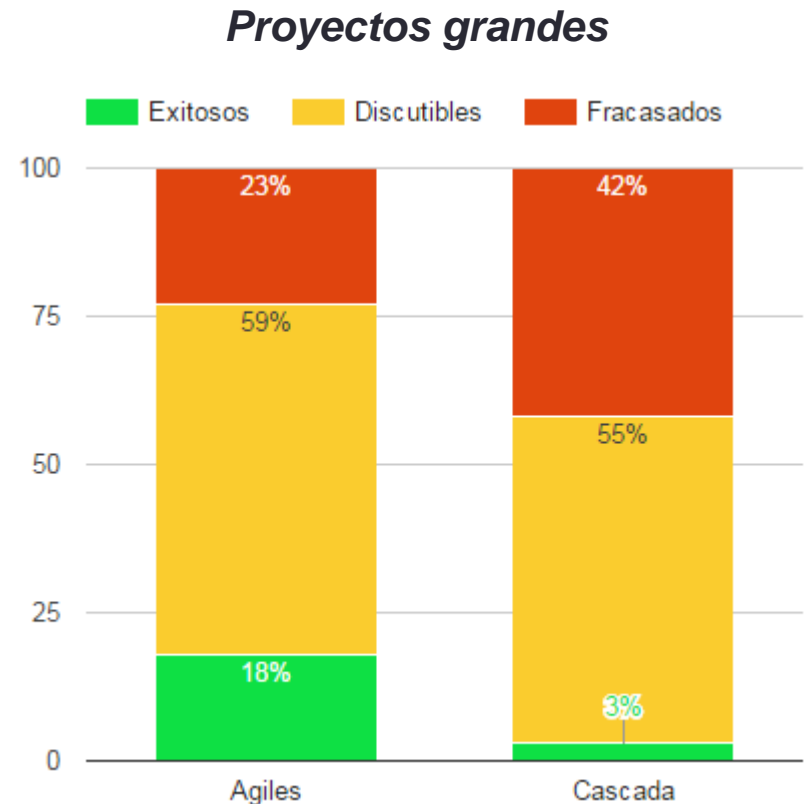


Fuente. Informe del CAOS 2015 (Chaos Report 2015) o cómo de bien o mal fueron los proyectos en el año 2015

Agiles vs. Cascada



Comparativa



Fuente. Informe del CAOS 2015 (Chaos Report 2015) o cómo de bien o mal fueron los proyectos en el año 2015