UT2-IMPLANTACIÓN,INSTALACIÓN Y CONFIGURACIÓN DE UN SGE

SGE

Rosa María Zapata Calle

PARTE 1-Índice

- Licencias de Software
- Licencias de Software propietarias
- Programa de licencias abiertas (OLP)
- Licencias de Software libres o de código abierto
- General Public License (GPL)

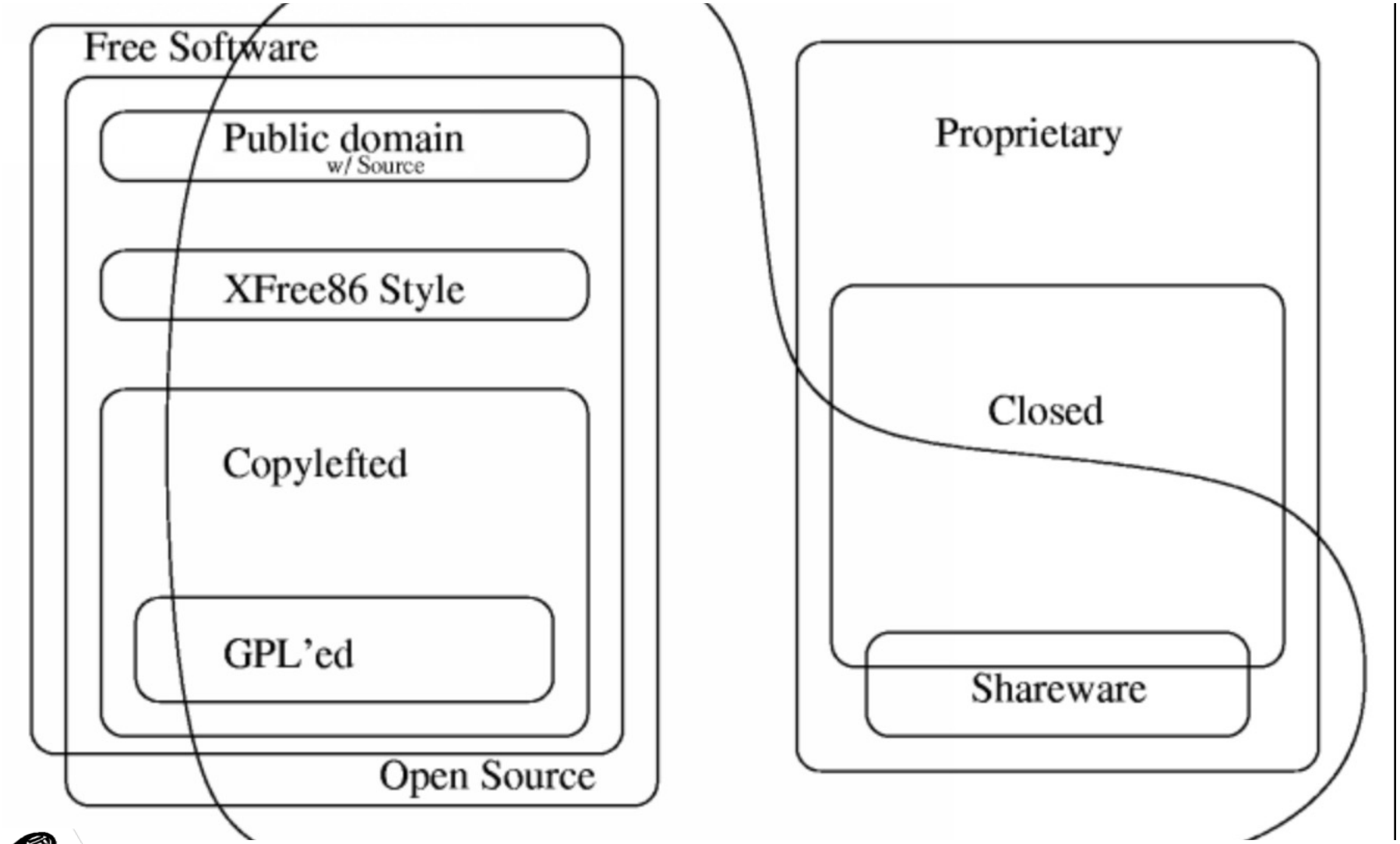
Índice

- Licencias de Software libre permisivas
- Free Software Foundation
- Software libre (Free Software)
- Definición y las cuatro libertades.
- Iniciativa código abierto
- Free Software foundation vs open source initiative (código abierto)
- Licencias GNU
- Infraestructuras

Licencias Software

- Una licencia software es un instrumento legal (normalmente es un contrato legal, con o sin material impreso) que controla el uso o redistribución del software.
- Las licencias de Software puede generalmente incluirse dentro de las siguientes categorías:
  - Licencias propietarios.
  - Licencias libres y de Código Abierto

Licencias Software



Licencias de Software Propietarias/privativas

- Se conoce como Software privativo a aquél cuyo uso, redistribución o modificación está prohibido o necesita una autorización.

- Este aspecto de licencia de software privativo significa que ciertos derechos del software están reservados para el distribuidor. No obstante, es típico en ellos incluir términos que definen el uso del software, como el número de instalaciones permitidas o términos de distribución.

Licencias Software Propietario/privativo

- El efecto más significativo de este tipo de licencias es el hecho de que si el propietario del software mantiene con el distribuidor este tipo de licencia, el usuario final debe aceptar la licencia de software. En otras palabras, sin aceptar la licencia, el usuario final no puede usar el software.
  - Un ejemplo de licencia de software propietario es Microsoft Windows.

Licencias de Software Propietario.Microsoft

Como ejemplo de licencias privativas o propietarias, veremos el caso de Microsoft.

Basicamente, existen tres tipos de licencias en Microsoft:

- Licencias **OEM**: Licencia que viene incluida en los equipos cuando se compran.
- Licencias **Retail** (también llamadas RTL o FPP) :Licencias obtenidas mediante la web, útiles para usuarios que requieren menos de 5 licencias.
- Licencias por **Volumen** (también llamadas GVLK-Global Volumen license key): útiles para grandes empresas, centros educativos,etc.

Ejemplo Licencias Microsoft

HACER EJERCICIO 1

- Visita la siguiente página web:

https://www.solvetic.com/page/articulos/s/profesionales/tipos-de-licencias-windows-oem-retail-volumen

-Realiza un esquema con un resumen de las características, ventajas y desventajas de cada tipo de licencia. Este esquema se incluirá en los apuntes del módulo y serán evaluables.

Licencias Software libres o de código abierto

- Las licencias libres o de código abierto generalmente se incluyen en dos categorías:
  - Aquellas cuyo objetivo es tener los mínimos requisitos sobre como el software puede redistribuirse (licencias permisivas).
  - Aquellas cuyo objetivo preserva las libertades que son dadas.

General Public License (GPL)

- Un ejemplo de una licencia de software libre **copyleft es la Licencia GNU (General Public License :GPL)**.
- Esta licencia tiene como objetivo dar a todos los usuarios libertad indefinida en el uso, estudio y modificación del software y si los usuarios se adhieren a los términos y condiciones GPL, libertad para redistribuir el software o cualquier modificación del mismo.
- Por ejemplo, algunas modificaciones hechas y redistribuidas por el usuario final debe incluir el código fuente de estas, y la licencia de cualquier trabajo derivado no puede incluir ninguna restricción adicional sobre lo que la licencia GPL permite.

LICENCIAS DE SOFTWARE LIBRE PERMISIVAS

- Ejemplos de licencias software libres permisivas son:
  - **La licencia BSD**

    Las cuales dan un ilimitado permiso de uso, estudio y modificación privada del software, e incluye solo unos requisitos mínimos de redistribución.
  - **La licencia MIT**

    Esta licencia de Software libre permisiva impone muy pocas limitaciones en la reutilización y por tanto posee una excelente Compatibilidad de licencia. La licencia MIT permite reutilizar software dentro de un Software propietario.

Free Software Foundation

- La fundación por el Software libre o Free Software Foundation (FSF) se dedica a eliminar las restricciones sobre la copia, redistribución, entendimiento y modificación de programas de computadoras. Con este objeto promociona el desarrollo y uso del Software libre en todas las áreas de la computación, pero muy particularmente, ayudando a desarrollar el sistema operativo GNU.
- La FSF elabora, mantiene y defiende la Licencia pública General GNU (GNU GPL).

Free Software Foundation

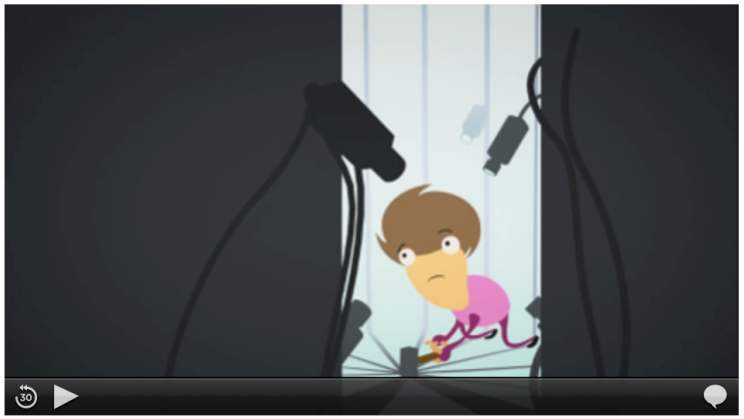
FREE SOFTWARE FOUNDATION

Log in Help! Members forum JOIN FSF NOW

about campaigns licensing membership resources community donate shop

The Free Software Foundation (FSF) is a nonprofit with a worldwide mission to promote computer user freedom and to defend the rights of all free software users. Read more.

We're hiring! Join us as a deputy director, Web developer, or outreach & communication coordinator.



Download links and credits

Embed on your site or blog: <iframe width="640" height="390" src="http://static.fsf.org/nosvn/FSF30-video/FSF_30_720p.webm" frameborder="0" allowfullscreen></iframe>

Free software developers guarantee everyone equal rights to their programs; any user can study the source code, modify it, and share the program. By contrast, most software carries fine print that denies users these basic rights, leaving them susceptible to the whims of its owners and vulnerable to surveillance.

- The FSF provides critical infrastructure and funding for the GNU project, the foundation of the popular GNU/Linux family of free operating systems and the keystone of the Internet.
- Our Campaigns Team creates educational materials about free software, convenes the yearly LibrePlanet conference and goes toe to toe against powerful interests that threaten computer user rights.
- Our Licensing & Compliance Lab defends freely licensed software from proprietary hoarding, advises on licensing issues, and certifies devices that Respect Your Freedom.

With your support, we've done these things for almost 30 years. Help launch us into 30 more; please become a member today.

Software libre

- El Software libre es un software que da la libertad a los usuarios de ejecutar el software para cualquier propósito como también estudiarlo, modificarlo y distribuir el software original y las versiones adaptadas a partir de él. Los derechos de estudio y modificación del software implica que el código fuente del resultado también será accesible por otros usuarios

Definición y las cuatro libertades

- Para que un software pertenezca al software libre deben cumplirse las 4 libertades siguientes:
  - **Libertad 0**: Libertad de ejecutar el programa cuando se desea.
  - **Libertad 1:** Libertad de estudiar cómo funciona el programa y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.
  - **Libertad 2**:libertad de redistribuir copias para ayudar a su prójimo.
  - **Libertad 3**: libertad de redistribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones.
- La libertad 1 y 3 requiere que el código fuente esté disponible para ser estudiado y modificado.

Iniciativa de Código Abierto

Iniciativa de Software libre

http://opensource.org

- **Open Source** (Código abierto) es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones éticas y morales las cuales destacan en el llamado software libre

Definición de código abierto

- La idea del código abierto se centra en la premisa de que al compartir el código, el programa resultante tiende a ser de calidad superior al software propietario, es una visión técnica. Por otro lado, el software libre tiene tendencias filosóficas e incluso morales: el software propietario, al no poder compartirse, es <<antiético>> dado que prohibir compartir entre seres humanos va en contra del sentido común. Ninguna adaptación ni cambios que no haya realizado previamente la empresa fabricante.
- El código abierto ofrece:

↔ **Acceso al código fuente**: Para modificarlo, corregirlo o añadir más prestaciones.

↕ **Gratuidad**: El software puede obtenerse libremente.

⇠ **La posibilidad de evitar monopolios de software propietario**: Para no depender de un único fabricante de software.

⇢ **Un modelo de avance**: Por lo cual la información no se oculta.

Decálogo Código abierto

Al igual que el software libre, el código abierto tiene una serie de requisitos necesarios para que un programa pueda considerarse dentro de este movimiento, estos son:

- **Libre redistribución**: el software debe poder ser regalado o vendido libremente.
- **Código fuente**: el código fuente debe estar incluido u obtenerse libremente.
- **Trabajos derivados**: la redistribución de modificaciones debe estar permitida.
- **Integridad del código fuente del autor**: las licencias pueden requerir que las modificaciones sean redistribuidas solo como parches.
- **La licencia no debe discriminar a ninguna persona o grupo**: nadie puede dejarse fuera.

Decálogo Código abierto

- **Sin discriminación de áreas de iniciativa**: los usuarios comerciales no pueden ser excluidos.
- **Distribución de la licencia**: deben aplicarse los mismos derechos a todo el que reciba el programa
- **La licencia no debe ser específica de un producto**: el programa no puede licenciarse solo como parte de una distribución mayor.
- **La licencia no debe restringir otro software**: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
- **La licencia debe ser tecnológicamente neutral**: no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

Este decálogo es compatible con las cuatro libertades del software libre.

GNU Licenses

El siguiente enlace nos muestra la mejor forma de aprender sobre las licencias GNU.

- https://www.gnu.org/licenses/gpl-faq.es.html

Ejercicios

- Investiga sobre las diferentes licencias de sistemas Software ERP que hemos visto en la unidad uno.
- Deberás usar una tabla para ser más claro.

Insfraestructura de sistemas

- Las tres opciones más importantes son:
  - on-premise ERP.
  - in-house ERP
  - cloud ERP .

Insfraestructura de sistemas

- Tradicionalmente, on-premise e in-house ERP proporciona la mayor flexibilidad de adaptación de paquetes que las necesidades organizacionales de las corporaciones tienen, pero es costoso.

- Las características de las soluciones CLOUD ERP con herramientas de desarrollo robustas emergen rápidamente, pero en el modelo de solución basado en cloud, se debe estar seguro de encontrar las necesidades buscadas y cerciorarse de que tus datos están a salvo.

On-Premise ERP System

- Sistemas on-premise son los instalados y ejecutados en los computadores dentro de la empresa (en el edificio) de la persona y/o organización.

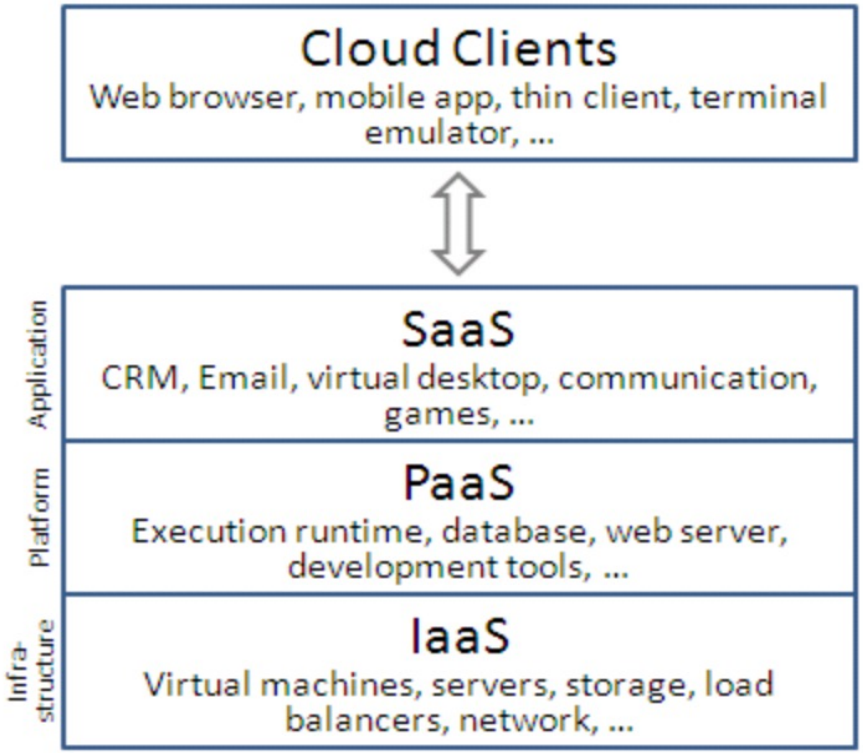
Sistemas ERP CLOUD

Sistema de almacenamiento y uso de recursos informáticos basado en el servicio en red, consistente en ofrecer al usuario un espacio virtual, generalmente en Internet, en el que puede disponer de las versiones más actualizadas de hardware y software.

Existen tres tipos:

- Infraestructura como servicio (IaaS , de Infraestructure as a Service)
- Plataforma como servicio (PaaS , de Platform as a Service)
- 3 . Software como servicio (SaaS , de Software as a Service)

Sistema ERP Cloud



Infraestructura como servicio (IaaS)

- 1 . Infraestructura como servicio (IaaS , de Infraestructure as a Service) , en el que el usuario contrata únicamente las infraestructuras tecnológicas (capacidad de proceso , de almacenamiento y / o de comunicaciones) , sobre las que instala sus plataformas (sistemas operativos) y aplicaciones . El usuario tiene el control total sobre las plataformas y aplicaciones , pero no tiene ningún control sobre las infraestructuras .

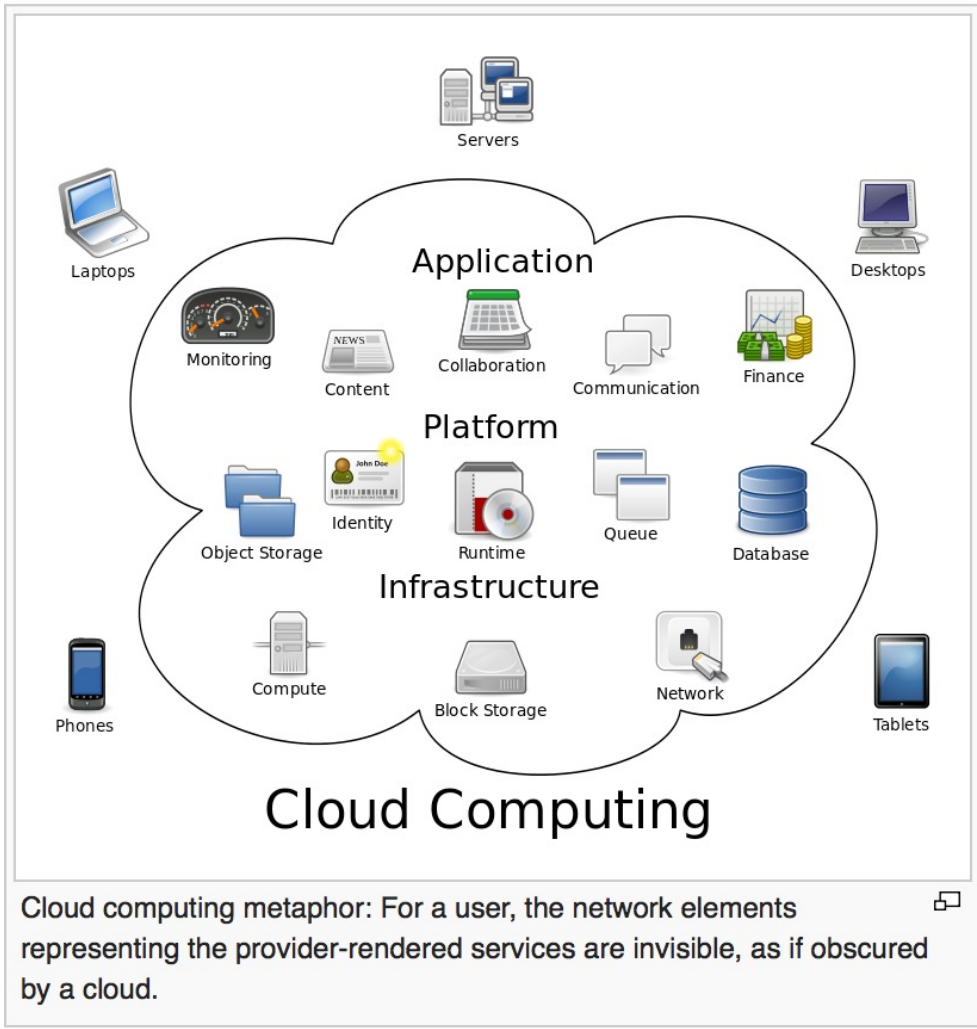
Plataforma como servicio (PaaS)

- 2 . Plataforma como servicio (PaaS , de Platform as a Service) , en el que el usuario contrata un servicio que le permite alojar y desarrollar sus propias aplicaciones (ya sean desarrollos propios o licencias adquiridas) en una plataforma que dispone de herramientas de desarrollo para que el usuario pueda elaborar una solución , en este modelo , el proveedor ofrece el uso de su plataforma que a su vez se encuentra alojada en infraestructuras (de su propiedad o ajena) . El usuario no tiene ningún control sobre la plataforma ni sobre la infraestructura pero mantiene el control total sobre las sus aplicaciones .

Software como servicio (SaaS)

3 . Software como servicio (SaaS , de Software as a Service) , en el que el usuario contrata la utilización de unas determinadas aplicaciones , sobre las que únicamente puede ejercer acciones de configuración y parametrización permitidas por el proveedor . El usuario no tiene c ontrol sobre la aplicación ni sobre la plataforma ni sobre la infraestructura .

Sistemas ERP CLOUD

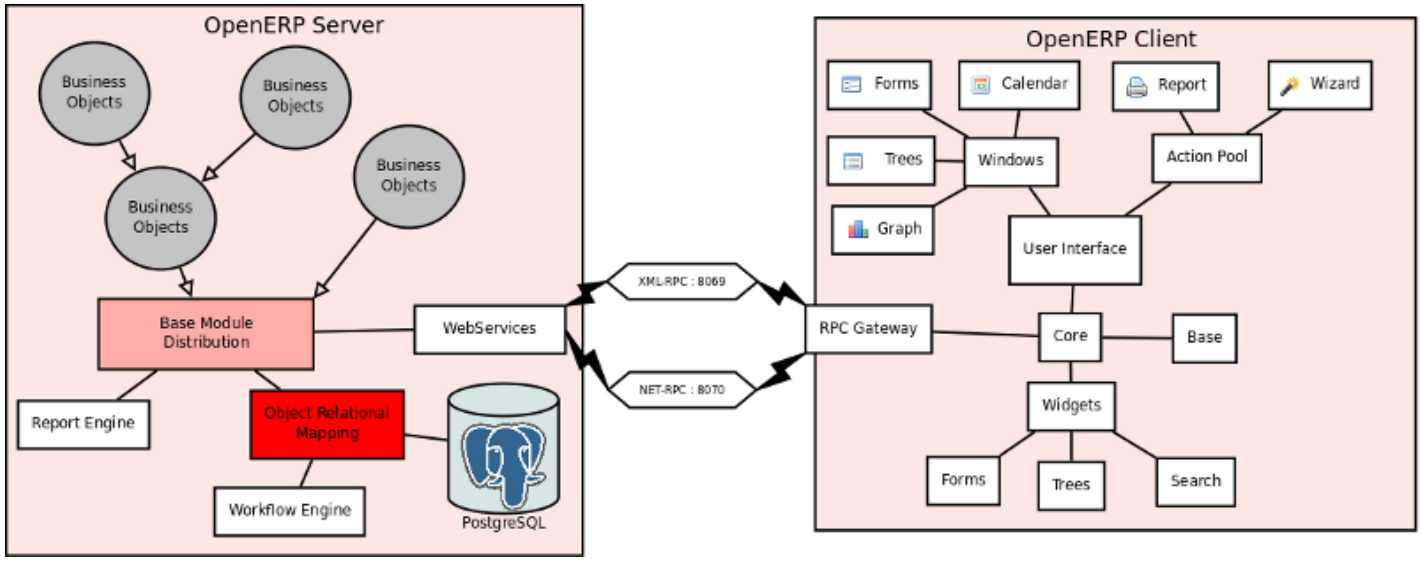


Cloud computing metaphor: For a user, the network elements representing the provider-rendered services are invisible, as if obscured by a cloud.

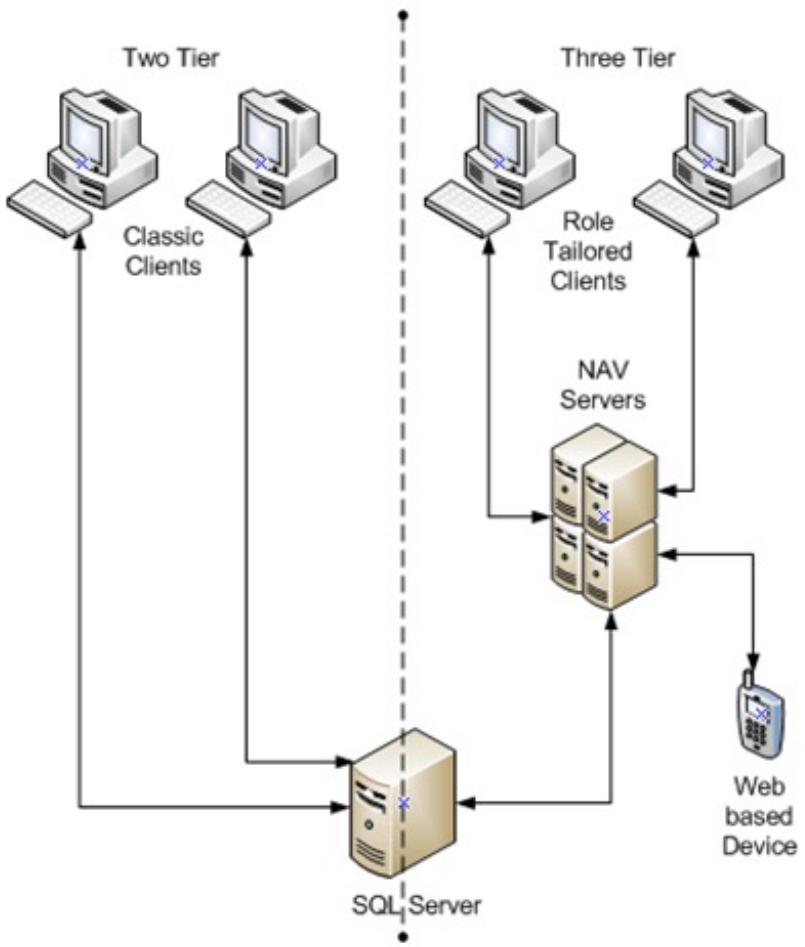
Instalación Sistemas ERP

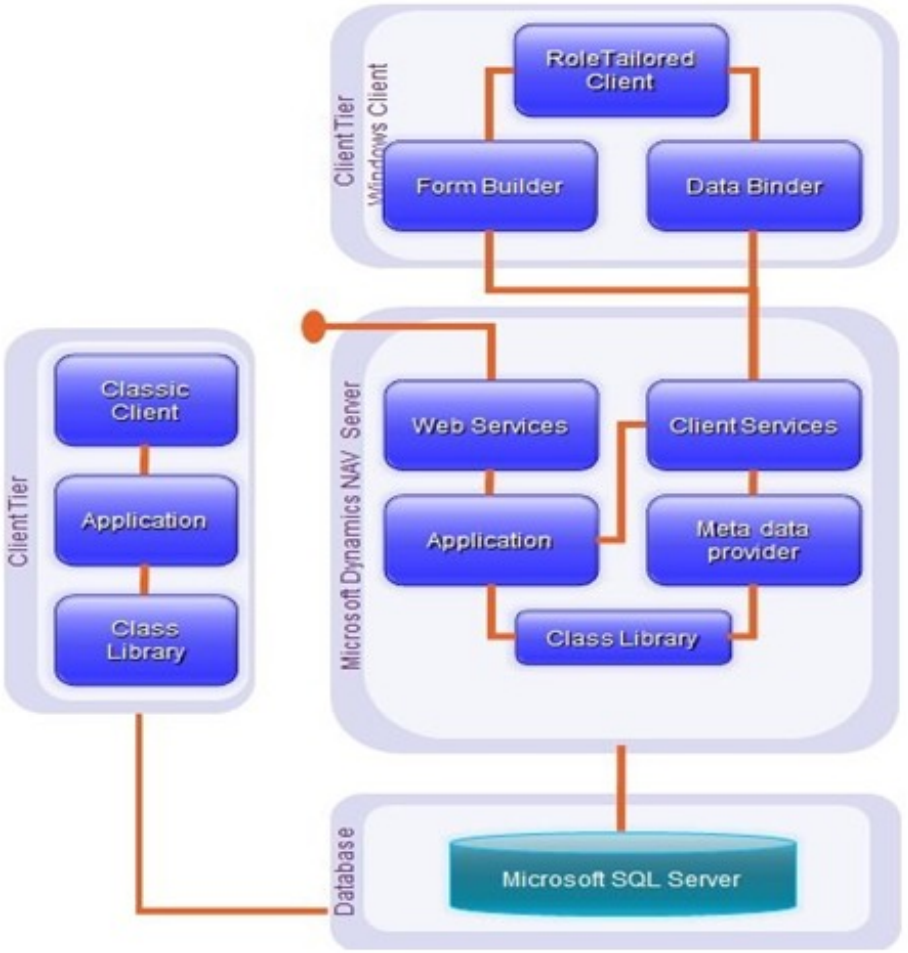
- Ahora, vamos a instalar el paquete SAGE.

Estructura ODOO

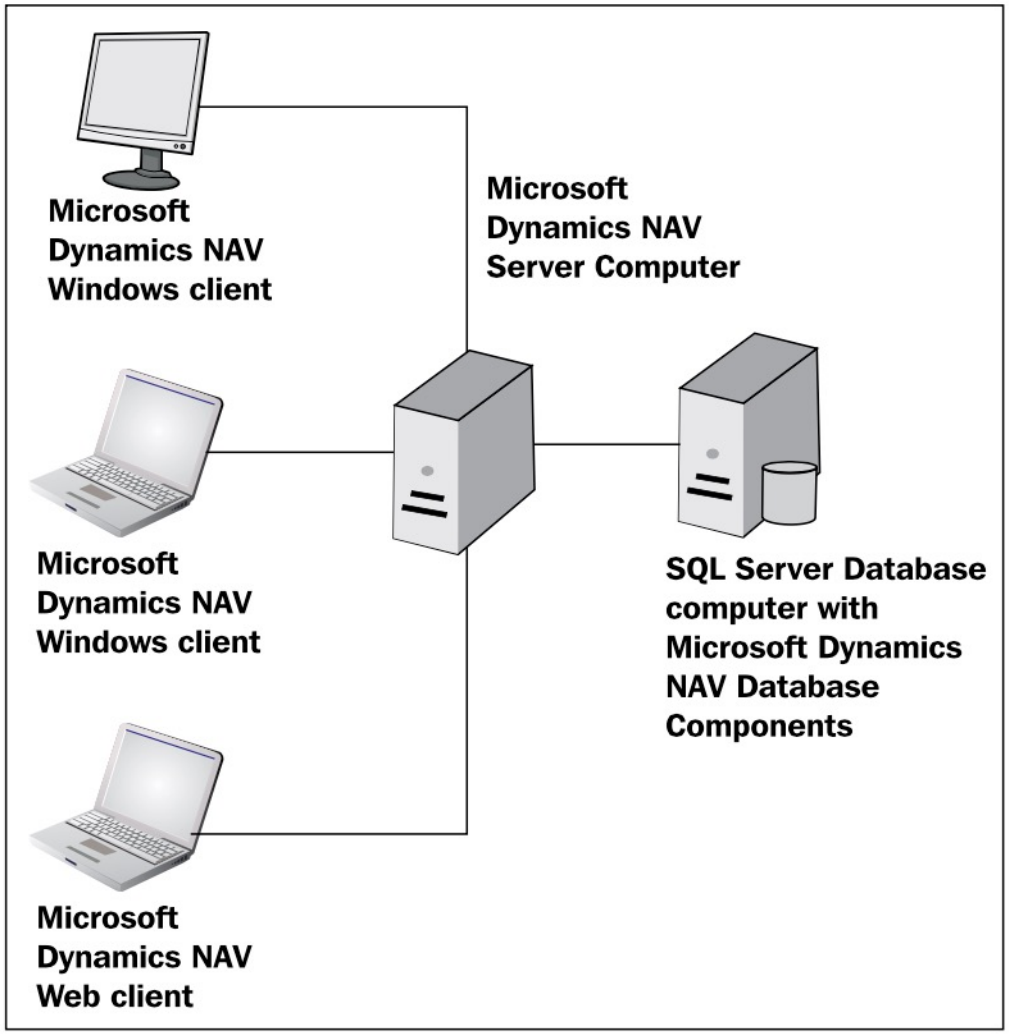


Dynamic Nav


Two Tier
Three Tier
Classic Clients
Role Tailored Clients
NAV Servers
Web based Device
SQL Server


Client Tier
Windows Client
RoleTailored Client
Form Builder
Data Binder
Client Tier
Classic Client
Application
Class Library
Microsoft Dynamics NAV Server
Web Services
Client Services
Application
Meta data provider
Class Library
Database
Microsoft SQL Server

Dynamic Nav


Microsoft
Dynamics NAV
Windows client
Microsoft
Dynamics NAV
Server Computer
Microsoft
Dynamics NAV
Windows client
SQL Server Database
computer with
Microsoft Dynamics
NAV Database
Components
Microsoft
Dynamics NAV
Web client

Free Software foundation vs open source initiative

EJERCICIO 2:

¿Cuál es la diferencia entre la Fundación de Software libre y la iniciativa de código abierto?

Lee el siguiente articulo, escrito por Richard Stallman:

https://www.gnu.org/philosophy/open-source-misses-the-point.es.html

Investiga sobre el tema y describe las diferencias y errores conceptuales entre ambos, según su autor.

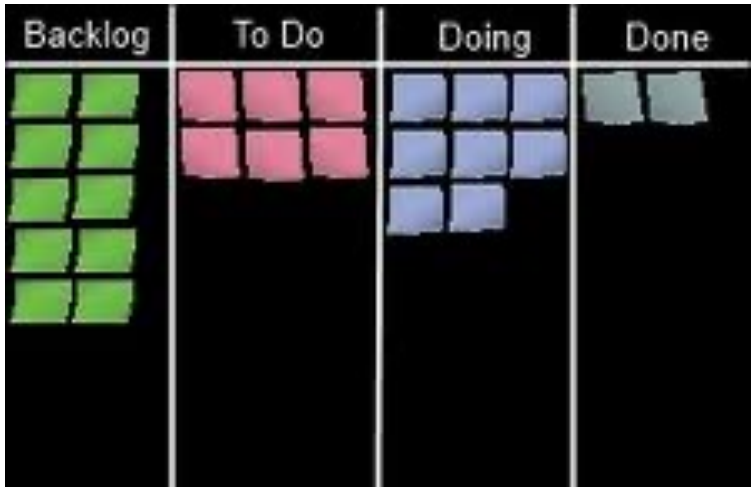
INDICE

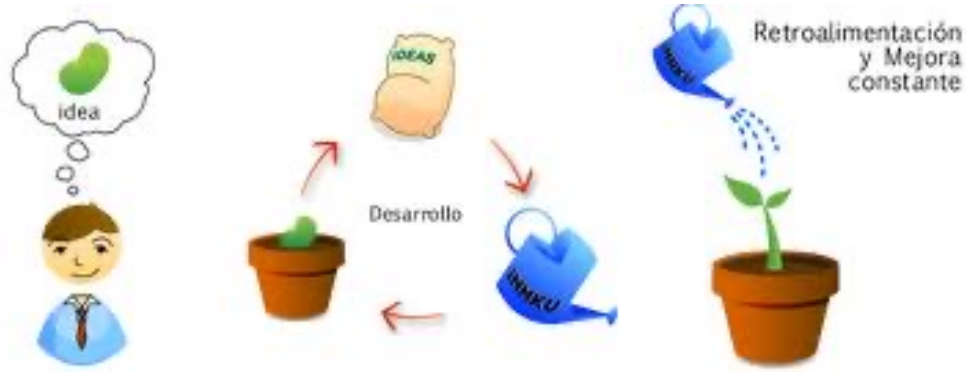
1.GESTOR DE PROYECTOS

El paso preliminar a cualquier desarrollo de un proyecto y más en los proyectos Software, es la elección del gestor de proyectos que el equipo de desarrollo en conjunto con el jefe del mismo van a utilizar para el desarrollo completo del proyecto.
Anteriormente, el desarrollo de los proyectos Software pasaban por metodologías rígidas, especialmente ante los cambios.
Para mejorar estos aspectos, surgieron las metodologías ágiles. Entre las más importantes, es el gestor de proyectos SCRUM.

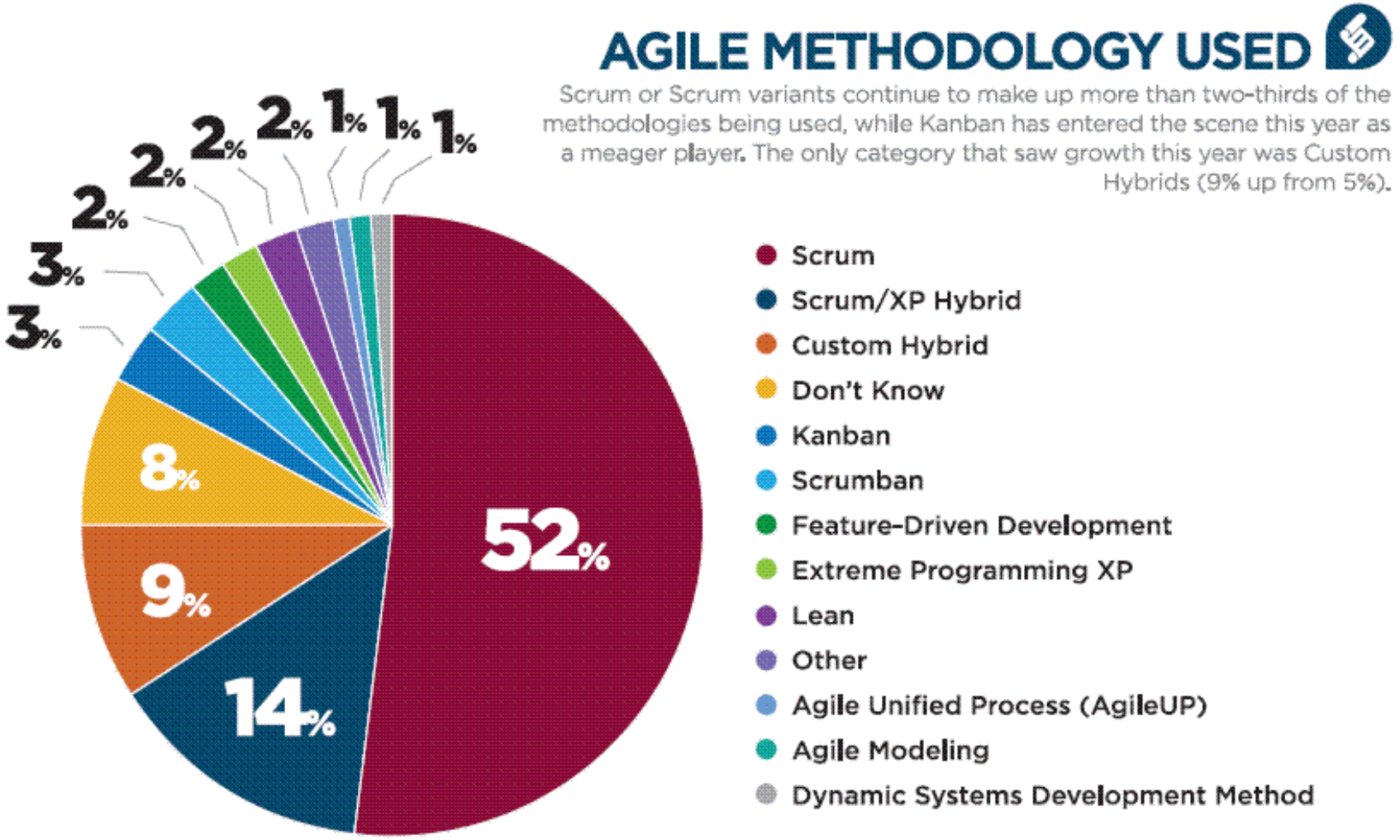
Busca información sobre **SCRUM**

METODOLOGíAS ÁGILES EN LA IMPLANTACIÓN DE UN SGE





METODOLOGíAS ÁGILES EN LA IMPLANTACIÓN DE UN SGE



Métodologías ágiles frente a metodologías tradicionales

Metodología tradicional	Metodología ágil
Planes rígidos	Flexibilidad ante el cambio
Negociación contractual	Colaboración con el cliente
Se valoran los procesos	Se valoran las personas

METODOLOGíAS ÁGILES EN LA IMPLANTACIÓN DE UN SGE

- Principales principios “ágiles” para sistemas de gestión empresarial
  - Asumir simplicidad
  - Acoger cambios con naturalidad
  - Realizar los cambios incrementalmente
  - Valorar el esfuerzo de los participantes en el proyecto
  - Plantear objetivos en las tareas de gestión
  - Aceptar las múltiples vistas del proyecto y consensuar las soluciones
  - Buscar facilidades para que la retroalimentación sea ágil
  - Tener presente la filosofía de working progress

METODOLOGíAS ÁGILES EN LA IMPLANTACIÓN DE UN SGE

SCRUM

Introducción al modelo

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la

adaptación continua a las circunstancias de la evolución del proyecto.

Scrum es una metodología ágil, y como tal:

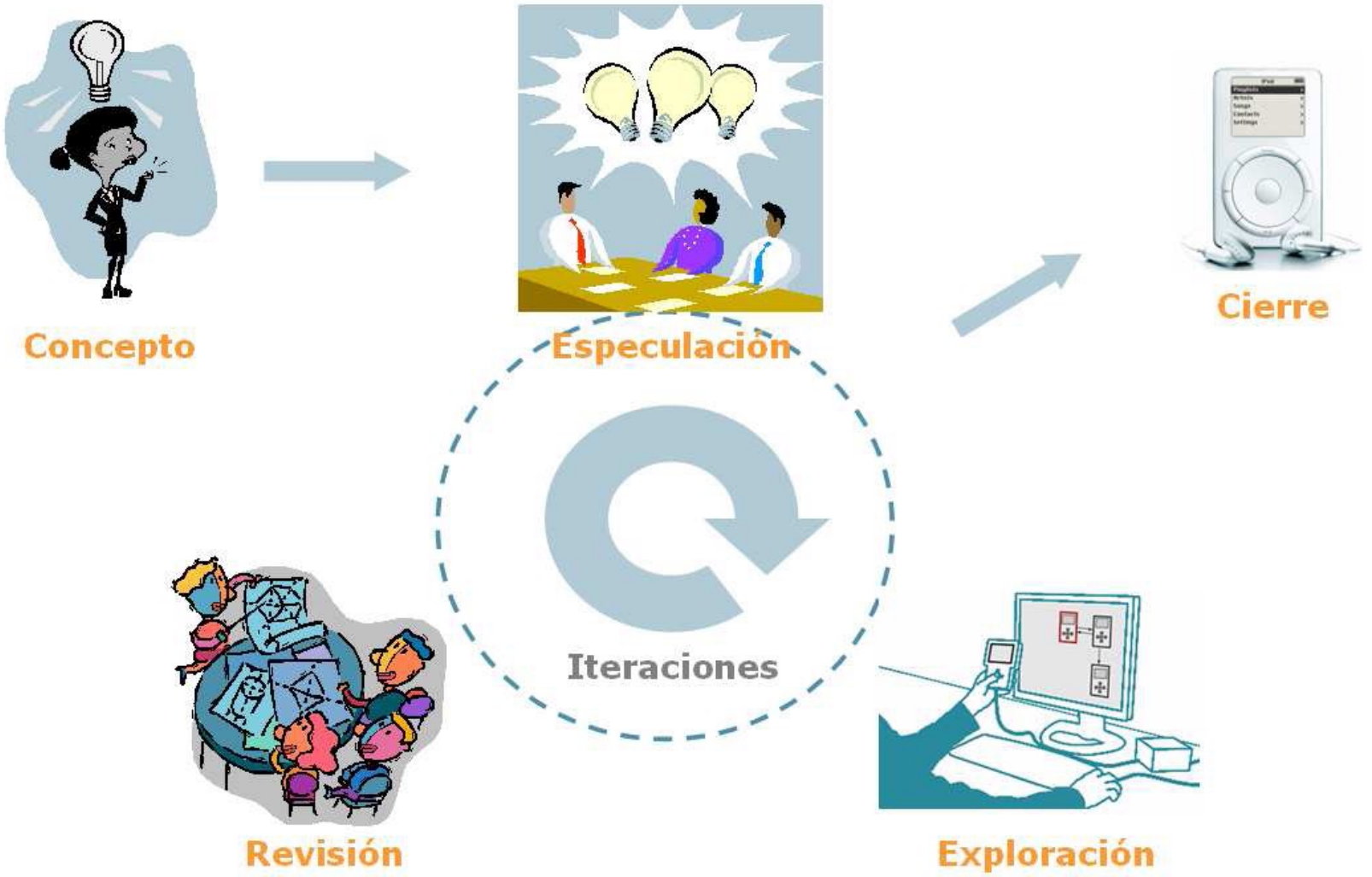
Es un modo de desarrollo de carácter adaptable más que predictivo.

Orientado a las personas más que a los procesos.

Emplea la estructura de desarrollo ágil:

incremental basada en iteraciones y revisiones.

Introducción al modelo



Introducción al modelo

Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve normalmente de 30 días).

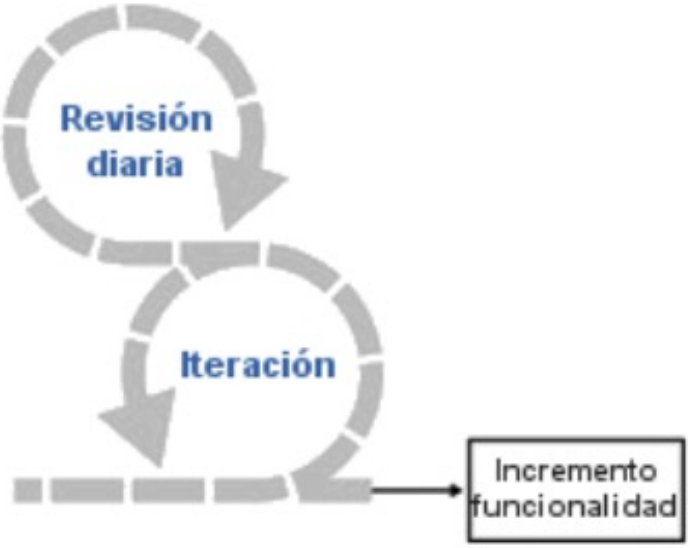
Cada uno de estos periodos de desarrollo **es una iteración que finaliza con la producción de un incremento operativo del producto**.

Introducción al modelo

Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de

reuniones breves diarias en las que todo el

equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.



Control de la evolución del proyecto

Scrum controla de forma empírica y adaptable la evolución del proyecto, empleando las siguientes prácticas de la gestión ágil:

- Revisión de las iteraciones
- Desarrollo incremental
- Desarrollo evolutivo
- Auto-organización
- Colaboración

Prácticas de la gestión ágil

- Revisión de las iteraciones

Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.

Prácticas de la gestión ágil

- **Desarrollo incremental**

Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones.
El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

Prácticas de la gestión ágil

Desarrollo evolutivo

Los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos.

Intentar predecir en las fases iniciales cómo será el producto final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.

Prácticas de la gestión ágil

Para qué predecir los estados finales de la arquitectura o del diseño si van a estar cambiando. En Scrum se toma a la inestabilidad como una premisa, y se adoptan técnicas de trabajo para permitir esa evolución sin degradar la calidad de la arquitectura que se irá generando durante el desarrollo.

El desarrollo Scrum va generando el diseño y la arquitectura final de forma evolutiva durante todo el proyecto. No los considera como productos que deban realizarse en la primera "fase" del proyecto.

(El desarrollo ágil no es un desarrollo en fases)

Prácticas de la gestión ágil

Auto-organización

Durante el desarrollo de un proyecto son muchos los factores impredecibles que surgen en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.

En Scrum los equipos son auto-organizados (no auto-dirigidos), con margen de decisión suficiente para tomar las decisiones que consideren oportunas.

Prácticas de la gestión ágil

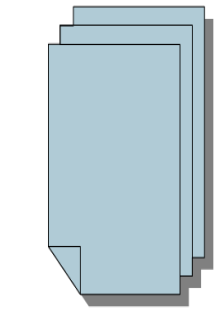
Colaboración

Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto-organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

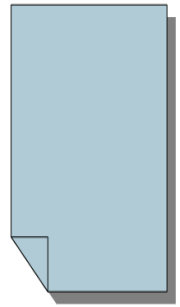
Elementos del desarrollo de Scrum

Los elementos de Scrum son:

- **Pila del producto:** lista de requisitos de usuario que se origina con la visión inicial del producto y va creciendo y evolucionando durante el desarrollo.
- **Pila del sprint:** Lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.
- **Incremento:** Resultado de cada sprint



Pila del producto



Pila del sprint

Incremento

Elementos del desarrollo de Scrum

Los roles

Scrum clasifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto en:

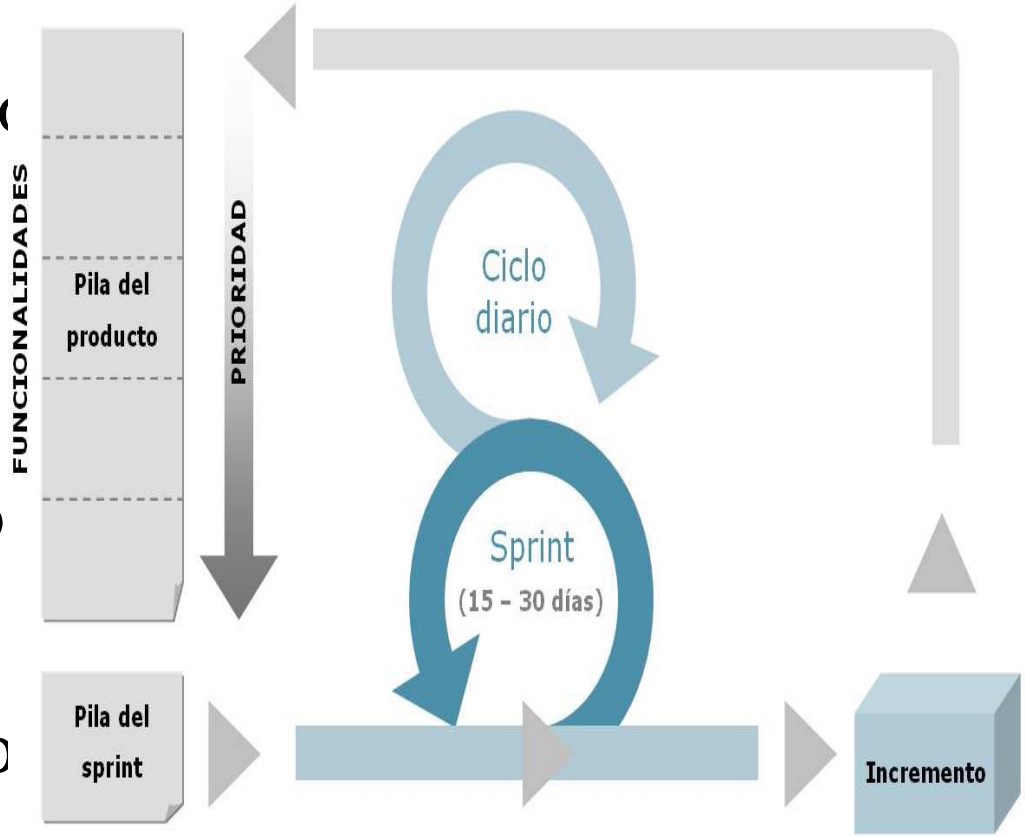
propietario del producto, equipo, gestor de Scrum (también Scrum Manager o Scrum Master) y “otros interesados”.

Los tres primeros grupos (propietario, equipo y gestor) son los responsables del proyecto.

mientras que el resto de interesados.

Visión general del proceso

- Scrum denomina "sprint" a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días, por ejemplo.
- El sprint es por tanto el núcleo central que proporciona la base de desarrollo iterativo e incremental.



Elementos del desarrollo de Scrum

- Los elementos que conforman el desarrollo
- Scrum son:
- Las reuniones.
- Los elementos
- Los roles

Elementos del desarrollo de Scrum

Las reuniones

Planificación de sprint:

Jornada de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en esa iteración.

Reunión diaria:

Breve revisión del equipo del trabajo realizado hasta la fecha y la previsión para el día siguiente.

Revisión de sprint:

Análisis y revisión del incremento generado.

Elementos del desarrollo de Scrum

Cerdos y gallinas.

Esta metáfora ilustra de forma muy gráfica la diferencia de implicación en el proyecto entre ambos grupos:

Una gallina y un cerdo paseaban por la carretera.

La gallina dijo al cerdo: “Quieres abrir un restaurante conmigo”.

El cerdo consideró la propuesta y respondió: “Sí, me gustaría. ¿Y cómo lo llamaríamos?”.

La gallina respondió: “Huevos con beicon”. El cerdo se detuvo, hizo una pausa y contestó:

“Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada”.

Elementos del desarrollo de Scrum

COMPROMETIDOS (cerdos)	IMPLICADOS(gallinas)
Propiet. del producto Equipo Scrum Manager	Otros interesados (Dirección general Dirección comercial Marketing Usuarios, etc)

Propietario del producto: El responsable de obtener el mayor valor de producto para los clientes, usuarios y resto de implicados.

Equipo de desarrollo: grupo o grupos de trabajo que desarrollan el producto.

Scrum Manager: gestor de los equipos que es responsable del funcionamiento de la
metodología Scrum y de la productividad del equipo de desarrollo.

VALORES

Scrum es una "carrocería" para dar forma a los principios ágiles. Es una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil:

Cristal, DSDM, etc.

La carrocería sin motor, sin los valores que dan sentido al desarrollo ágil, no funciona.

Delegación de atribuciones (*empowerment) al* equipo para que pueda auto-organizarse y tomar las decisiones sobre el desarrollo.

Respeto entre las personas. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.

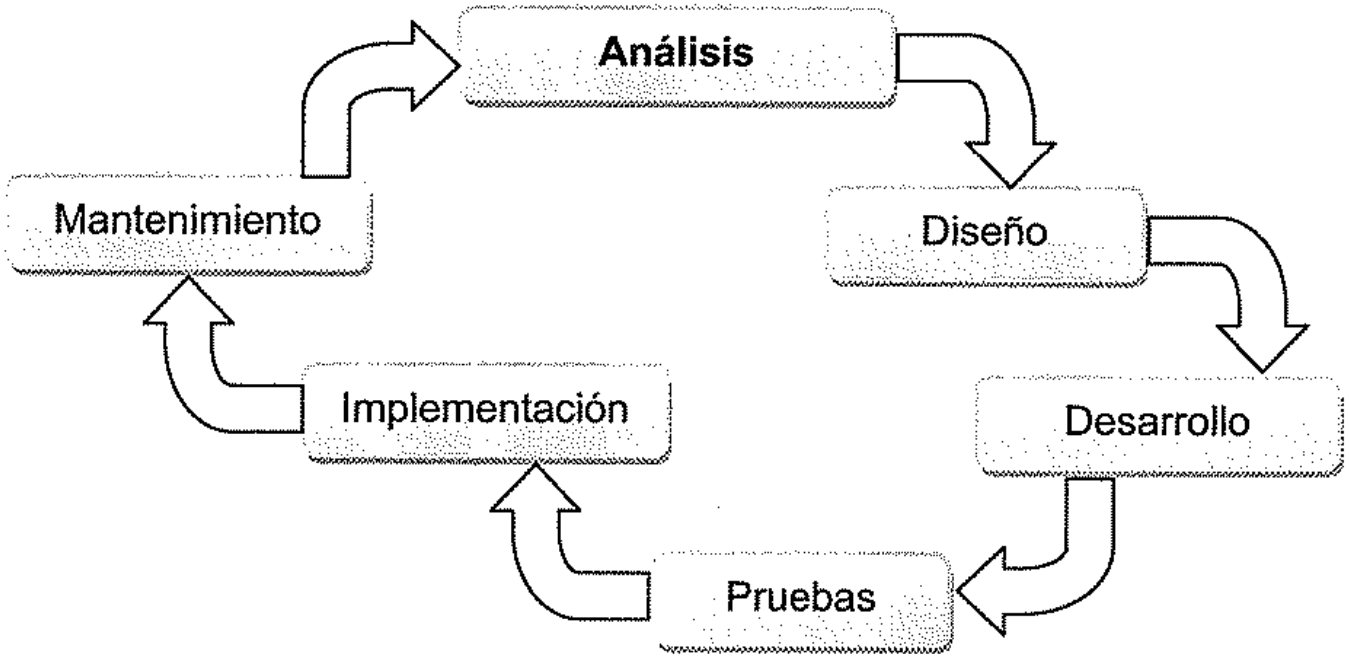
Responsabilidad y auto-disciplina (no disciplina impuesta).

Trabajo centrado en el desarrollo de lo comprometido .
Información, transparencia y visibilidad del desarrollo del proyecto

2.IMPLANTACIÓN DE UN ERP

- Una vez decidido marco de trabajo, que podría ser SCRUM, por los beneficios que nos aporta. Pasamos a conocer la metodología de implantación de un SGE.
- Una metodología de implantación es un proceso estructurado y metodológico para llevar a buen término el desarrollo.
- Una propuesta de metodología de implantación es la adaptada al ciclo de vida clásico del desarrollo de Software.

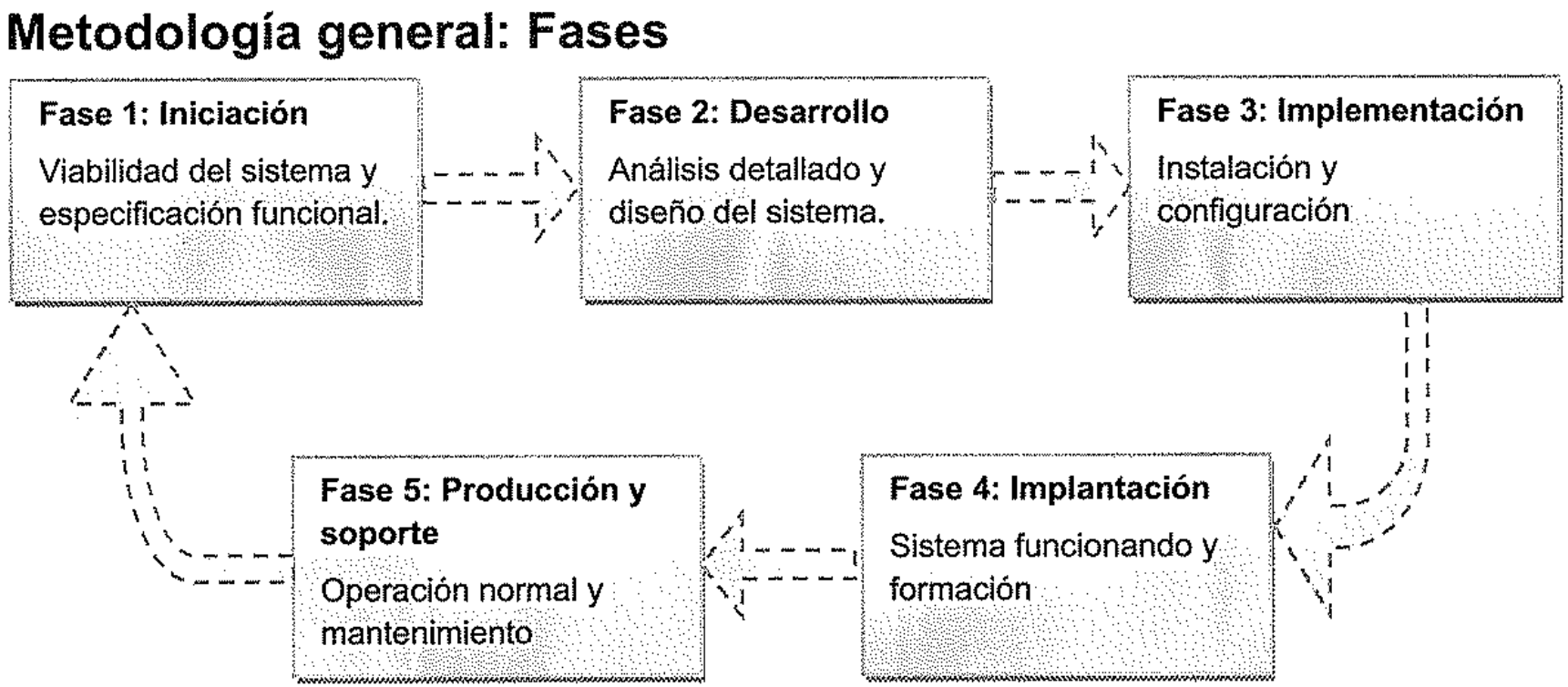
IMPLANTACIÓN DE UN ERP



Ciclo de vida clásico

Ciclo de vida clásico de un proyecto Software

IMPLANTACIÓN DE UN ERP



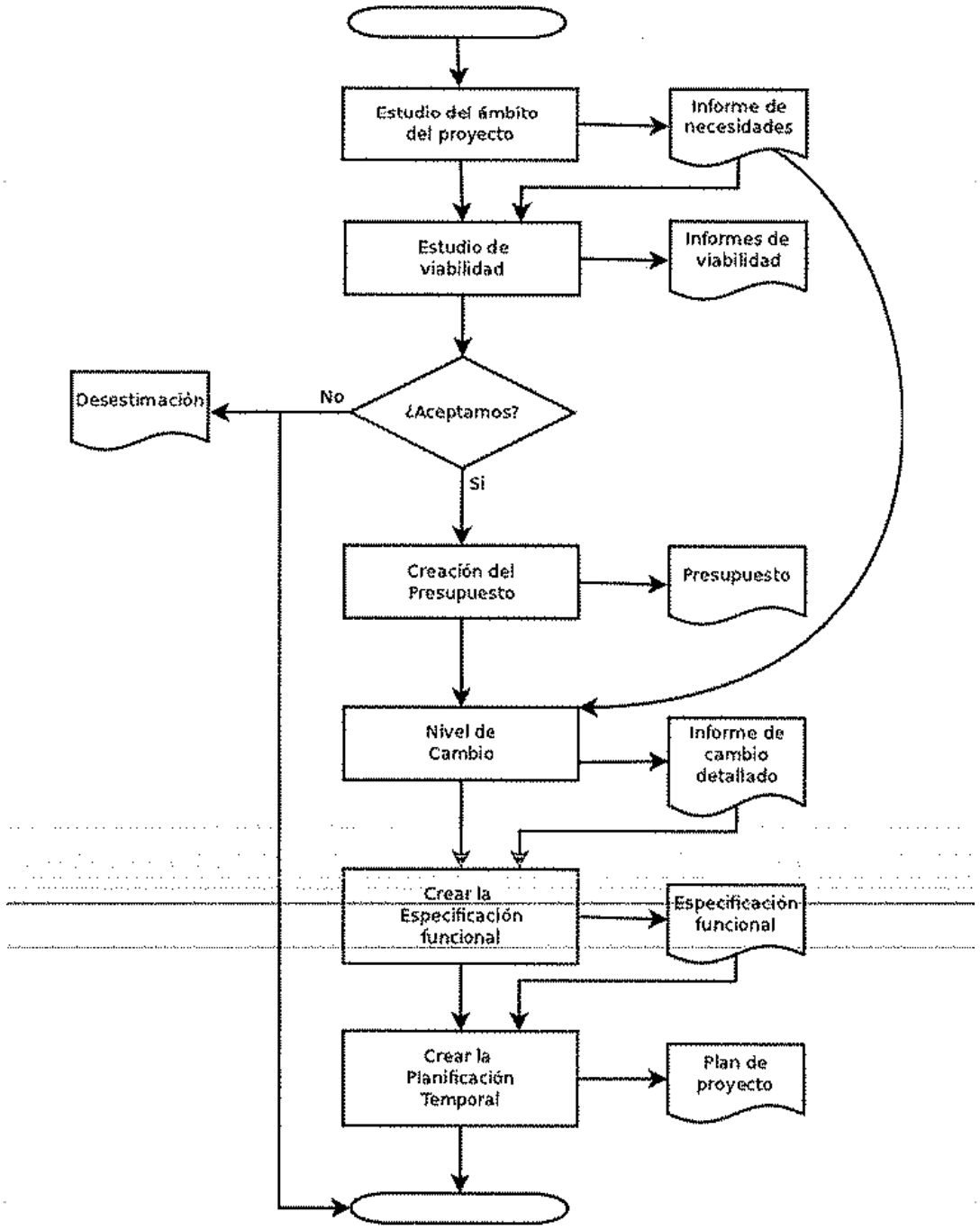
IMPLANTACIÓN DE UN ERP

Fase 1: Iniciación

Subfases

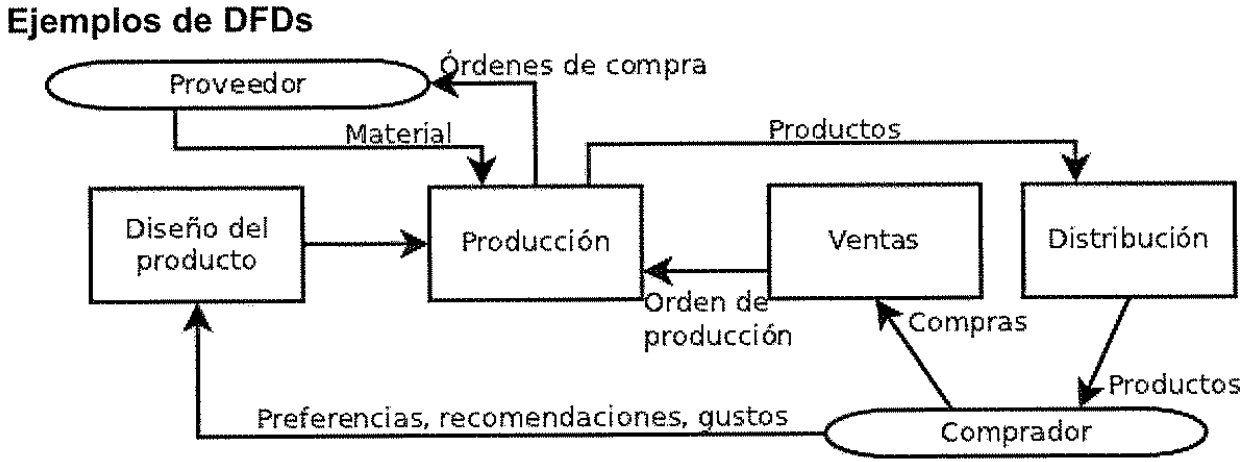
1. Estudiar el ámbito del proyecto.
2. Realizar un estudio de viabilidad económica, técnica y organizativa.
3. Determinar el nivel de cambio del nuevo sistema con respecto al original.
4. Organizar y planear el proyecto.

SUBFASE	ENTRADA	SALIDA
1. Estudiar ámbito del proyecto		Informe necesidades expresadas por el cliente y la fecha de finalización del proyecto, una relación de componentes de laempresa que se van a ver afectados y en qué camtidad, toda la información que se estime importante respecto a cambios a introducir en el sistema productivo y su funcionamiento.
2.Estudio viabilidad	Informe anterior	Propuesta de aceptación (presupuesto) o desestimación del proyecto.
3.Determinar el nivel de cambio del nuevo sistema con respecto al original (si existe)	Informe de necesidades	Informe de cambios detallado.
4.Organizar y planear el proyecto		Especificación funcional del sistema y plan del proyecto.

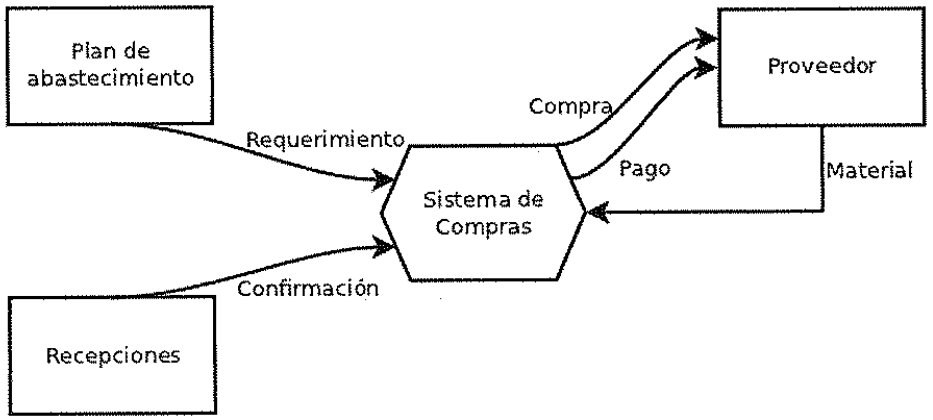


DFD resumen de la fase Uno

Ejemplos DFD (diagramas para el análisis)

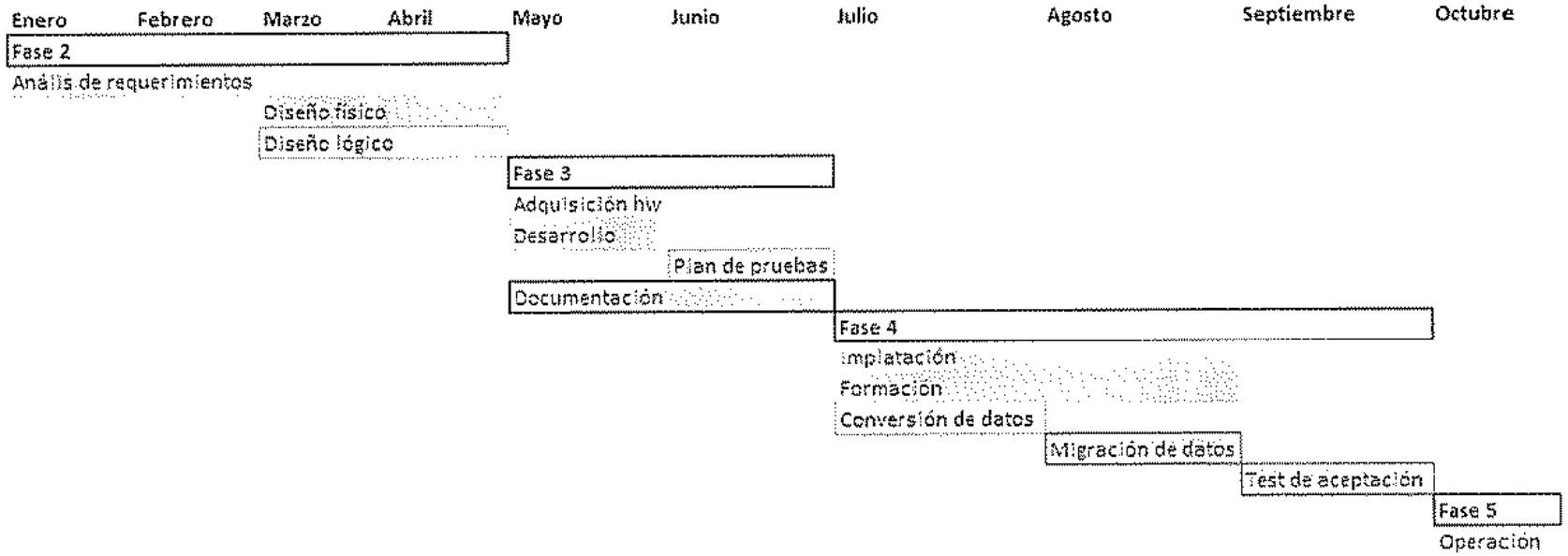


DFD general de una empresa (subfase 1)



DFD del sistema de compras (subfase 4)

Diagrama de Gannt-Planificación proyecto



Plan de proyecto simplificado (subfase 4)

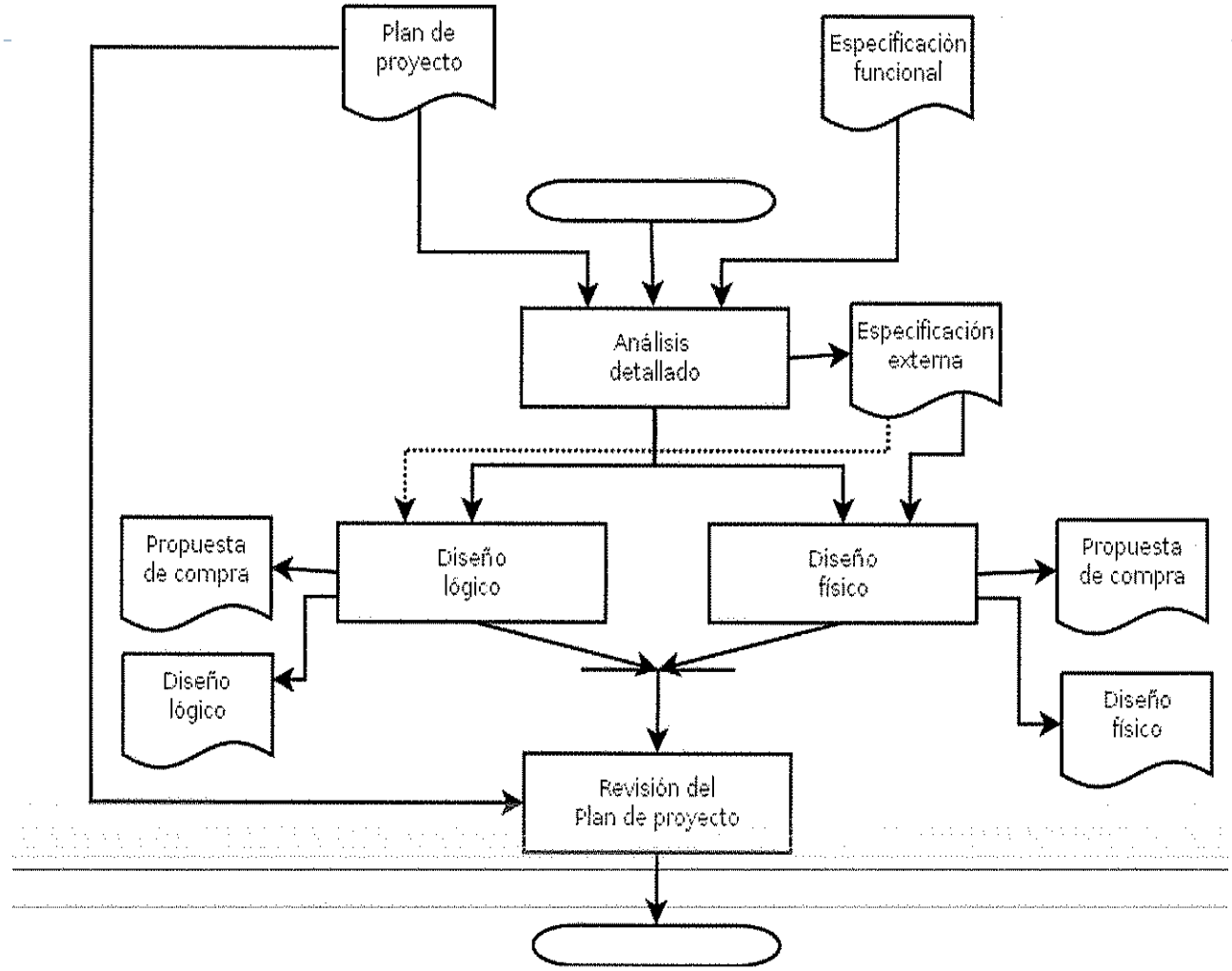
IMPLANTACIÓN DE UN ERP

Fase 2: Desarrollo

Subfases:

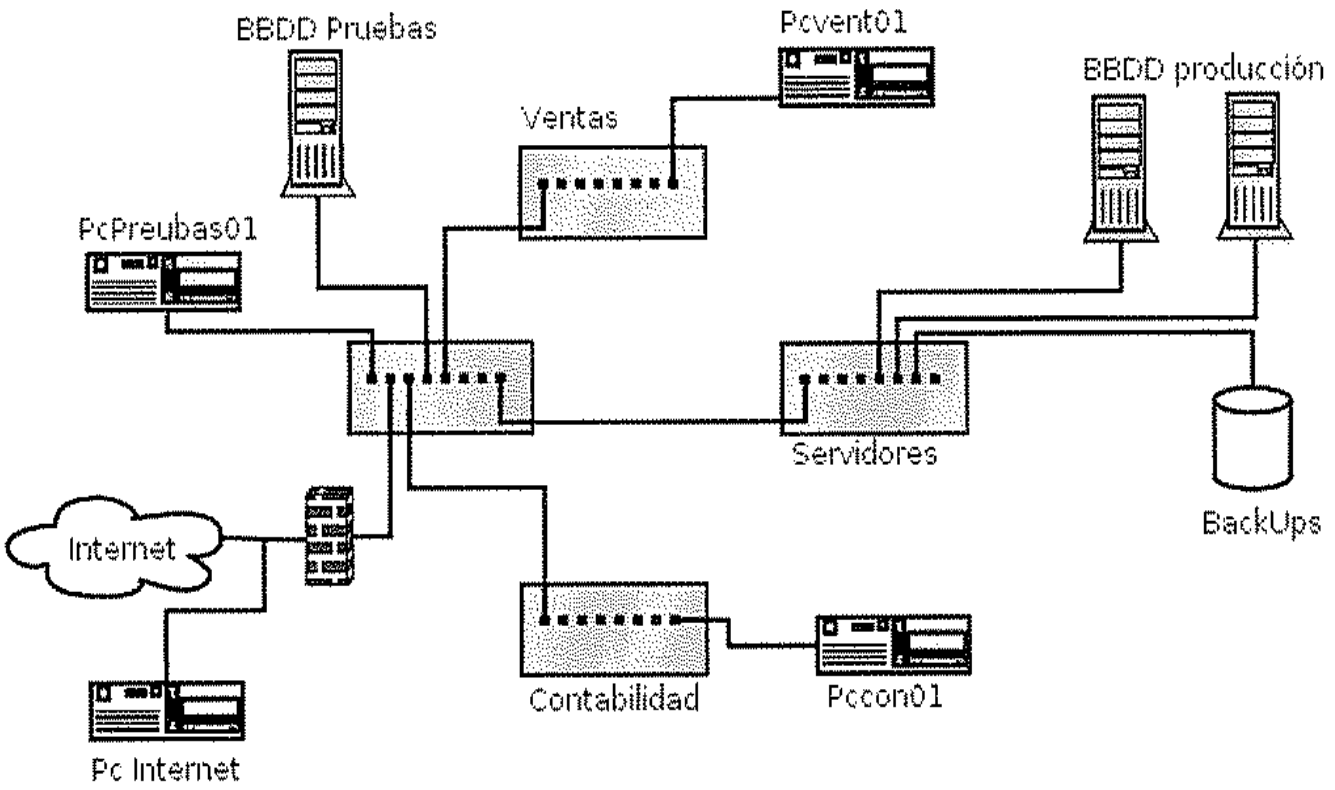
1. Análisis detallado.
2. Diseño físico del sistema (hardware).
3. Diseño lógico del sistema (software).
4. Revisión de las previsiones.

SUBFASE	ENTRADA	SALIDA
1.Análisis Detallado	Especificación funcional y plan de proyecto	Especificación externa.
2.Diseño físico del sistema	Especificación externa	Diseño físico y propuesta de compra de Hardware
3.Diseño lógico del sistema	Exsecificación externa	Diseño lógico y propuesta de compra de Software
4.Revisión de las previsiones	Plan de proyecto	Plan de proyecto revisado



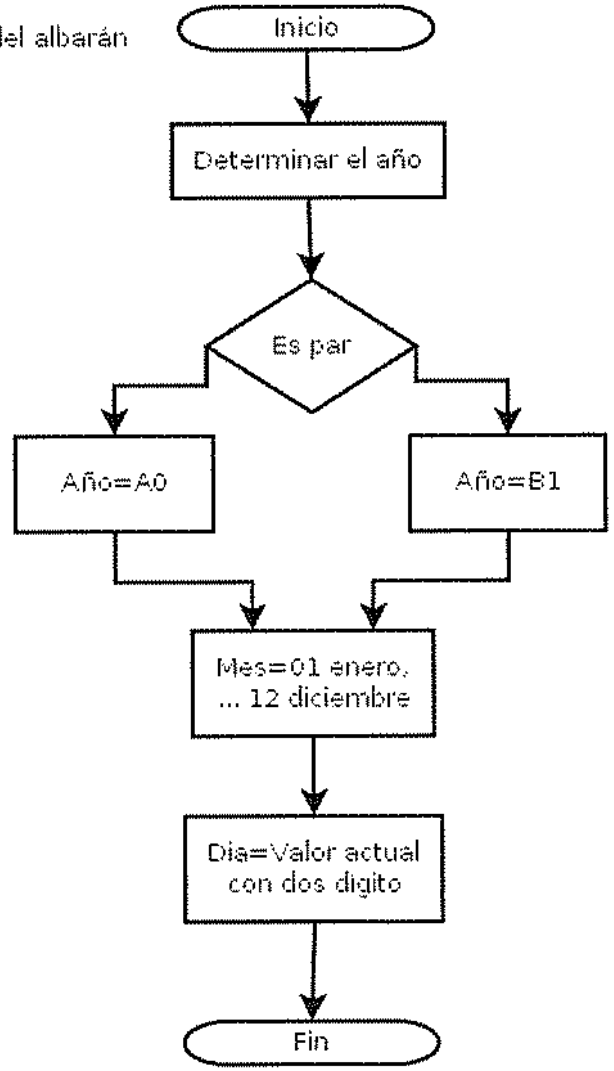
DFD resumen de la fase Dos

Ejemplos de DFDs



Diseño físico esquema general (subfase 2)

Cálculo del número del albarán
año par: A0MMDD
apo impar:B1MMDD



Cálculo del número del albarán (subfase 4)

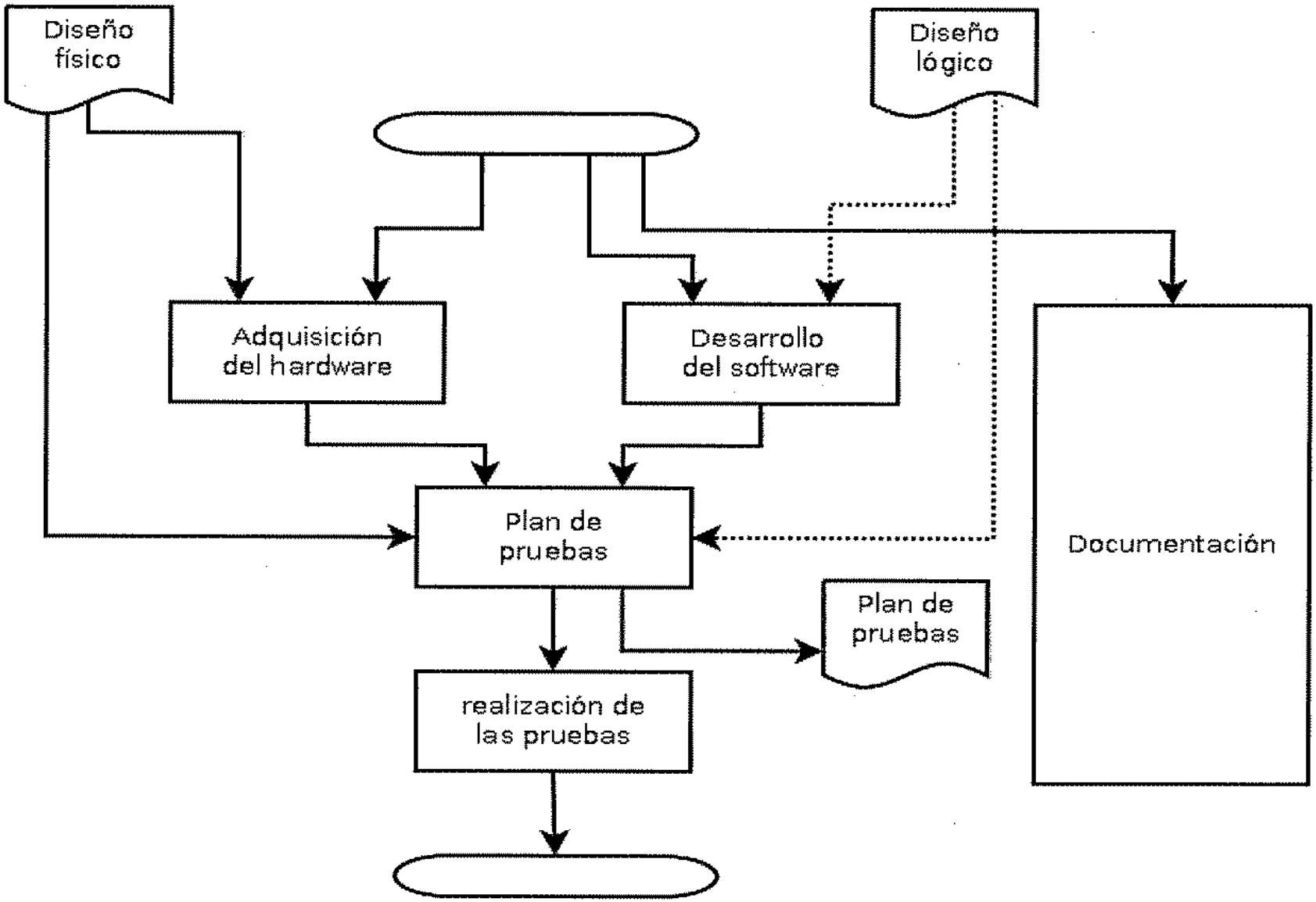
IMPLANTACIÓN DE UN ERP

Fase 3: Implementación

Subfases:

1. Adquisición del hardware.
2. Desarrollo de software.
3. Plan de pruebas.
4. Documentación.

SUBFASE	ENTRADA	SALIDA
1.Adquisición de Hardware.	Diseño físico y propuesta de compra	ninguna
Desarrollo de Software	Diseño lógico y propuesta de Compra	ninguna
Plan de pruebas	Diseño físico y diseño lógico	Plan de pruebas y documento de sistema probado
Documentación.	Diseño físico, diseño lógico,plan de pruebas,toda la documentación anterior	Documentación técnica del sistema



DFD resumen de la fase Tres

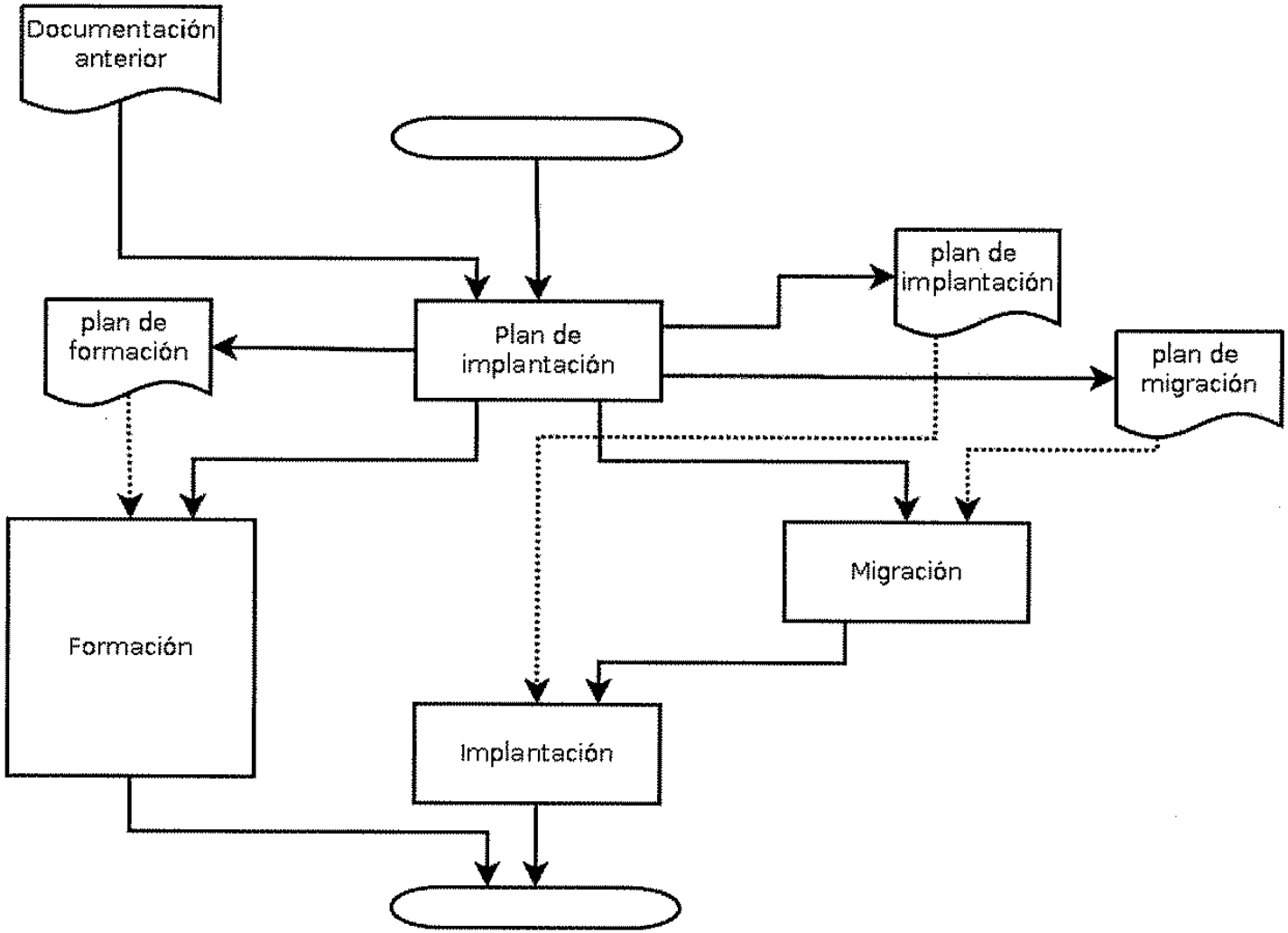
IMPLANTACIÓN DE UN ERP

Fase 4: Implantación

Subfases:

1. Plan de implantación.
2. Implantación.
3. Formación.
4. Conversión y migración de datos.
5. Test de aceptación.

SUBFASE	ENTRADA	SALIDA
1.Plan de implantación	Documentación anterior	Plan de implantación, plan de formación,plan de migración y conversión.
2.Implantación	Plan de implantación	ninguna
3.Formación	Plan de formación	ninguna
4.Conversión y migración de datos.	Plan de migración,conversión.	ninguna
5.Test de aceptación	Pruebas a realizar	Documento de aceptación del sistema



DFD resumen de la fase Cuatro

IMPLANTACIÓN DE UN ERP

Fase 5: Producción y soporte

Subfases:

1. Operación normal.
2. Soporte.
3. Mantenimiento.
4. Documentación al cliente.

EJEMPLO

- VER DOCUMENTO EJEMPLO FASE 1(páginas desde 195 hasta 199)