

# **Tema 2**

## **Fases del Proyecto y Organización**

**2.1 Ciclos de vida y Metodologías**

**2.2 Proyecto Genérico**

**2.3 Organización Estructural**

**2.4 Organización en la Empresa**

Pablo.Bermejo@uclm.es

## 2.1 Ciclos de vida y Metodologías

- El **ciclo de vida** del proyecto se define por un conjunto de fases y las relaciones entre ellas.
- La **metodología** elegida nos indica cómo y cuándo deben realizarse las actividades (o si no hacen falta) correspondientes a cada fase.
  - Metodología de empresa: hay empresas que definen su propio modo de trabajo (o que no tienen).
  - Metodología estándar: XP, RUP,...
- Las metodologías clásicas (<1990) de proyectos software han perdido validez rápidamente:
  - Se exige más rapidez en la respuesta y comunicaciones

## 2.1 Ciclos de vida y Metodologías

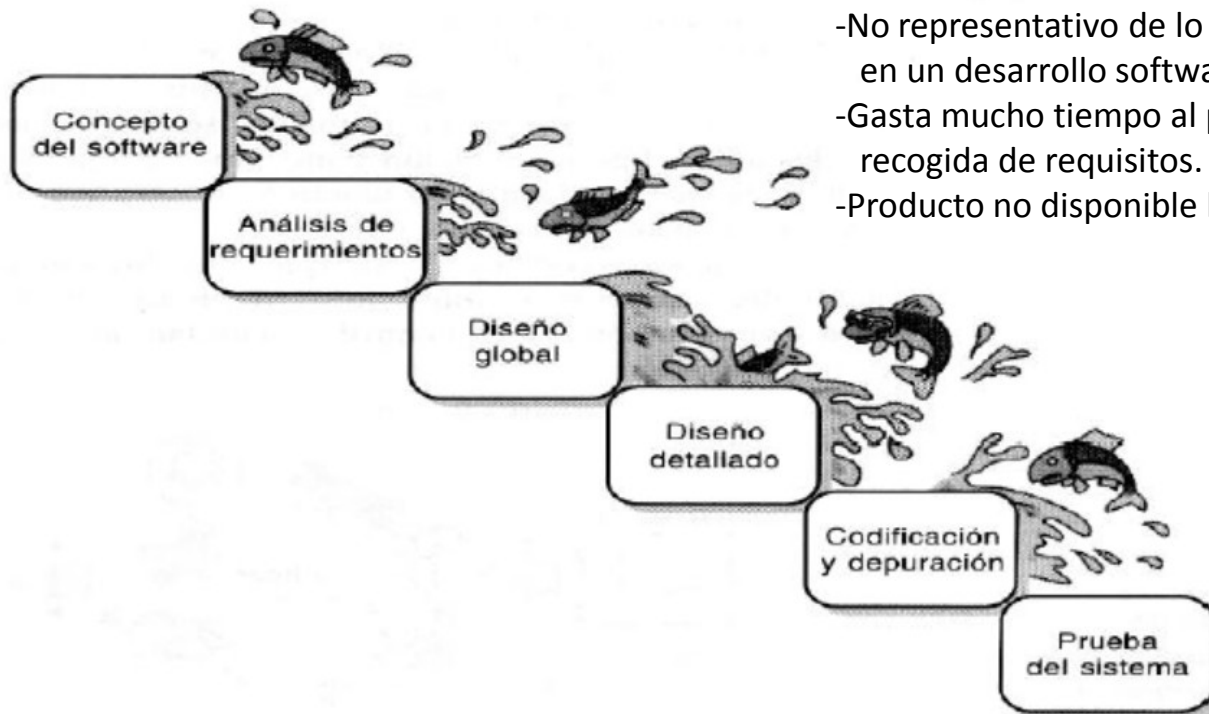
- Los interesados (positivos y negativos) intercambian información en tiempo real: más flexibilidad
- Un proyecto software dista mucho de un proyecto burocrático
- **No** hay una **metodología mejor** para todos los casos. Si te toca decidir o dar tu opinión, ten en cuenta:
  - ¿Sois un grupo grande o pequeño?
  - ¿El cliente quiere una sola entrega final o quiere participar en el desarrollo?
  - Siempre elige metodologías actuales (hoy, desconfía de los años 80!)

## 2.1 Ciclos de vida y Metodologías

- Es posible que por instinto sepas realizar tu trabajo y dirigir a tu equipo **sin metodología**; pero en cuanto el proyecto crezca estarías destinado al **fracaso**.
- Ciclo de vida  $\subset$  Metodología  
Pero lo contrario no es verdad
- Ciclo de vida  $\cap$  Metodología  $\neq \emptyset$
- Ciclo de vida del **proyecto** Vs. Ciclo de vida del **producto**
  - La última fase del producto es su retiro
  - Un producto puede tener asociados varios proyectos:
    - Proyecto de desarrollo
    - Proyecto para añadir funcionalidad
    - Proyecto de migración

## 2.1 Ciclos de vida y Metodologías

- Ejemplos de ciclos de vida bien conocidos (no confundir con metodologías):
  - Cascada (70s)

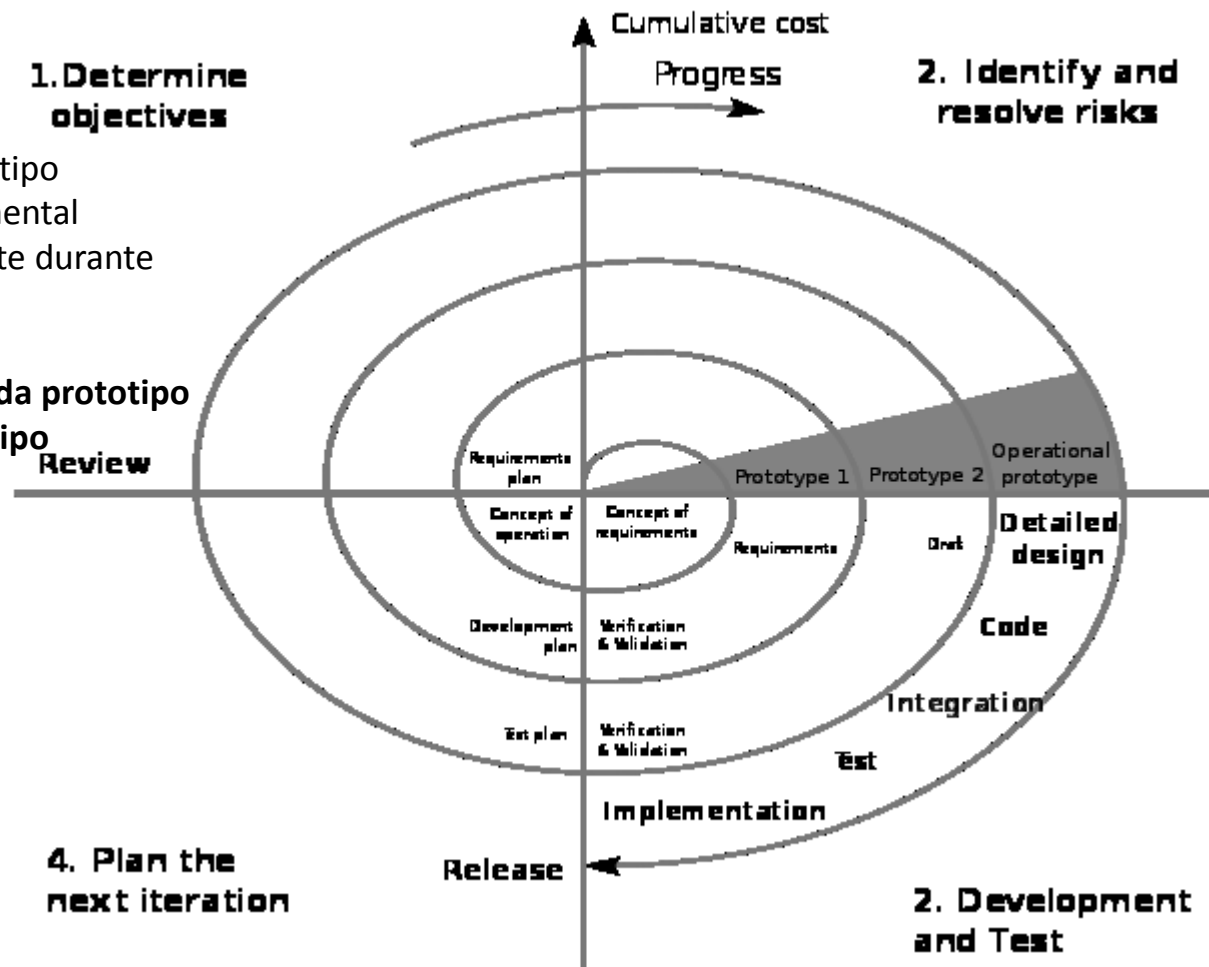


- Una fase no comienza hasta acabar la anterior
- No representativo de lo que realmente ocurre en un desarrollo software
- Gasta mucho tiempo al principio para evitar fallos en la recogida de requisitos.
- Producto no disponible hasta el final

## 2.1 Ciclos de vida y Metodologías

### – Espiral (80s)

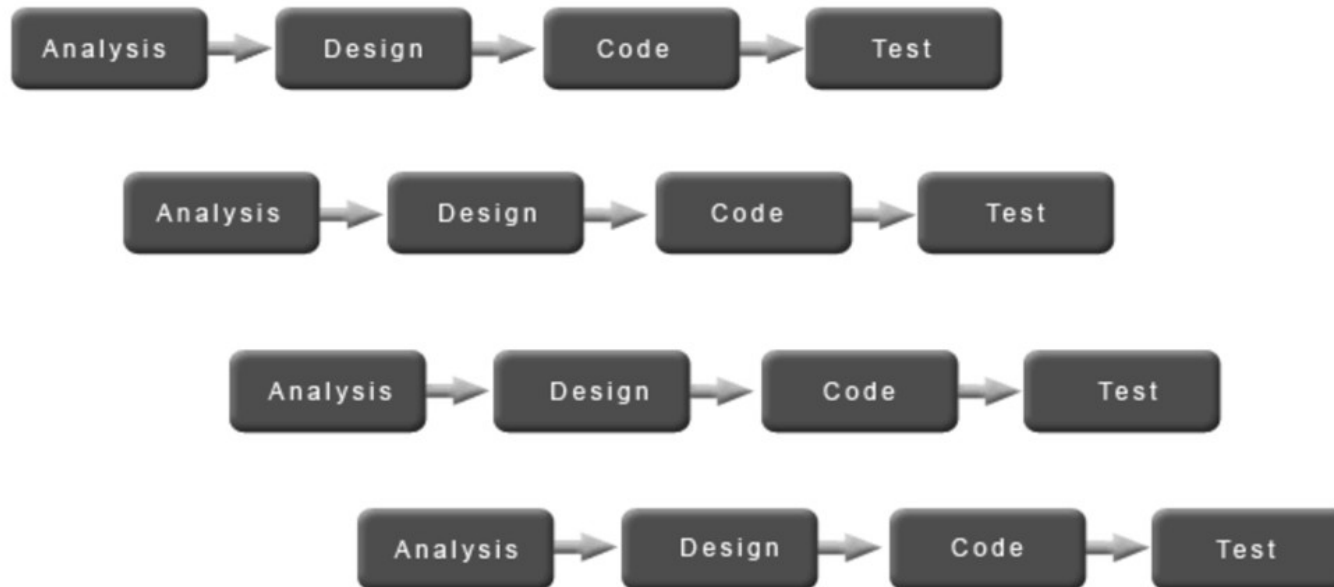
- De cada iteración resulta un prototipo
- Refinamiento del producto incremental
- Requiere la participación del cliente durante todo el ciclo de vida
- Más flexible a cambios
- Realiza **gestión de riesgos** para cada prototipo
- Solo es funcional el último prototipo



## 2.1 Ciclos de vida y Metodologías

### – Incremental (80s)

- El producto se construye por entregas
- Parecido al modelo de espiral, pero sin gestión de riesgos

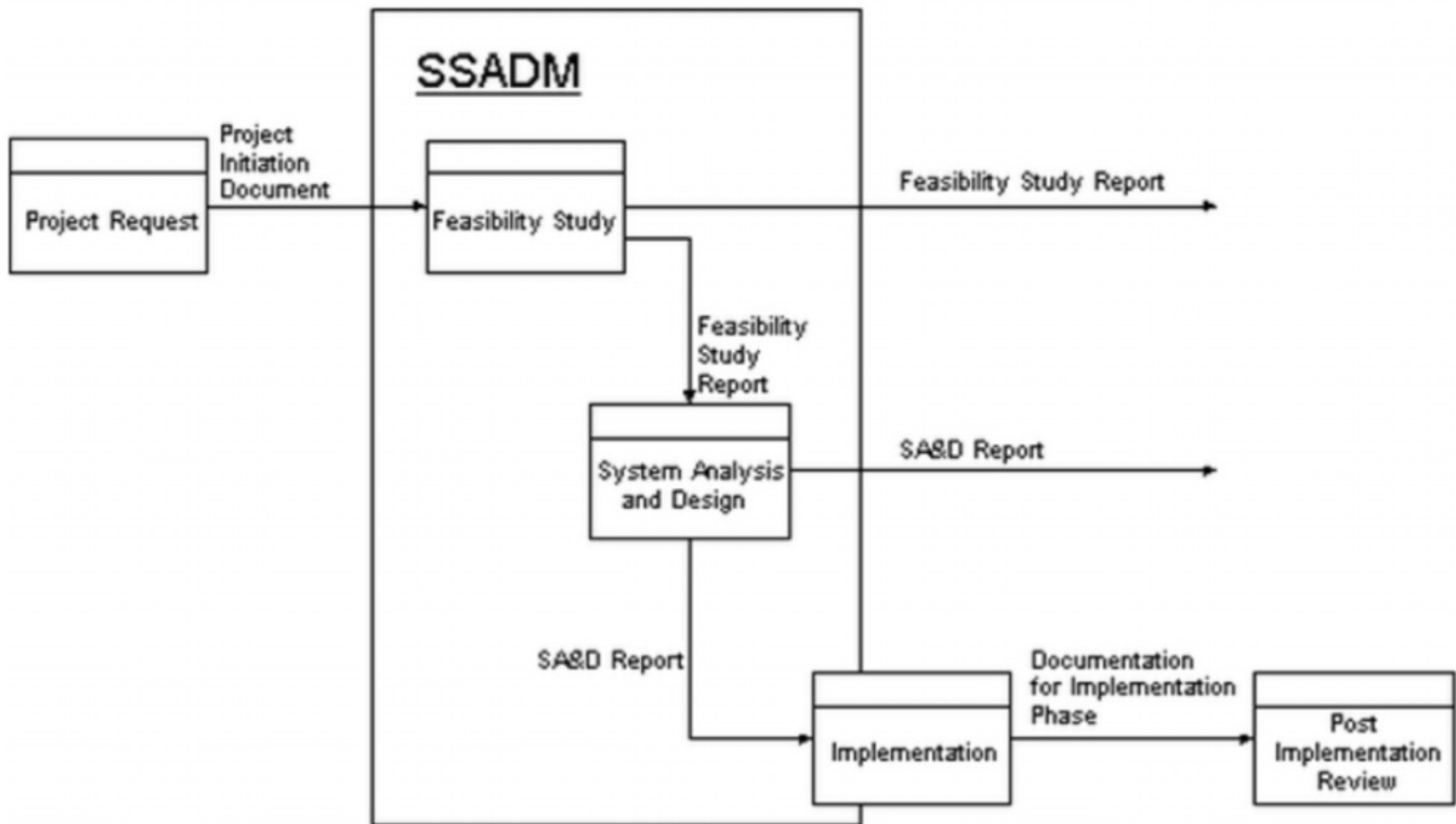


(página en blanco)

## 2.1 Ciclos de vida y Metodologías

- Ejemplos de metodologías (aunque no profundicemos en los tipos de entregables, no confundir con ciclos de vida):
  - **Structured System Analysis and Design Method (SSADM) (80s)**
    - Ciclo de vida de cascada con fases un poco superpuestas
    - Metodología **para las fases de análisis de requisitos y diseño del sistema.**
    - No contiene especificaciones para desarrollo ni pruebas.
    - Estándar abierto y gratuito. Disponibles herramientas CASE (Computer-aided Software Engineering).
    - Utilizado en departamentos de estado de Reino Unido

## 2.1 Ciclos de vida y Metodologías

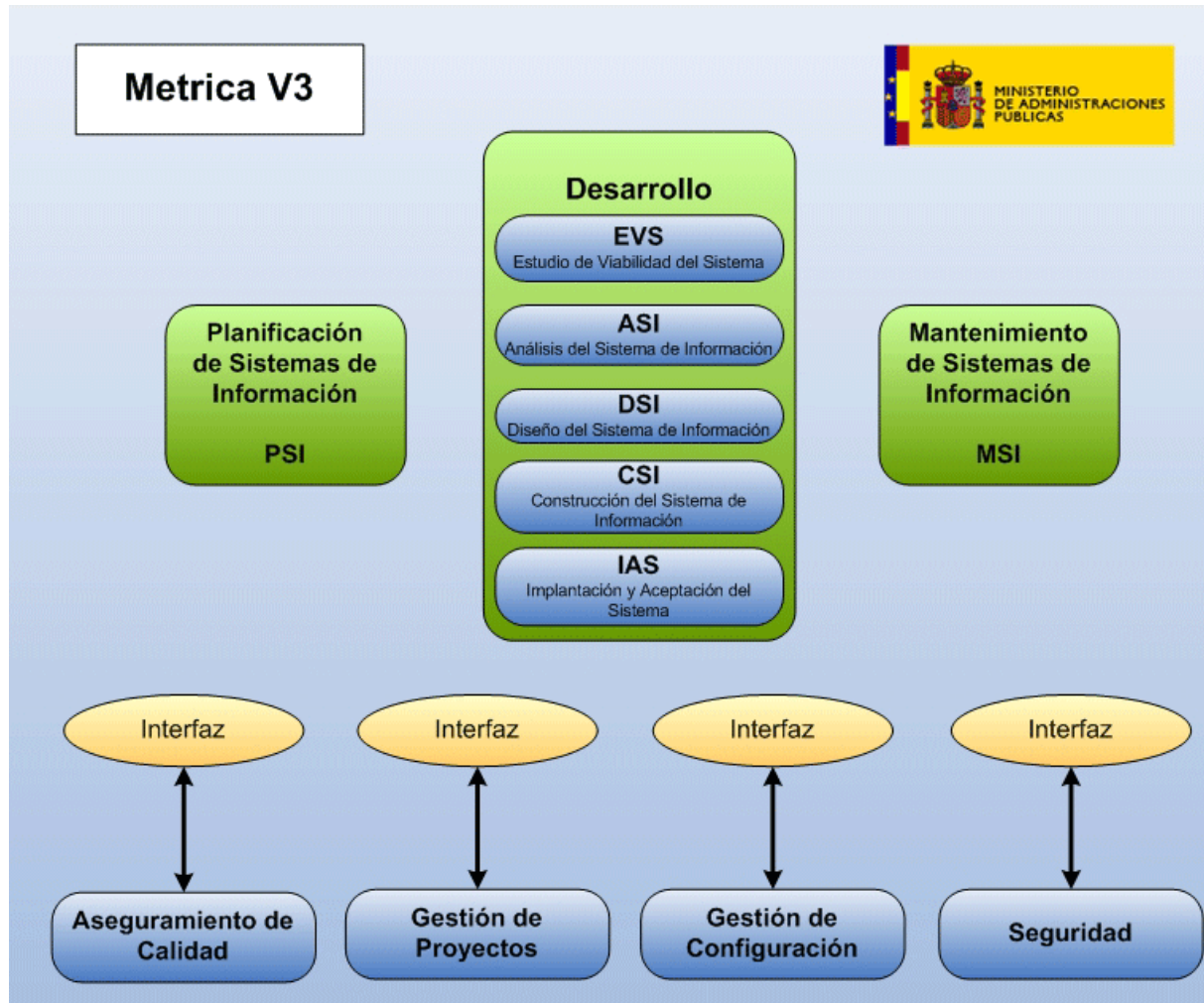


## 2.1 Ciclos de vida y Metodologías

### – METRICA v3 (00s)

- Ciclo de vida cierta similitud a cascada
- Proyectos públicos en España para desarrollar Sistemas de Información.
- Estructura de Procesos: Planificación, Desarrollo y Mantenimiento. Las salidas de un proceso se utilizan como entradas de otros procesos.
- Gran importancia a la captura de requisitos.
- Gestión de planificación, coste y calidad a lo largo de todo el ciclo de vida.
- Múltiples herramientas CASE.

## 2.1 Ciclos de vida y Metodologías



### Ejemplo de entregables PSI

- Modelo de sistemas de información
- Arquitectura tecnológica
- Plan de mantenimiento

### DESARROLLO

- Valoración de Riesgos (EVS)
- Coste/beneficio (EVS)
- Requisitos (ASI)
- Glosario de términos (ASI)
- Resultado de Pruebas (CSI)
- Producto software(CSI)
- Plan de Implantación (IAS)

### MSI

- Catálogo peticiones de cambio
- Estudio de la petición
- Plan de cambio

## 2.1 Ciclos de vida y Metodologías

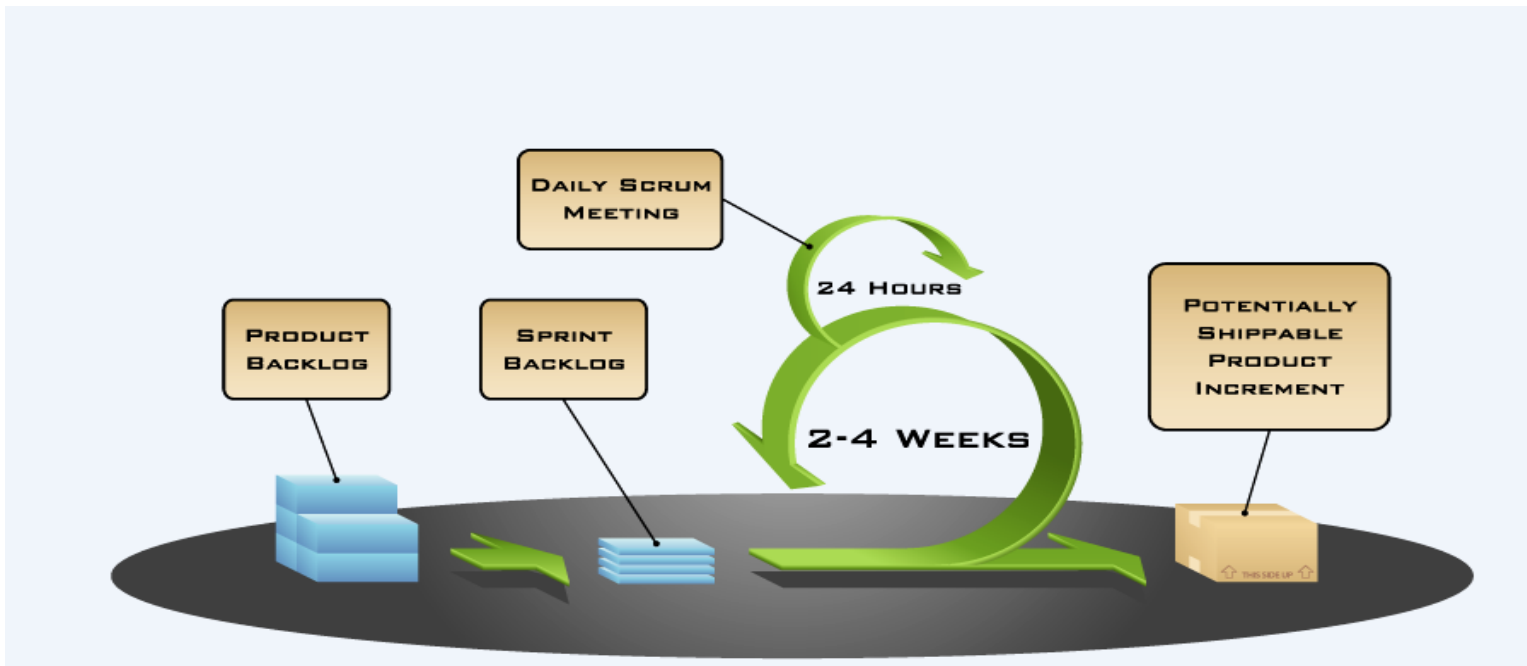
### – Scrum (90s)

- Ciclo de vida incremental
- Para equipos pequeños o grandes que se dividen en subgrupos
- Plan inicial (backlog) + *sprints* incrementales
- **Sprint**: fase de 1 a 4 semanas de programación y reuniones intensivas para añadir funcionalidad a la última versión del producto.
- Antes de cada sprint, el equipo se reúne y reordena las tareas del backlog según prioridad.
- **Durante el sprint, no se permiten cambios.**
- **La fecha de cierre de un sprint es inalterable.**
- **No gestiona la calidad ni riesgos**
- Esta metodología define papeles:
  - Scrum Master → project manager
  - Product Owner → miembro del equipo que representa al cliente
  - Equipo
- Reuniones diarias de 20 minutos.

## 2.1 Ciclos de vida y Metodologías

Ejemplo de entregables:

- El Product Owner genera informes de actualización de requisitos para el backlog
- El Scrum Master genera informes de las nuevas funcionalidades añadidas al producto después del sprint.



## 2.1 Ciclos de vida y Metodologías

- **Extreme Programming , XP (90s)**
  - Prioridad a dar respuesta rápida a cambios del cliente
  - Código de alta calidad:
    - Múltiples pruebas
    - Programación en parejas
    - Refactorización
  - Todos los miembros del equipo son iguales
  - Falta de documentación sobre el diseño global
  - Programación durante corto período de tiempo + pruebas

## 2.1 Ciclos de vida y Metodologías

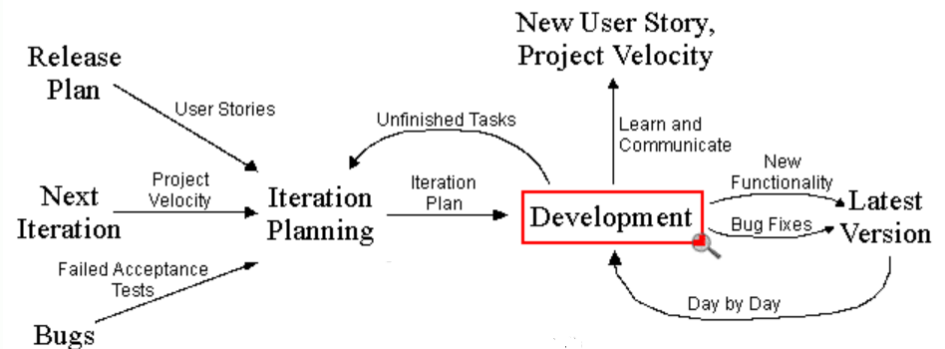
Se asume que los requisitos del cliente cambiarán en cualquier momento, así que se Mantiene un *user story*: tarjetas que reflejan la evolución temporal de los requisitos.

### Extreme Programming Planning/Feedback Loops



© J. Donovan Wells

### Iteration



## 2.1 Ciclos de vida y Metodologías

### – Rational Unified Process, RUP (00s)

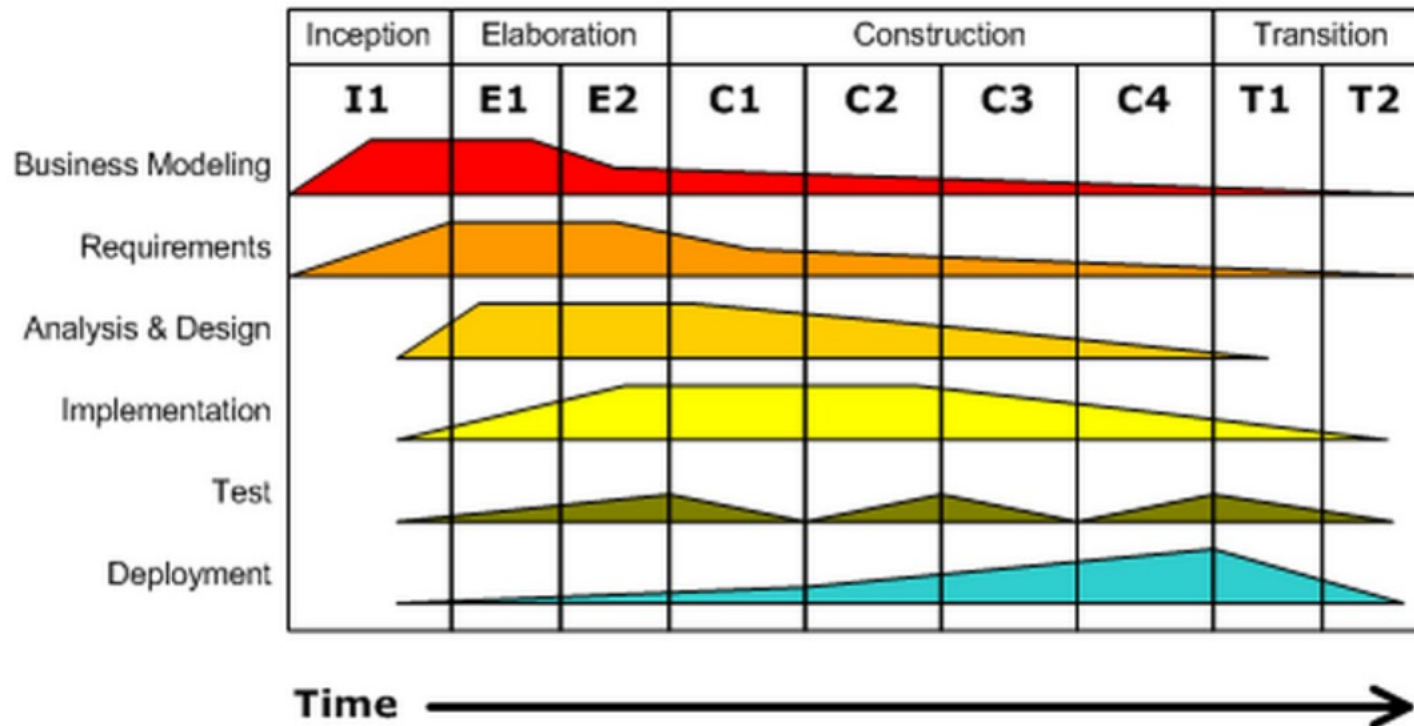


- Ciclo de vida iterativo
- Proyectos largos
- Describe tanto los entregables a realizar como las habilidades necesarias
- Define papeles: jefe de proyecto, gestor de calidad, gestor de configuración,...
- Cada fase finaliza con un hito o entrega
- Fases superpuestas:
  - Inicio: descripción y evaluación del proyecto, riesgos, calidad e interesados.
  - Elaboración: arquitectura del sistema
  - Construcción: implementación
  - Transición: pruebas, entrenamiento y chequeo de calidad.

## 2.1 Ciclos de vida y Metodologías

### Ejemplos de entregables:

- Requisitos del producto (Inicio)
- Casos de uso (Elaboración)
- Código (Construcción)
- Manual de instrucciones(Transición)



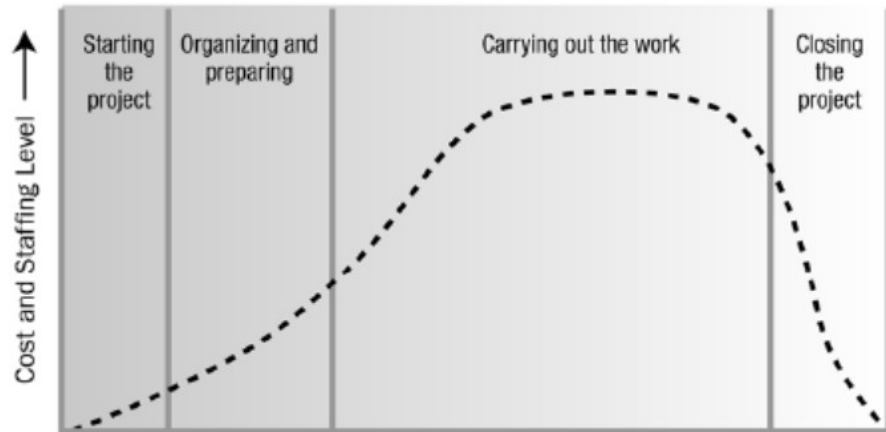
## 2.1 Ciclos de vida y Metodologías

- Conclusiones a bajo nivel:
  - El ciclo de vida describe el tipo de relación entre fases
  - La metodología es un marco de trabajo: fases, roles, entregables,...
  - El tipo de metodología depende del tipo de proyecto y grupo
- Conclusiones a alto nivel:
  - No te cases con ninguna metodología
  - La que hoy funciona, dentro de una década estará obsoleta
  - Hay que conocer todos los tipos de fases y procesos que puedan existir, y así estarás preparado! El PMI es la *biblia* de procesos de gestión de proyectos (a partir del Tema 3).

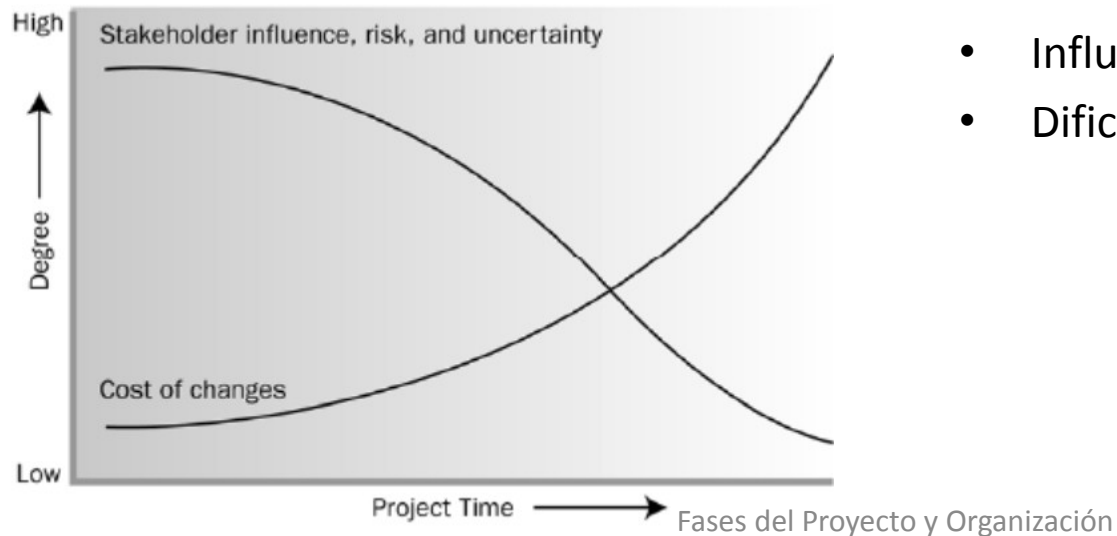
## 2.2 Proyecto Genérico

- Un programa puede tener proyectos que se dirigen con distintas metodologías
- Para poner en común el avance del trabajo, se hablará en términos de un ciclo de vida genérico, para comparar o estudiar:
  - Costes
  - Riesgos
  - Evolución
  - ...
- Un proyecto genérico tiene el siguiente ciclo de vida:
  - Inicio
  - Organización
  - Ejecución
  - Cierre

## 2.2 Proyecto Genérico



- Coste y personal



- Influencia de los riesgos e interesados
- Dificultad para realizar cambios

## 2.2 Proyecto Genérico

- ¿Por qué dividir un proyecto en fases?



- Suponen una base estructurada para controlar el proyecto
- El fin de cada fase sirve como un STOP para revisar



- Cronogramas
- Riesgos
- Entregables
- **Estudiar viabilidad:** *seguir o no seguir*



- Comunicación más sencilla con agentes externos:

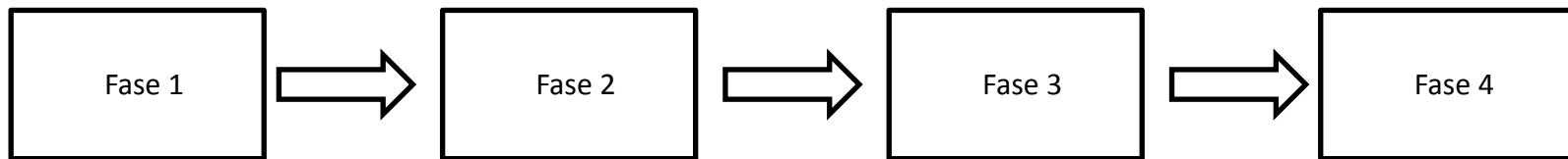


- Gestor de portafolio
- Cliente

## 2.2 Proyecto Genérico

- Tipo de relación entre fases:

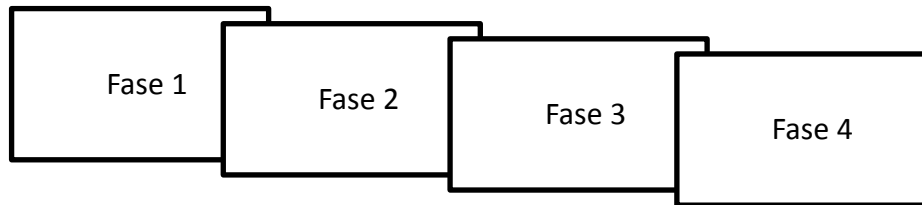
### Fases Secuenciales:



- El trabajo a realizar en una fase es totalmente distinto del resto
- Cada fase finaliza con un hito que ha completado todo el trabajo correspondiente
- No se puede volver atrás
- No permite adelantar tiempo ni recursos
- Ej: cascada

## 2.2 Proyecto Genérico

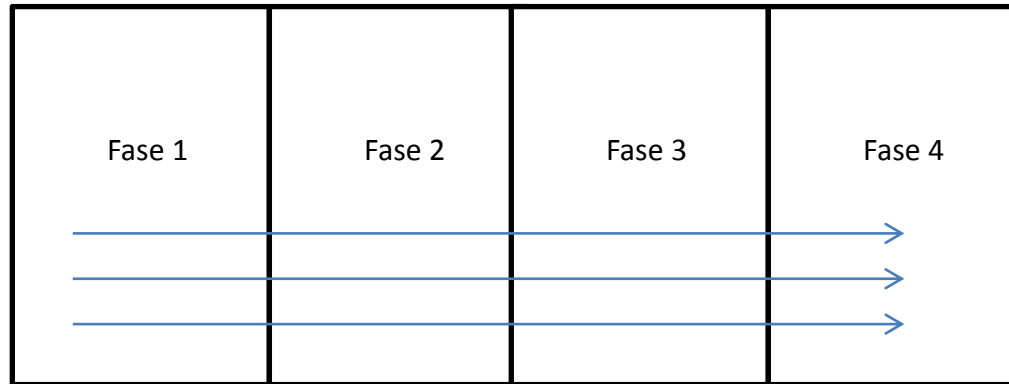
### Fases Superpuestas:



- Una fase se inicia antes de terminar la anterior
- Así puede ahorrarse tiempo, aunque la planificación será menor
- Aunque una tarea dentro de una fase deba empezar, es posible que quede bloqueada por retraso de otra tarea de la fase anterior que sirve como entrada a la actual.
- Ej: SSADM

## 2.2 Proyecto Genérico

### Fases Iterativas:



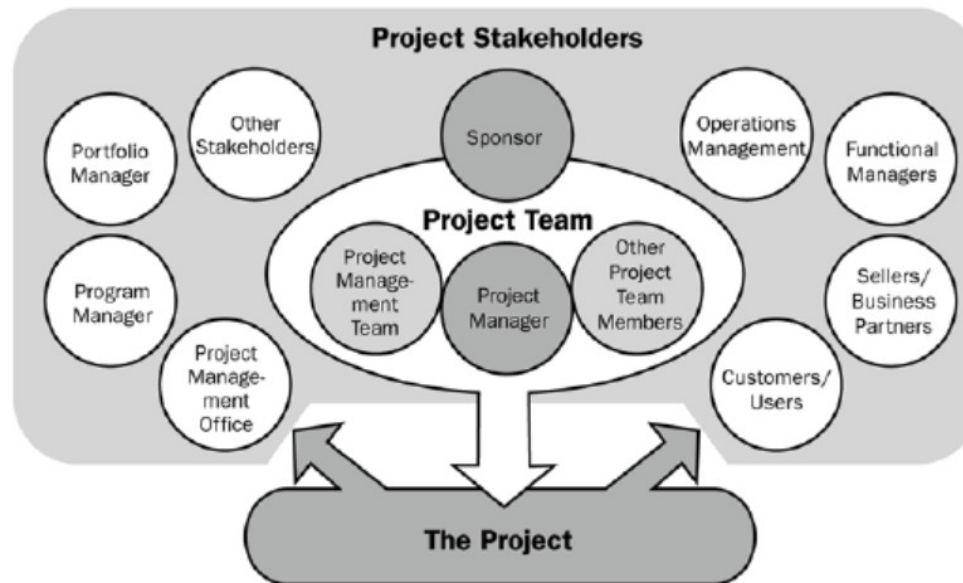
- Asume que una planificación a largo plazo no es realista
- Aunque las metodologías y estándares a cumplir se establezcan al inicio, las especificaciones de más bajo nivel se van realizando conforme avanza el proyecto.
- Requiere que todo el personal esté disponible en varias fases contiguas.
- Ej: RUP (iterativo puro), Scrum (iterativo incremental)

## 2.2 Proyecto Genérico

- Las fases de un proyecto pueden tener distintas relaciones; es decir, un ciclo de vida puede estar compuesto por varios ciclos de vida. Ej:
  - Scrum es iterativo en el sentido de que el primer sprint no se ejecuta hasta que se crea el primer backlog; pero es iterativo ya que se van creando distintos backlogs e informes conforme se realizan sprints.
  - La relación en RUP entre las fases de Inicio y Transición es prácticamente secuencial, y entre el resto es iterativa.
  - XP comienza a hacer tests antes de que el código esté finalizado.
- **La selección de una relación entre fases depende de:**
  - *Nivel de incertidumbre*: ninguna, media o mucha incertidumbre para relación secuencial, superpuesta e iterativa, respectivamente.
  - *Rapidez deseada*: ninguna, media o mucha para relación secuencial, superpuesta e iterativa, respectivamente.
  - *Disponibilidad de todos los recursos a la vez*: ninguna, media o mucha para relación secuencial, superpuesta e iterativa, respectivamente

## 2.3 Organización Estructural

- Independientemente de la metodología aplicada a nuestro proyecto, hay un conjunto de entidades que influyen directamente:
  - El *equipo de proyecto*, que será creado por el PM.
  - Los *interesados*, que serán identificados por el PM.



## 2.3 Organización Estructural

- **El Equipo de Proyecto** debería ser creado y dirigido por el PM.
- El PM es seleccionado por el jefe de departamento, gestor de programa o gestor de portafolio, dependiendo del tipo de empresa.
- **La selección del PM** ya no se realiza entre aquellos que se encuentran *libres*.
- Las **cualidades que el PM debe mostrar** para ser seleccionado son: [Mantel et al. 4Ed]
  - *‘Hard workers are easy to find. What is rare is an individual who is driven to finish the job’.*
  - *Credibilidad*
    - Técnica: dominar varias áreas de conocimiento del proyecto
    - Administrativa: los informes a los superiores deben ser exactos y a tiempo.
  - *Anticipación*: capaz de predecir y solucionar tensiones en el equipo o interesados.
  - *Liderazgo*:
    - Facilitador en lugar de autoritario
    - Flexible cuando hay incertidumbre en el proyecto
    - Sentido de la ética


## 2.3 Organización Estructural

- El PM debe procurar que su **equipo cumpla las siguientes cualidades**:
  - Cada individuo *competente* en su tarea.
  - Deben ser *comprometidos* con el proyecto. Es conveniente que sean personas con voluntad propia para la resolución de problemas inesperados que puedan aparecer, sin necesidad de que el PM lo ordene:
    - Programador enfermo
    - Máquina estropeada
    - Cronograma reajustado
  - *Confianza para ser sinceros*. Tienen que ser capaces de evitar la tentación de ocultar errores, por miedo a afectar al plan del proyecto. El PM debe evitar crear un ambiente hostil que favorezca esas situaciones.


*Si no has trabajado antes con ellos, no sabes si cumplen ciertos requisitos.*

*Por si acaso, ¡enséñales!: entrenamiento, motivación y recompensa*

## 2.3 Organización Estructural

- **Los interesados** pueden ser personas, departamentos o empresas. Su relación con el proyecto es una de estas tres:
  - Participan en el proyecto
  - El éxito del proyecto les afecta positivamente
  - El éxito del proyecto les afecta negativamente (*interesados negativos*) 
- Algunas veces los intereses de los interesados son contradictorios, y el PM deberá saber balancearlos
- Pueden ser interesados:
  - Cliente y usuario final
  - Patrocinador
  - Director de portafolio y programa – Oficina de Gestión de Proyectos
  - Equipo de proyecto

## 2.3 Organización Estructural

- Algunos interesados tendrán *autoridad* sobre el PM, otros *responsabilidad* dentro del proyecto, y otros simplemente *interés*. TODOS DEBEN SER IDENTIFICADOS Y CONTROLADOS.
  - Encuestas al usuario final
  - Reuniones con los analistas
  - Informe de presupuesto al jefe de departamento
  - Departamento legal 

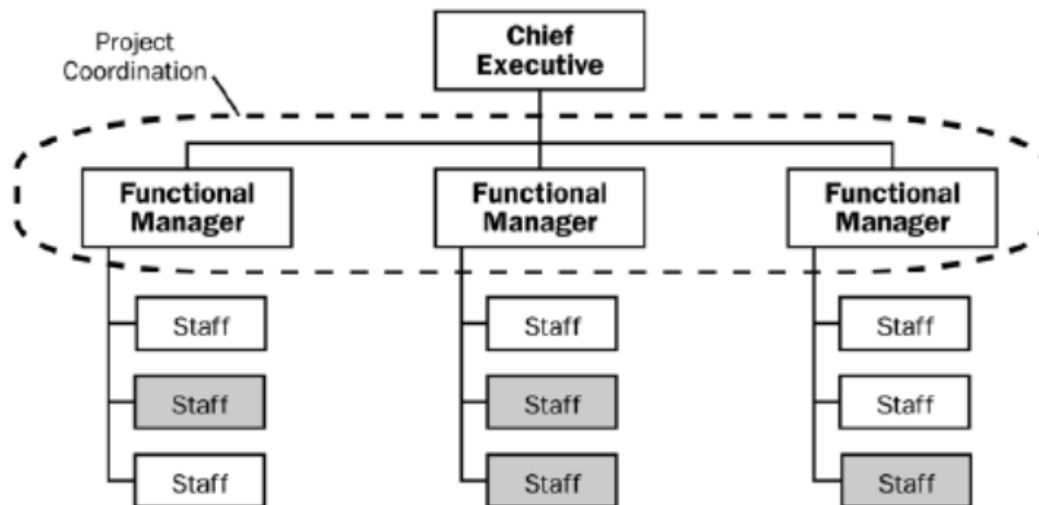
## 2.4 Organización en la Empresa

- Las **influencias** dentro de una empresa **puede variar** mucho **de la oficial a la real**:
  - Culturas únicas en cada empresa:
    - Normas no escritas
    - Tipo de relación correcta con las autoridades
    - Procedimientos a seguir
  - Las autoridades reales no son siempre las oficiales. Como PM, si eres nuevo en al empresa necesitarás identificar rápidamente quién tiene autoridad sobre tu proyecto.
- Aparte de conocer cómo fluyen las influencias, es necesario ser consciente del tipo de **organización de los departamentos**, respecto a la gestión de proyectos  
[PMBOK,4Ed]
  - Funcional
  - Matricial
  - Orientada a proyectos

## 2.4 Organización en la Empresa

- **Empresa Funcional**


- ✓ – Cada empleado tiene un *superior claro*
- ✓ – *Acceso directo* a los requisitos internos.
- ✓ – Resulta *barato* para la empresa ya que el personal es contratado a tiempo parcial o, por lo menos, no están destinados únicamente al desarrollo de proyectos.



(Gray boxes represent staff engaged in project activities.)

## 2.4 Organización en la Empresa

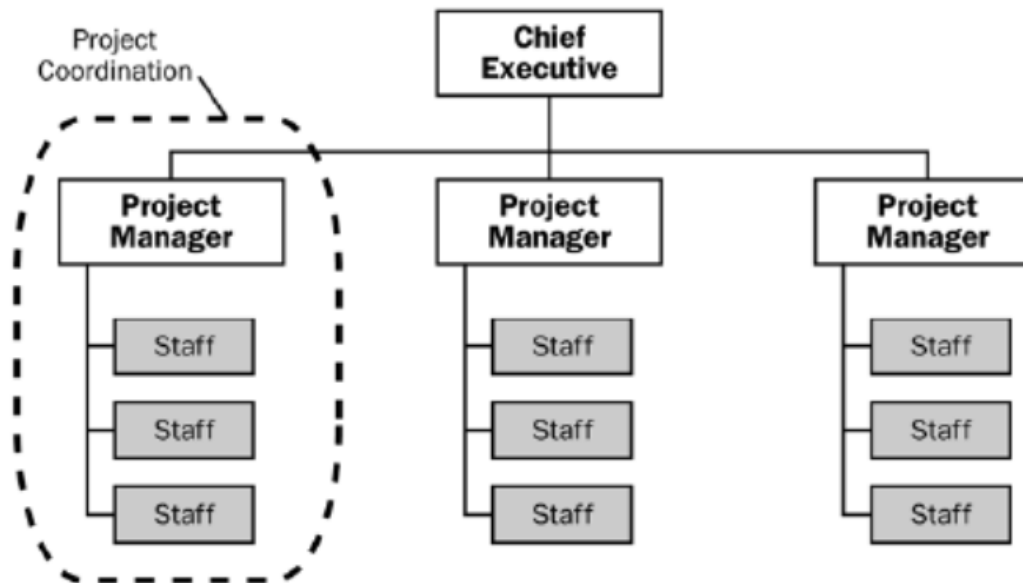
(...funcional)

- ✗ – Cada departamento desarrolla su trabajo correspondiente del proyecto de forma *independiente* a los otros departamentos:
  - Problemas al necesitar asistencia: técnica, programadores, renegociación de presupuesto,...
  - Profundidad tecnológica, pero no extendida.
- ✗ – La *comunicación* interdepartamental es extremadamente lenta, y generalmente ha de pasar por el jefe de departamento:
  - Retrasos en el proyecto
  - Comunicación con el cliente más lenta
- ✗ – El proyecto *no es prioritario* en ninguno de los departamentos en los que se reparte. 

## 2.4 Organización en la Empresa

- **Empresa Orientada a Proyectos**

- ✓ – Los miembros del equipo siguen una *misma distribución* física y lógica
- ✓ – Los *recursos* de la empresa están dirigidos al desarrollo de los proyectos
- ✓ – *Especialista* de cada área dentro del proyecto



(Gray boxes represent staff engaged in project activities.)

## 2.4 Organización en la Empresa

- (...orientada a proyectos)
  - ✗— *Cuando el proyecto acaba*, el equipo vuelve a su ubicación a la espera de otro proyecto:
    - El gestor de presupuesto no es necesario en todo momento
    - Poco trabajo frustra tanto como el exceso
    - Esta organización es útil para grandes proyectos o empresas con tráfico continuo de proyectos
  - ✗— Pocos especialistas y *profundidad tecnológica baja*: en ocasiones es necesario realizar *outsourcing* o intentar salvar la falta:
    - Cuando un especialista asignado no es competente
    - Cuando toda la empresa está envuelta en proyectos hace falta contratar temporalmente recursos humanos externos

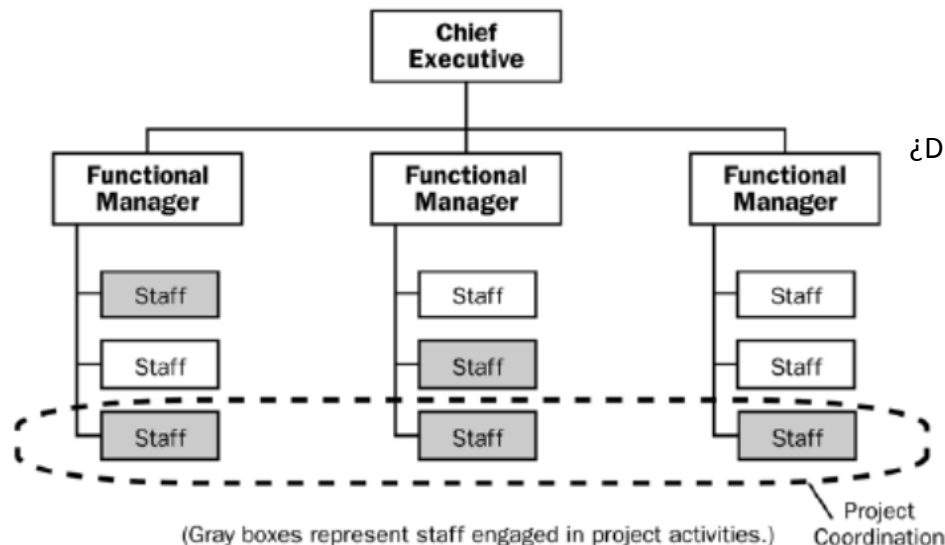
## 2.4 Organización en la Empresa

- **Empresa Matricial: débil, equilibrada y fuerte**

- Distintas maneras de aprovechar las ventajas de las distribuciones funcional y orientada a proyectos.


### Matricial débil

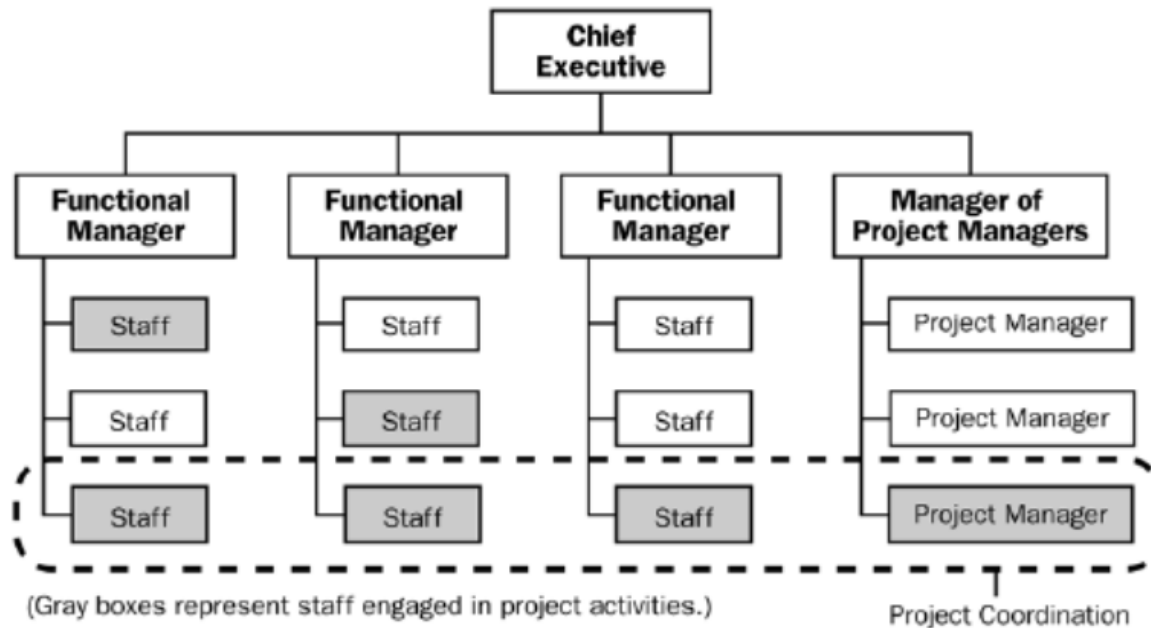
- El PM es seleccionado dentro de un departamento, y es el encargado de coordinar los departamentos para el desarrollo del proyecto. **PM coordina pero no ordena.**
- Por lo demás sigue una organización funcional.



¿Diferencia en la gestión del proyecto entre funcional y matricial débil?

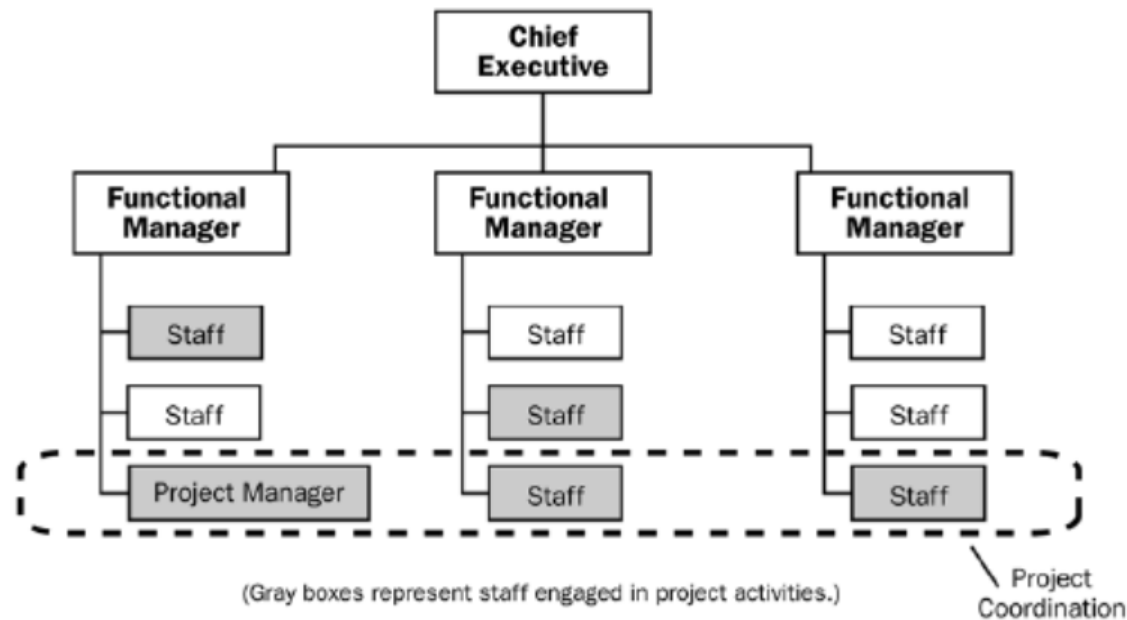
## 2.4 Organización en la Empresa

- Matricial fuerte
  - PM y personal a tiempo completo
  - ¿Cuántos jefes tendrá cada miembro? 



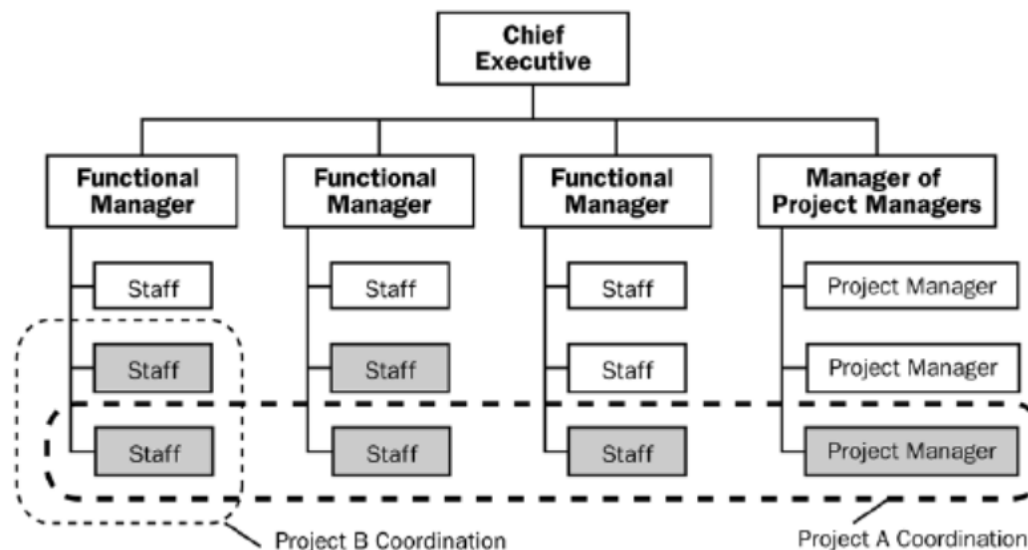
## 2.4 Organización en la Empresa

- Matricial equilibrada
  - El PM tiene autoridad, pero no plena, en especial sobre el presupuesto.
  - El PM y personal no tienen por qué ser a tiempo completo



## 2.4 Organización en la Empresa

- Por supuesto, una empresa tener departamentos con distinta organización.
- Así, puede ser funcional en general, pero tener un departamento dedicado a la gestión de proyectos:
  - Proyectos gestionados de forma funcional o matricial débil.
  - Proyectos gestionados por un miembro del departamento de gestión de proyectos



(Gray boxes represent staff engaged in project activities.)

## 2.4 Organización en la Empresa

### Activos de la empresa

- Algunas **actividades** y **entregables** indicados por la metodología seleccionada para el desarrollo del proyecto ya están definidos por la empresa, y no pueden realizarse con un estilo propio:
  - Plantillas
  - Auditorías tras el proyecto
  - Procedimientos de control de riesgo
  - ...
- Tras finalizar un proyecto, es responsabilidad del PM de añadir a la Base de Conocimiento de la empresa, la información adquirida:
  - Puntos base para medir el esfuerzo humano
  - Puntos base para la estimación del coste
  - Lecciones aprendidas:
    - Personas clave para el éxito
    - Clientes o personal con los que no conviene trabajar

## 2.4 Organización en la Empresa

- La **madurez** se ve fuertemente influenciada por la organización de la empresa. Los modelos de madurez:
  - Consideran positivo una organización orientada a proyectos
  - No tienen en cuenta que para algunas empresas ese elevado coste es inapropiado
  - PMMM (Project Management Maturity Model) es un modelo de madurez que incluye:
    - CMM
    - Áreas del Conocimiento identificadas por el PMI

En un nivel de 5 puntos, la mayoría de las empresas no superan el 2 en madurez, solo empresas bien conocidas.



*¿Lo vas a tener en cuenta cuando seas  
Jefe de Proyecto?*