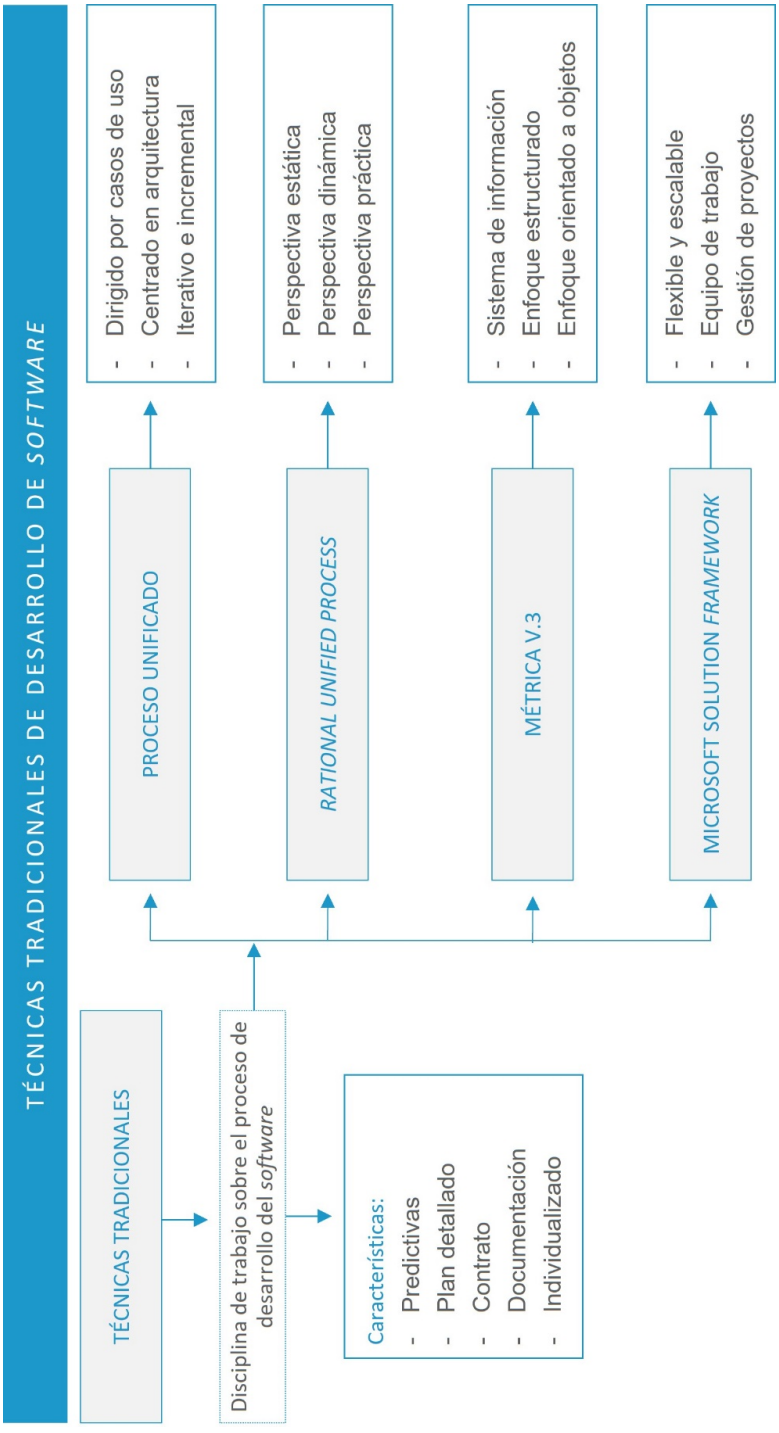
Metodologías, Desarrollo y Calidad en la Ingeniería de Software

Tema 2. Técnicas tradicionales de desarrollo de software

Índice

Esquema



2.1. Introducción y objetivos

En este tema nos centramos en el estudio de algunas de las **técnicas tradicionales** más conocidas en el desarrollo de productos *software*. Estas tienen la ventaja de que el ingeniero de *software* puede utilizarlas como guía elaborada de tareas, actividades y fases, que puede emplear para organizar el marco de gestión y el desarrollo de un proyecto de *software*.

Estas técnicas aportan una **visión práctica** del ciclo de construcción de sistemas *software*, esta visión es de vital importancia a la hora de su adopción en proyectos reales. Ofrecen un marco coherente e integrado que **guía las etapas y fases** fundamentales de todo desarrollo de *software.*

Con el estudio de este tema pretendemos alcanzar los siguientes objetivos:

- Conocer las características principales de las técnicas tradicionales.
- Estudiar el Proceso Unificado de desarrollo de *software.*
- Conocer la técnica de desarrollo *Rational Unified Process*.
- Estudiar la técnica «métrica v.3», que permite desarrollar sistemas de información. tanto con un enfoque estructurado como con un enfoque orientado a objetos.
- Conocer el marco de desarrollo *Microsoft Solution Framework*.

2.2. Técnicas tradicionales

Las técnicas o metodologías tradicionales comenzaron basándose en **metodologías existentes** en otras áreas, siendo utilizadas para el desarrollo o la producción de productos físicos, como edificios o coches, que se caracterizan por realizar un estudio de los requisitos, planificar las tareas y evaluar el presupuesto previo al inicio del desarrollo, invirtiendo un gran esfuerzo.

Estas técnicas tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software,* con el fin de conseguir un *software* más eficiente. Hacen énfasis en la **planificación** de todo el trabajo a realizar al inicio y, una vez que está todo detallado, comienza el desarrollo del *software.* Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado, y documentación detallada. Además, estas técnicas presentan **inflexibilidad e inadaptabilidad** ante las posibles modificaciones, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no se pueden predecir o pueden cambiar.

Las principales características de las técnicas tradicionales son:

- Son predictivas. Se define todo desde el comienzo.
- Determinan un plan detallado.
- Establecen contratos estrictos.
- Retrasan el error. No detectan el error cuanto antes para resolverlo.
- Prima más lo individual que lo colectivo.
- No se adaptan al cambio.

- Exceso de documentación.
- No hay mucha retroalimentación. El cliente delega su responsabilidad.

Carencias actuales

> El enfoque tradicional no ha dejado de considerarse «acertado», ni hemos pasado a decir ahora que el enfoque ágil es ahora el «correcto».

No se trata de que uno es «malo» y el otro «bueno»: el **contexto de gestión** (en general, en cualquier industria o sector) está cambiando. ¿Hacia qué está cambiando?

Cualquiera sea la industria, podemos decir que:

- **Existe una alta incertidumbre.** Quien lee esto, ¿puede decir qué es lo que va a suceder con seguridad en su empresa durante los próximos tres o cuatro años o incluso en un horizonte de un año?
- **Existe un altísimo ritmo de cambio:** ¿cuántas veces al año se cambian los planes de trabajo? ¿Cuántas veces aparecen imprevistos que hacen replanificar y replantear el trabajo?
- **Todo basado en tecnología:** ¿podríamos hacer referencia a algún tipo de negocio que no necesite tecnología? La tecnología no deja de cambiar y de evolucionar. Esto nos afecta en muchos sentidos, por lo tanto, es una fuente de cambio innegable y constante.

Pues bien, el **enfoque ágil** es el que parece más adecuado para enfrentarnos a este tipo de contexto. Es un hecho que ya, a largo plazo, los enfoques «tradicionales» irán desapareciendo (si entendemos como enfoque «tradicional» llevar a cabo una planificación predictiva plasmada como diagrama de Gantt con un jefe de proyecto o similar como figura «que piensa» y asigna el trabajo de forma centralizada, con entrega de valor basada en el cumplimiento de hitos fijos…). Este enfoque de trabajo

no cuadra nada bien con el tipo de contexto actual, más propio y consecuencia natural de la **transformación digital** y de los negocios que estamos viviendo en todos los ámbitos.

2.3. El proceso unificado de desarrollo de software

El **proceso unificado de desarrollo de *software*** (*Unified Software Development Process*, USDP), más conocido de manera simplificada como **proceso unificado** *(Unified Process, UP)*, se corresponde con un proceso de desarrollo de *software* elaborado por los autores del Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML): Grady Booch, James Rumbaugh e Ivar Jacobson.

Un proceso de desarrollo efectivo debe describir lo que hace cada participante, cuándo y cómo lo hace.

Esto es lo que define exactamente UP en términos de los siguientes conceptos claves:

- **Roles:** quién.
- **Artefactos:** qué debe hacerse.
- **Actividades:** cómo.
- Fases, iteraciones y detalles organizados en **flujos de trabajo:** cuándo.

Las características principales de UP son las siguientes:

-
  - Es adecuado en proyectos donde existe mucha incertidumbre y que requieren una constante revisión y aceptación.
  - Su estructura se adapta a una comprensión incremental del problema.
  - El desarrollo se aborda mediante refinamientos sucesivos.
  - El riesgo se gestiona a medida que el proyecto avanza.

- Se dispone de versiones ejecutables, permitiendo una mayor retroalimentación, verificación y validación efectiva.
- Adaptable a cambios durante el ciclo de vida del proyecto.

De acuerdo con sus creadores (Jacobson, Booch y Rumbaugh, 2000), **UP es un proceso basado en componentes,** lo cual quiere decir que el sistema *software* en construcción está formado por componentes *software* interconectados a través de interfaces bien definidas (es decir, el conjunto de operaciones que son utilizadas para especificar la funcionalidad que ofrece un componente). En este sentido, un componente *software* sería la parte física y reemplazable de un sistema *software* que se ajusta a, y proporciona la realización de, un conjunto de interfaces (Booch, Rumbaugh y Jacobson, 2005, p. 453).

Además, el proceso unificado utiliza UML para modelar todos los artefactos del sistema *software.* Pero los auténticos aspectos definitorios y distintivos de UP se resumen en los siguientes **conceptos básicos** (Jacobson, Booch y Rumbaugh, 2000):

- **Dirigido por casos de uso.** El proceso de desarrollo avanza a través de una serie de actividades que parten de los casos de uso. Los casos de uso se especifican, se diseñan y son la fuente a partir de la cual los ingenieros de pruebas construyen los casos de prueba del sistema *software.*
  - Los casos de uso son un medio de capturar y representar requisitos funcionales.
  - Los casos de uso dirigen el diseño, la implementación y las pruebas.
  - Aumentan la probabilidad de satisfacer las necesidades del usuario final, puesto que se definen con el foco puesto en ellos.
  - Permiten la trazabilidad a lo largo de todo el proceso de desarrollo.

Ideas clave

- **Centrado en la arquitectura.** El papel de la arquitectura *software* es parecido al papel que juega la arquitectura en la construcción de edificios. El edificio se contempla desde varias perspectivas: estructura, servicios, conducción de la calefacción, fontanería, electricidad, etc. Esto permite al constructor ver una imagen completa antes de que comience la construcción. Análogamente, la arquitectura en un sistema *software* se describirá mediante diferentes vistas del sistema en construcción (estática, dinámica…). Con lo cual, el concepto de arquitectura *software* incluye los aspectos estáticos y dinámicos más significativos del sistema *software.*

- **Iterativo e incremental.** La idea de este modelo de proceso es dividir el trabajo en partes más pequeñas o mini-proyectos, donde cada mini-proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en la actividad y los incrementos hacen referencia al crecimiento del producto. Al ser mini-proyectos comienzan con los casos de uso y continúan con las tareas de análisis, diseño, implementación y pruebas, que terminan convirtiendo en código ejecutable los casos de uso que iniciaban la iteración, es decir, cada iteración va a contener todos los elementos de un proyecto de desarrollo de *software:* planificación; análisis y diseño; construcción; integración y pruebas; y una versión interna o externa.

Cada iteración genera una línea base que comprende una versión parcialmente completa del sistema final y toda la documentación del proyecto asociada, (para generar una línea base es necesario que todos los artefactos de la iteración que representa hayan sido revisados y aprobados a través de procedimientos formales). La diferencia entre dos líneas base consecutivas es lo que se ha denominado incremento. Además, en cada iteración, el proceso unificado engloba cinco **flujos de trabajo** principales:

- **Requisitos,** que recoge lo que el sistema debería hacer.

- **Análisis,** que refina y determina la estructura de los requisitos del sistema.

- **Diseño,** que realiza los requisitos que se dan en la arquitectura del sistema.
- **Implementación,** que construye el *software.*
- **Pruebas,** que verifica que la implementación funciona correctamente y cumple los requisitos.

El proceso unificado se repite a lo largo de una serie de **ciclos** que constituyen la vida de un sistema *software.* Cada ciclo constituye una versión del sistema y se encuentra dividido en cuatro fases, las que se describen a continuación (Jacobson, Booch y Rumbaugh, 2000):

- **Inicio.** Esta fase tiene como principal objetivo realizar el **análisis del dominio del problema.** Durante la fase de inicio se desarrolla una descripción del producto final y se presenta el análisis estratégico para el producto. De esta manera, si la contribución del sistema *software* para el negocio es adecuada, entonces se sigue adelante con el proyecto. En esta fase se genera el modelo de casos de uso con los casos de uso más críticos, se esboza la arquitectura a través de los subsistemas más importantes, se identifican y priorizan los riesgos más importantes, se planifica en detalle la fase de elaboración y se realiza una estimación orientativa del proyecto.
- **Elaboración.** El objetivo principal de esta fase es el **análisis del dominio.** Se trata de establecer una línea base para la arquitectura y desarrollar un plan de proyecto. Otro de los objetivos es obtener una visión del sistema en toda su extensión mediante una descripción no detallada en profundidad. Durante la fase de elaboración se especifican en detalle la mayoría de los casos de uso del producto (es decir, comprensión del dominio del problema) y se diseña la arquitectura del sistema.
- **Construcción.** El objetivo principal de esta fase es el **desarrollo de los componentes de diseño y su integración** para dar soporte a la funcionalidad del sistema, que deberá ser probada posteriormente. Durante la fase de construcción se crea el producto, en esta fase la línea base de la arquitectura crece hasta convertirse

en el sistema completo. Como ya se comentaba anteriormente, la línea base constituye un conjunto de artefactos revisados y aprobados que representa un punto de acuerdo para la posterior evolución y desarrollo del sistema, y que solamente puede ser modificado a través de procedimientos formales.

- **Transición.** El objetivo principal de esta fase es la de **liberar el producto *software* al entorno de los usuarios.** Durante la fase de transición se cubre el periodo durante el cual el producto se convierte en versión que se podría llamar beta, que se irá refinando hasta su versión final. En esta fase los desarrolladores corrigen los problemas e incorporan las mejoras que se estimen convenientes.

Cada fase termina con un **hito** que se determina por la disponibilidad de un conjunto de artefactos (es decir, información que ha sido desarrollada para construir el sistema y se encuentra en un determinado estado). El número de iteraciones que se realizarán en cada fase dependerá del **tamaño del proyecto.** La Figura 1 resume este proceso.

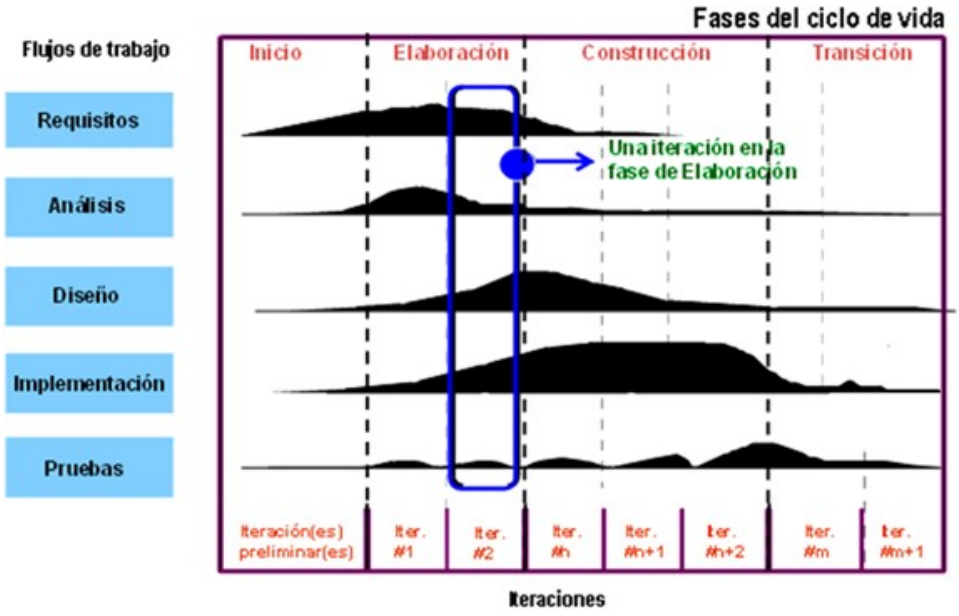


Figura 1. Fases del ciclo de vida del proceso unificado. Fuente: Jacobson, Booch y Rumbaugh (2000, p. 11).

2.4. Rational Unified Process (RUP)

***Rational Unified Process* (RUP)** (Kruchten, 2003) es un producto comercial de IBM (desarrollado por la empresa *Rational Software Corporation,* que hoy en día forma parte del conglomerado de empresas propiedad de IBM) que constituye un ejemplo de modelo de proceso resultado de la colaboración de UML (Booch, Rumbaugh y Jacobson, 2005) y el USDP (Jacobson, Booch y Rumbaugh, 2000). En pocas palabras, RUP es una variante o un producto comercial del UP descrito anteriormente.

Esta técnica está desarrollada con el objetivo de que cada empresa pueda adaptarla según su organización y el equipo de desarrollo del que dispone, por lo que no se trata de un marco de trabajo estático.

RUP defiende que los modelos de proceso convencionales presentan una **única perspectiva del proceso,** es decir, una secuencia de actividades conectadas que incorporan estrategias para llevar a cabo la evolución del *software*. Sin embargo, RUP, de acuerdo con Sommerville (2005, p. 76), se puede describir bajo tres perspectivas distintas:

- **Perspectiva estática.** La perspectiva estática muestra cómo están dispuestas las actividades (lo que se conoce como flujos de trabajo en terminología RUP) que comprende el proceso:
  - **Modelado del negocio,** donde se modelan los procesos de negocio mediante los casos de uso de negocio.
  - **Requisitos,** donde se identifican los actores que interactúan con el sistema y se desarrollan casos de uso que modelan los requisitos del sistema.

- **Análisis y diseño,** donde se genera y se documenta un modelo de diseño utilizando modelos arquitectónicos, modelos de componentes, modelos de objetos y modelos de secuencia.

- **Implementación; pruebas; despliegue; gestión de la configuración y del cambio; gestión del proyecto; y entorno,** donde se pone a disposición del equipo de desarrollo un conjunto de herramientas *software* que ayudarán a la implementación del sistema.

- **Perspectiva dinámica.** La perspectiva dinámica muestra las fases del modelo de proceso a lo largo del tiempo.

- **Perspectiva práctica.** La perspectiva práctica sugiere buenas prácticas que se deben utilizar durante el proceso.

La distinción entre fases y flujos de trabajo (actividades), y el reconocimiento del despliegue del *software* como parte del proceso de desarrollo, son algunas de las **innovaciones más importantes** que merece la pena destacar de RUP. Las fases son dinámicas y tienen objetivos. Los flujos de trabajo son estáticos y se corresponden con actividades que no están asociadas a una única fase, pero que se pueden utilizar a lo largo del proceso de desarrollo para alcanzar los objetivos de cada fase.

La Figura 2 muestra las distintas fases de RUP. Cada **fase** se puede llevar a cabo de modo iterativo, donde cada resultado se desarrolla de manera incremental. Por otro lado, el conjunto de todas las fases (es decir, un ciclo) también se puede llevar a cabo de manera incremental, tal y como muestra la flecha en la Figura 2, que parte de la fase de transición y vuelve hacia la fase de inicio.

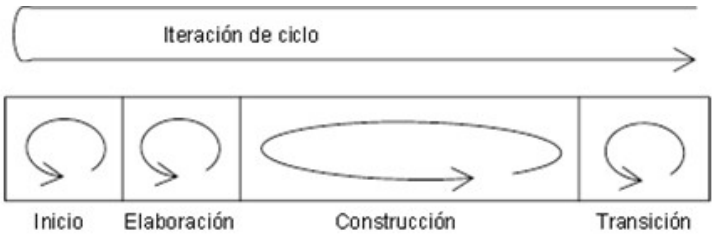


Figura 2. Fases en RUP. Fuente: Sommerville (2005, p. 77).

Bajo el enfoque de RUP, el proyecto *software* será llevado a cabo por un conjunto de actores que de cara al sistema podrán adoptar distintos **roles.** Cada rol participará en una o más actividades del proyecto y entre todos producirán los distintos artefactos. De esta manera, actividades, roles y artefactos constituyen los elementos de proceso básicos de RUP.

Durante la **fase de inicio,** se desarrolla una descripción del sistema final y se presenta un análisis de negocio para el sistema. El objetivo de esta fase es la **definición del alcance del proyecto** para poder realizar la validación del presupuesto del proyecto. El producto final de esta fase será el **modelo de negocio** que se pretende conseguir una vez terminado el proyecto, incluyendo un pronóstico financiero. Para ello es necesario realizar un estudio de los casos de uso del producto final, su evaluación y el plan de riesgos que pueden surgir durante el proyecto.

Para poder obtener el informe final de esta fase inicial, es importante definir los siguientes criterios:

- Usuarios claves del proyecto (usuarios, clientes, proveedores, etc.).
- Requisitos para poder definir los casos de uso del proyecto.
- Planificación, definición del calendario de trabajo, incluyendo los horarios.
- Presupuesto de los gastos estimados.

- Informe de los posibles riesgos.
- Definición del proceso de desarrollo.
- Establecer una línea base tanto de la planificación como de los costes estimados para poder realizar la comparativa del estado del proyecto en cualquier punto del mismo.

Durante la **fase de elaboración** el proyecto empieza a tomar forma, ya que se identifican los elementos claves de riesgo para elaborar un plan que pueda mitigar el impacto de los mismos. El objetivo de esta fase es **obtener el análisis y la arquitectura del proyecto.** Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.

En RUP la arquitectura se expresa en forma de vistas a través de modelos del sistema (o modelos *software*), los cuales, todos juntos, representan el **sistema completo.** Con lo cual, se tendrán los siguientes modelos que expresan las distintas vistas arquitectónicas: modelo de casos de uso (aunque en la terminología clásica el modelo de casos de uso formaría parte del modelo de análisis), modelo de análisis, modelo de diseño, modelo de implementación y modelo de despliegue. Además, en esta fase se realizan los casos de uso más críticos que se identificaron en la fase de inicio.

El resultado de esta fase debe ser un informe que incluya los siguientes puntos:

- Modelo de caso de uso y los usuarios que intervienen en cada uno de los casos de uso.
- La descripción de la arquitectura teniendo en cuenta los casos de uso más críticos.
- Lista de los riesgos que pueden surgir y el plan de mitigación de cada uno de ellos.
- Plan de desarrollo del proyecto.

Durante la **fase de construcción** se crea el sistema. Al final de esta fase el sistema contiene todos los casos de uso que se acordaron para el desarrollo de esta versión. Sin embargo, no se garantiza todavía que el sistema final se encuentre libre de errores. en esta fase se llevan a cabo las tareas de desarrollo de los componentes del *software.* En proyectos de gran tamaño se pueden realizar varias iteraciones de esta fase, ya que es conveniente dividir en segmentos los distintos casos de uso para simplificar la complejidad de los mismos y hacer más manejables los prototipos.

La **fase de transición** cubre el periodo durante el cual el producto constituye una versión beta del sistema (es decir, una versión que no se considera final, pero que ya se puede probar y entregar como una primera aproximación). Así, en la versión beta un número reducido de usuarios prueba el sistema e informa de los problemas encontrados. La fase de transición incluye por tanto actividades del tipo: formación del cliente, asistencia técnica y corrección de errores. El objetivo de esta fase es el **paso del sistema del entorno de desarrollo al de producción;** es decir, realizar el despliegue del sistema y ponerlo a disposición del usuario final. En esta fase también hay que realizar las tareas de validación final de los usuarios claves del sistema y enseñar a los usuarios la utilización de los mismos.

Las **buenas prácticas** recomendadas por RUP para su utilización en el desarrollo de sistemas son las que se describen a continuación (Sommerville, 2005, p. 78):

- **Desarrollar *software* de manera iterativa.** Planificar los distintos incrementos del sistema basándose en las prioridades del cliente, y desarrollar y entregar las características del sistema que tienen mayor prioridad al comienzo del proceso de desarrollo.

- **Gestión de requisitos.** Documentar todos los requisitos del cliente y llevar el control de todos los cambios que se producen en cada requisito. Analizar el impacto de los cambios sobre el sistema antes de aceptarlos.

- **Hacer uso de arquitecturas basadas en componentes.** Estructurar la arquitectura del sistema en forma de componentes.
- **Modelado visual del *software.*** Utilizar modelos gráficos UML para presentar las vistas estáticas y dinámicas del sistema.
- **Verificar la calidad del sistema.** Comprobar que el software cumple los estándares de calidad de la organización.
- **Controlar los cambios que se producen en el *software.*** Utilizar herramientas y procedimientos de gestión de la configuración y gestión del cambio para controlar las modificaciones que se producen en el sistema.

Si bien es cierto que RUP no resulta adecuado en todos los proyectos de desarrollo de *software,* sí que representa una aproximación adecuada para procesos genéricos. En este sentido, RUP constituye un marco de trabajo *(framework)* con un conjunto más o menos completo de elementos de proceso que tiene que ser **adaptado a cada caso** (es decir, adaptado en función del tamaño del sistema, el dominio en el cual va a funcionar, su complejidad, las capacidades del personal de la organización, etc.) (Jacobson, Booch y Rumbaugh, 2000).

Con lo cual, antes de aplicar RUP a los procesos de una organización se deberá pensar en lo que realmente se necesita y qué cosas se podrían descartar (Hanssen, Westerheim & Bjørson, 2005).

Utilizar RUP como base para los procesos de desarrollo de una organización implica, aparte de la adaptación del modelo, conservar las **propiedades básicas** de RUP: documentación y diálogo con el cliente basado en casos de uso; un enfoque arquitectónico con procesos iterativos; y un desarrollo incremental del producto.

Sin embargo, no se encuentran en RUP muchas directrices que indiquen cómo hay que hacer esta adaptación. Hanssen, Westerheim y Bjørson (2005), realizan un

estudio y describen tres formas de adaptar RUP:

- El *framework* se adapta a cada proyecto de manera individual, lo que supone una gran cantidad de trabajo.
- El *framework* se adapta generando otro *framework* que será un subconjunto de RUP, pero ahora en sintonía con las características generales de la organización.
- La organización identifica y describe los distintos tipos de proyecto que abarca, y el *framework* se adapta a cada uno de los tipos que se han encontrado.

A lo largo del tiempo se han detectado diversos **problemas y limitaciones** en RUP, reconocidos por los mismos autores que lo crearon, es por ello por lo que uno de ellos, Ivar Jacobson, ideó otro enfoque con el objetivo de solventar y mejorar el modelo proceso propuesto por RUP. Esta nueva aproximación se conoce como *Essential Unified Process* (EssUP) y recoge diversas técnicas de RUP, del modelo de mejora de procesos *Capability Maturity Model Integration* (CMMI) y del desarrollo ágil para su aplicación en los proyectos *software.* Lo que propone EssUP es que la organización pueda tomar aquellas prácticas que considere útiles para un proyecto *software* en particular y las **combine** como estime necesario dentro del proceso de desarrollo de *software,* de manera aislada, si así se requiere, y no incluyendo todas las que a su vez pueda tener relacionadas de acuerdo a RUP, cuyo volumen y esfuerzo podría llegar a ser inviable para el proyecto.

2.5. Métrica v.3

Métrica v.3, es una técnica desarrollada por el Ministerio de Hacienda y Administraciones Públicas, que proporciona a las empresas y organizaciones unas directrices para sistematizar las actividades que forman el ciclo de vida de los sistemas de información (Consejo Superior de Informática, 2001).

Desde una estructura unificada, cubre el desarrollo estructurado orientado a objetos y facilita, mediante interfaces, procesos de apoyo como la gestión de proyectos y el aseguramiento de la calidad y de la seguridad.

En términos generales, Métrica v.3 permite alcanzar una serie de objetivos:

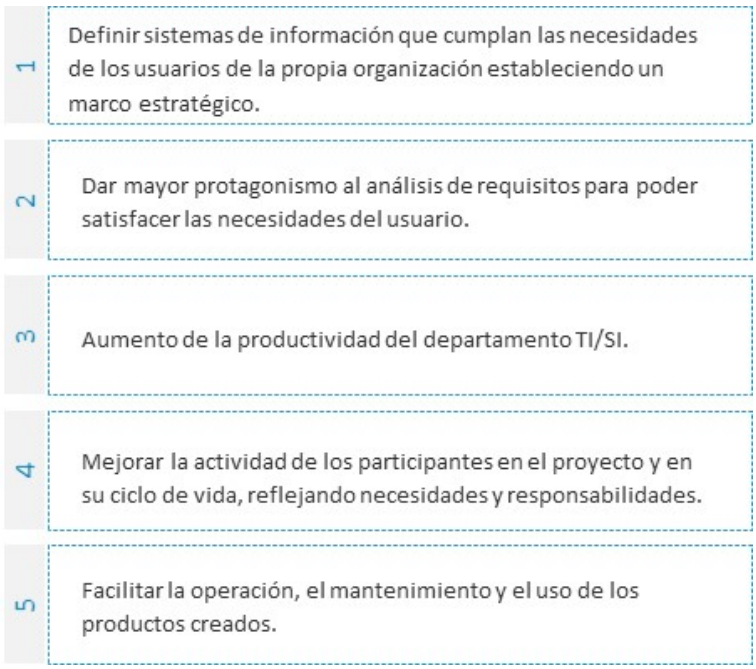


Figura 3. Objetivos de Métrica v.3. Fuente: Adaptada a partir de Consejo Superior de Informática, 2001.

En Métrica v.3 hay muchas actividades que se realizan **simultáneamente** y se **realimentan** entre sí (efecto de espiral), experimentando los modelos con varios estados de refinamiento progresivo.

La estructura principal de la Métrica v.3 abarca:

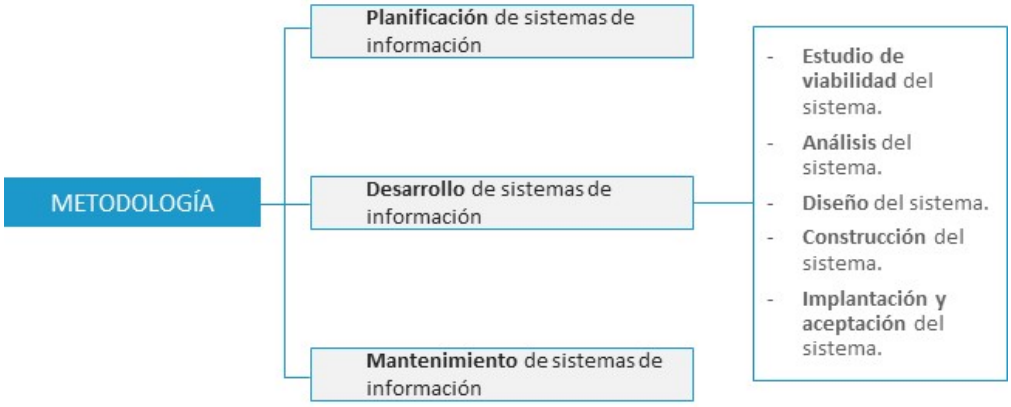


Figura 4. Estructura de la Métrica v.3. Fuente: adaptada a partir de Consejo Superior de Informática, 2001.

Junto a esta estructura, Métrica v.3 incluye un conjunto de procesos adicionales que definen una serie de actividades de tipo organizativo o de soporte al proceso de desarrollo y a los productos. Estos procesos son:

- **Gestión de proyectos (GP):** su finalidad principal es la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un SI. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos lo más pronto posible, lo cual evitará desviaciones temporales y económicas.

- **Seguridad (SEG):** tiene como objetivo incorporar en los SI mecanismos de seguridad adicionales a los que se proponen en la propia técnica, asegurando el desarrollo de cualquier tipo de sistema a lo largo de los procesos que se realicen.

- **Aseguramiento de la calidad (CAL):** su objetivo es proporcionar un marco común de referencia para la definición y puesta en marcha de planes específicos de aseguramiento de calidad aplicables a proyectos concretos.

- **Gestión de la configuración (GC):** su finalidad es identificar, definir, proporcionar información y controlar los cambios en la configuración del sistema, así como las modificaciones y versiones de los mismos.

Planificación de sistemas de información (PSI)

El PSI tiene como objetivo la **obtención de un marco de referencia** para el desarrollo de sistemas de información que responda a los objetivos estratégicos de la organización. Este marco de referencia consta de:

- Una descripción de la situación actual, que constituirá el punto de partida del PSI.
- Un conjunto de modelos que constituya la arquitectura de información.
- Una propuesta de proyectos a desarrollar en los próximos años, así como la prioridad de realización de cada proyecto.
- Una propuesta de calendario para la ejecución de dichos proyectos.
- La evaluación de los recursos necesarios para los proyectos a desarrollar en el próximo año, con el objetivo de tenerlos en cuenta en los presupuestos.
- Un plan de seguimiento y cumplimiento de todo lo propuesto mediante unos mecanismos de evaluación adecuados.

Es fundamental que la alta dirección de la organización tome **parte activa** en la decisión del PSI, con el fin de posibilitar su éxito. La presentación del PSI y la constitución del equipo supone el arranque del proyecto y es fundamental que las más altas instancias de la organización estén implicadas en ambos, dando el apoyo necesario y aportando todo tipo de medios. Explicar el plan a las personas de la organización y a las unidades organizativas afectadas sobre las que recaerá el plan, el apoyo de los altos directivos y la cualificación de los recursos de las distintas unidades implicadas, serán **factores críticos de éxito** del PSI.

Para la elaboración del PSI se estudian las **necesidades de información** de los procesos de la organización afectados por el plan, con el fin de definir los requisitos generales y obtener modelos conceptuales de información. Por otra parte, se evalúan

las opciones tecnológicas y se propone un **entorno.** Tras analizar las prioridades relacionadas con las distintas variables que afectan a los SI, se elabora un calendario de proyectos con una planificación lo más detallada posible de los más inmediatos. Además, se propone una sistemática para mantener actualizado el PSI para incluir en él todos los cambios necesarios, **garantizando el cumplimiento** adecuado del mismo.

En la Figura 5 se muestra la secuencia de actividades del proceso PSI.

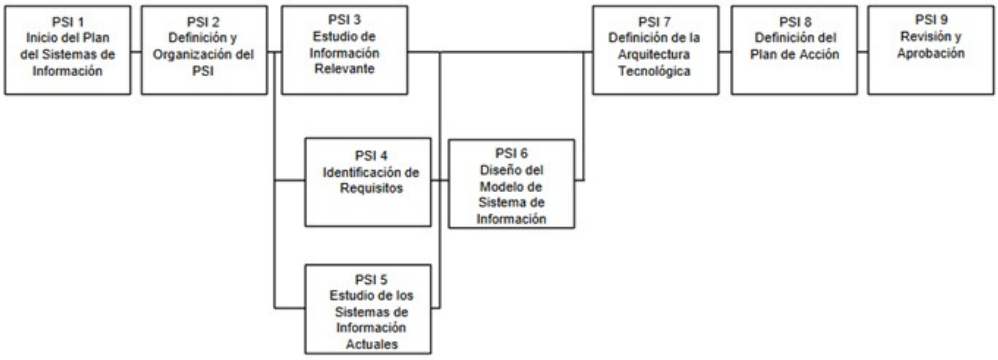


Figura 5. Secuencia de actividades del PSI. Fuente: Consejo Superior de Informática, 2001.

Desarrollo de sistemas de información

El proceso de desarrollo contiene todas las **actividades y tareas** que se deben llevar a cabo para desarrollar un sistema, cubriendo desde el análisis de requisitos hasta la instalación del *software.*

En Métrica v.3 se han abordado los dos tipos de desarrollo: **estructurado y orientado a objetos,** por lo que ha sido necesario establecer actividades específicas a realizar en alguno de los procesos cuando se utiliza la tecnología de orientación a objetos. Para este último caso, se han analizado algunas de las propuestas de otras metodologías orientadas a objetos y se han tenido en cuenta la mayoría de las técnicas que contempla UML (*Unified Modeling Language*).

A continuación, se describen los procesos de los que está compuesto el desarrollo de sistemas de información:

Estudio de viabilidad del sistema (EVS)

El **objetivo del EVS** es el análisis de un conjunto concreto de necesidades para proponer una solución a corto plazo que tenga en cuenta restricciones económicas, técnicas, legales y operativas. La solución obtenida como resultado del estudio puede ser la definición de uno o varios proyectos que afecten a uno o varios sistemas de información ya existentes o nuevos. Para ello se identifican los requisitos que se han de satisfacer y se estudia, si procede, la situación actual.

A partir del estado inicial, la situación actual y los requisitos planteados, se estudian las **alternativas de solución.** Dichas alternativas pueden incluir soluciones que impliquen desarrollos a medida, soluciones basadas en la adquisición de productos software del mercado o soluciones mixtas. Se describe cada una de las alternativas indicando los requisitos que cubre.

Una vez descritas las alternativas planteadas, se valora su impacto en la organización, la inversión a realizar en cada caso y los riesgos asociados. Esta información se analiza con el fin de evaluar las distintas alternativas y seleccionar la más adecuada, definiendo y estableciendo su **planificación.**

Si en la organización se ha efectuado con anterioridad un **plan de sistemas de información** que afecte al sistema objeto de este estudio, se dispondrá de un conjunto de productos que proporcionarán información parar tener en cuenta en todo el proceso.

Las actividades que engloba este proceso se recogen en la Figura 6, en la que se indican las actividades que pueden ejecutarse en **paralelo** y las que precisan, para su realización, **resultados** originados en actividades anteriores.

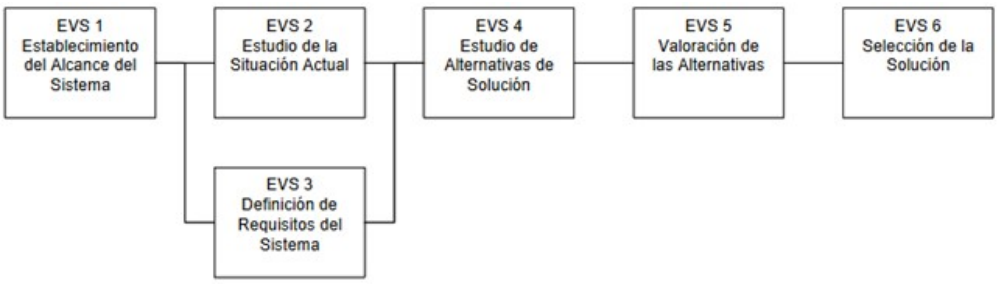


Figura 6. Secuencia de actividades del EVS. Fuente: Consejo Superior de Informática, 2001.

Análisis del sistema de información (ASI)

El objetivo del análisis del sistema de información es obtener su **especificación detallada,** de forma que satisfaga las necesidades de información de los usuarios y sirva de base para el posterior diseño del sistema.

Consta de **once actividades:** siete comunes, dos orientadas a objetos y dos estructuradas, como se puede observar en la Figura 7.

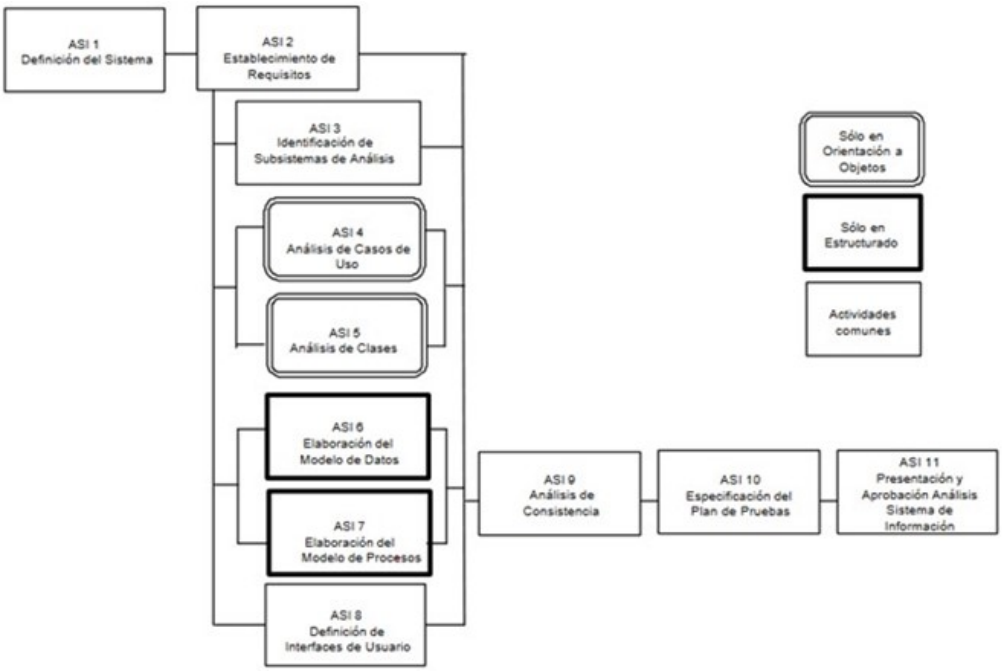


Figura 7. Secuencia de actividades del ASI. Fuente: Consejo Superior de Informática, 2001.

Ideas clave

- En **ASI 1, definición del sistema,** partiendo de los productos generados en el EVS, se lleva a cabo la definición inicial del sistema de información:
  - Delimitando el alcance del sistema.
  - Generando un catálogo de requisitos generales.
  - Describiendo el sistema mediante modelos iniciales de alto nivel.

Se identifica a los usuarios que participarán en el proceso de análisis, determinando sus perfiles y responsabilidades, así como el grado de dedicación que van a necesitar y, por último, se elabora el plan de trabajo a seguir.

A lo largo de esta actividad se van produciendo:

-
  - Un catálogo de requisitos.
  - Un glosario de términos.
  - El contexto del sistema.
  - Un modelo conceptual de datos.
  - Una descripción general del entorno tecnológico.
  - Un catálogo de normas.
  - Un catálogo de usuarios.
  - Un plan de trabajo.

Se utilizan técnicas tales como: sesiones de trabajo, técnicas de catalogación, DFD y modelo entidad/relación (E/R) extendido. Naturalmente, todo en muy alto nivel y en fase de trabajo inicial.

- En **ASI 2, establecimiento de requisitos,** se realiza principalmente la definición de requisitos del nuevo sistema. Su objetivo es formar un catálogo de requisitos detallado que permita describir con precisión el sistema de información y que, además, sirva de base para comprobar que es completa la especificación de los modelos obtenidos en otras actividades.

También se identificarán los **requisitos no funcionales,** es decir, las facilidades que ha de proporcionar el sistema y las restricciones a las que estará sometido en cuanto a rendimiento, frecuencia de tratamiento, seguridad y control de acceso, etc.

Para el perfeccionamiento y la obtención de los requisitos se toma como punto de partida el catálogo y los modelos elaborados en la actividad anterior, que se completarán con sesiones de trabajo con los usuarios, cuyo objetivo será conseguir la información necesaria para la especificación detallada del nuevo sistema.

Como técnica se emplea principalmente la de «casos de uso», que facilita la **comunicación con los usuarios,** y el análisis orientado a objetos, que constituye la base de la **especificación,** siendo por tanto obligatoria.

- El SI se estructura en subsistemas en **ASI 3, identificación de subsistemas de análisis,** para facilitar la especificación de los distintos modelos y la traza o seguimiento de los requisitos. Esto da como resultado un modelo de procesos y se utiliza la técnica DFD.

Es simultánea a las siguientes y a la anterior, aunque comienza posteriormente a ella y antes que las siguientes. Efectivamente, en paralelo se generarán los distintos **modelos** que servirán de base para el diseño.

En el caso del análisis **estructurado,** se procederá a la elaboración y descripción detallada del modelo de datos y de procesos y, en el caso de un análisis **orientado a objetos,** del modelo de clases y el comportamiento dinámico, mediante el análisis de casos de uso.

- **ASI 4, análisis de casos de uso,** trata de obtener el modelo de clases y un diagrama de interacción de objetos con técnicas de UML. Es una actividad exclusiva del enfoque orientado a objetos.
- **ASI 5, análisis de clases,** continúa elaborando y perfeccionando el modelo de clases. Es también exclusiva de la orientación a objetos.
- **ASI 6, elaboración del modelo de datos,** es una actividad exclusiva del enfoque estructurado. Se trata de obtener los modelos: conceptual, lógico y lógico normalizado, empleando técnicas de entidad/relación extendida y la normalización.
- En **ASI 7, elaboración del modelo de procesos,** se emplean técnicas de DFD y matriciales para conseguir la matriz de procesos/localización geográfica. Se describen también las interfaces con otros sistemas. Es una actividad específica del enfoque estructurado.
- En **ASI 8, definición de interfaces de usuario,** se especifican todas las interfaces entre el sistema y el usuario, como formatos de pantalla, diálogos, formatos de informes y formularios de entrada.

Se emplea el prototipado, diagramas de transición de estados o diagramas de interacción de objetos. Es una actividad **común a ambos enfoques.**

- En la actividad **ASI 9, análisis de consistencia y especificación de requisitos,** se realiza una validación y verificación de los modelos con el fin de asegurar que son:
  - **Completos:** cada modelo obtenido contiene la información necesaria recogida en el catálogo de requisitos.
  - **Consistentes:** cada modelo es coherente con el resto de las modelos.
  - **Correctos:** cada modelo sigue criterios de calidad predeterminados respecto a la técnica utilizada, calidad de los diagramas, elección de nombres, normas de calidad, etc.

Como resultado de esta actividad se elabora la **especificación de requisitos del *software*** (ERS), producto que constituye un punto de referencia en el desarrollo del *software* para formalizar las peticiones de cambio sobre los requisitos inicialmente especificados (UCM, 2008; IEEE, 1998).

- En la actividad **ASI 10, especificación del plan de pruebas,** se establece el marco general del plan de pruebas y se inicia su especificación, que se irá completando en los restantes procesos de desarrollo.

- Por último, en **ASI 11, presentación y aprobación del análisis del sistema de información,** se realiza la presentación del análisis del sistema de información al comité de dirección para la aprobación final del mismo.

Diseño del sistema de información (DSI)

El objetivo del proceso DSI es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

Basándose en los productos resultantes del diseño se generan las especificaciones de:

- La construcción del sistema.

- Las especificaciones técnicas del plan de pruebas.

- La definición de los requisitos de implantación.

- Los procedimientos de migración y carga inicial.

El diseño comprende tres grandes bloques de actividades (ver Figura 8):

- Definición de la arquitectura del sistema.

- Generación de especificaciones para la construcción del SI.

- Presentación y aprobación del resultado de los trabajos de diseño.

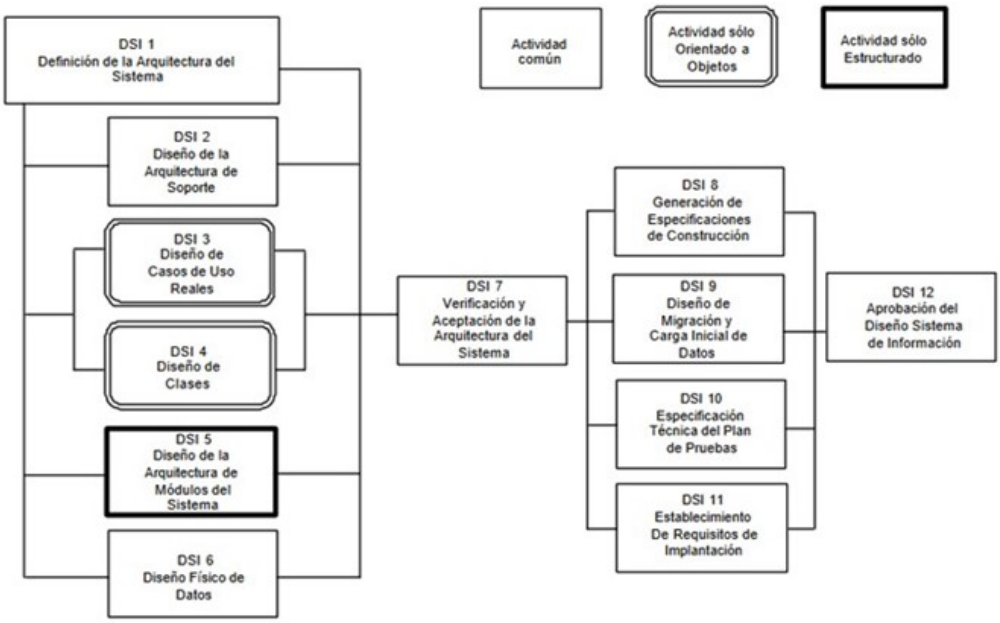


Figura 8. Secuencia de actividades del DSI. Fuente: Consejo Superior de Informática, 2001.

- El **primer bloque** es el más complicado e incluye actividades comunes al enfoque estructurado y al orientado a objetos, y que resultan específicas de cada uno de ellos. En él se establece la fragmentación física del SI, se realiza la organización en subsistemas de diseño y la especificación del entorno tecnológico, y se definen los requisitos de operación, administración, seguridad y control de acceso.

También se elabora un catálogo de **excepciones del sistema,** en el que se registran las situaciones de funcionamiento secundario o anómalo y todo ello se utiliza como referencia en la especificación técnica de las pruebas del sistema.

Al final de este bloque se hace la verificación de la arquitectura del sistema y un análisis de consistencia de las especificaciones de diseño. Esta es una actividad cuyas tareas son más bien de **aseguramiento de la calidad** que de diseño propiamente dicho.

- En el **segundo bloque** de actividades se generan todas las especificaciones

necesarias para la construcción del sistema de información:

- Elaboración de las especificaciones de construcción de los componentes del sistema, así como de las estructuras de datos.
- Diseño de la migración o carga de los datos.
- Especificación técnica del plan de pruebas.
- Especificación de los requisitos de implantación del sistema.

- Finalmente, en el **tercer bloque,** se hace una presentación formal del diseño, que debe ser aprobada por los usuarios.

Durante el proceso de diseño se transforma el modelo conceptual del análisis en el **modelo lógico de implementación.** El diseño está orientado a la obtención de las especificaciones de construcción a través de la elaboración del modelo lógico. Estas especificaciones son el documento base para los programadores del sistema de información.

Construcción del sistema de información (CSI)

Este proceso tiene como objetivo final la **construcción y prueba** de los distintos componentes del SI, a partir del conjunto de especificaciones lógicas y físicas del mismo, obtenido en el DSI. Se desarrollan los procedimientos de operación y seguridad, y se elaboran los manuales de usuario final y de explotación, con el fin de asegurar el correcto funcionamiento del sistema para su posterior implantación.

Para conseguir dicho objetivo, en este proceso se prepara el entorno de construcción, se genera el código de cada uno de los componentes del SI y se realizan las pruebas unitarias, las pruebas de integración de los subsistemas y componentes, y las pruebas del sistema, de acuerdo con el plan de pruebas establecido. Asimismo, se define la formación de usuario final y, si procede, se construyen los procedimientos de migración y carga inicial de datos.

Las actividades que engloba este proceso se recogen en la Figura 9.

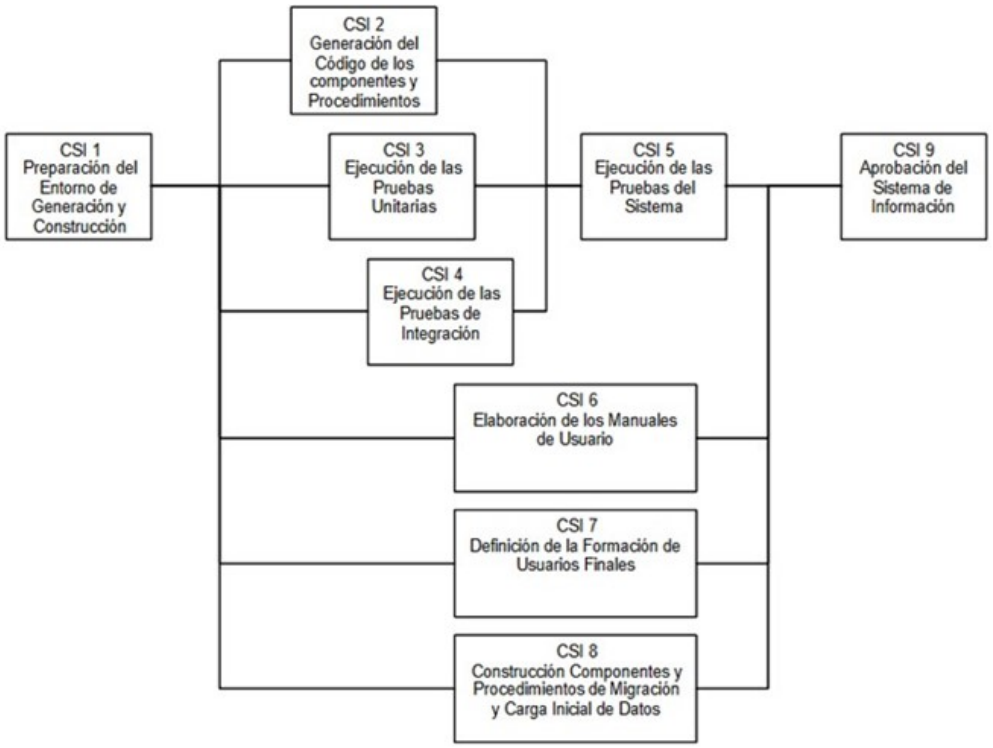


Figura 9. Secuencia de actividades del CSI. Fuente: Consejo Superior de Informática, 2001.

Implantación y aceptación del sistema (IAS)

El proceso IAS tiene como objetivo principal la **entrega y aceptación** del sistema en su totalidad y la realización de todas las actividades necesarias para su paso a producción. La relación de las actividades del IAS se muestra en la Figura 10.

En primer lugar, se revisa la estrategia de implantación que ya se determinó en el EVS. Se estudia su alcance y, en función de sus características, se define un **plan de implantación** y se especifica el equipo que lo va a llevar a cabo. Hay que destacar la participación del usuario de operación en las pruebas de implantación, del usuario final en las pruebas de aceptación y del responsable de mantenimiento, en su caso.

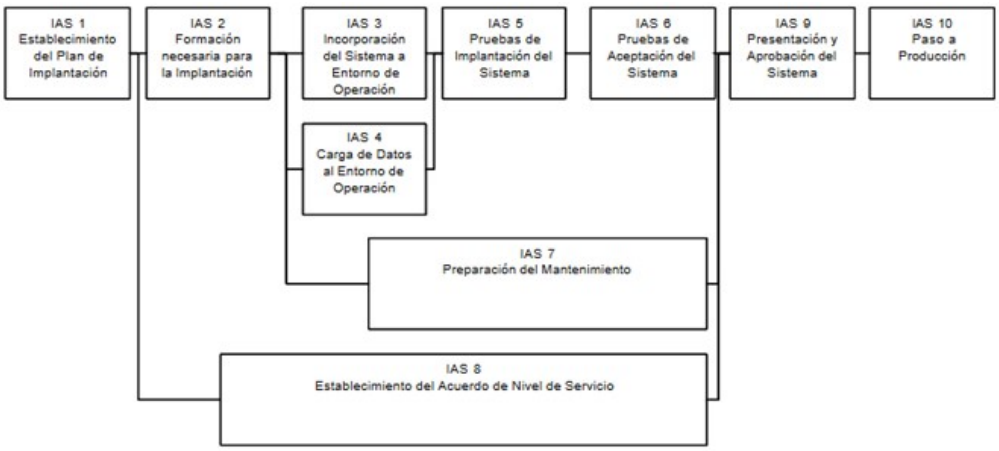


Figura 10. Secuencia de actividades del IAS. Fuente: Consejo Superior de Informática, 2001.

Las actividades previas al inicio de la producción incluyen:

- La preparación de la infraestructura necesaria para configurar el entorno.
- La instalación de los componentes.
- La activación de los procedimientos manuales y automáticos asociados.
- Cuando proceda, la migración o carga inicial de datos.

Para ello se tomarán como punto de partida los **productos *software* probados** (obtenidos en el CSI, con su documentación asociada).

Se realizan las pruebas de implantación y aceptación del sistema en su globalidad, que responden a los siguientes propósitos:

- Las **pruebas de implantación** cubren un rango muy amplio que va desde la comprobación de cualquier detalle de diseño interno hasta aspectos tales como las comunicaciones. Se debe comprobar que el sistema puede gestionar los volúmenes de información requeridos, se ajusta a los tiempos de respuesta deseados y que los procedimientos de respaldo, seguridad e interfaces con otros sistemas funcionan correctamente. Se debe comprobar el comportamiento del sistema bajo las condiciones más extremas.

- Las **pruebas de aceptación** se realizan por y para los usuarios, y tienen como objetivo validar formalmente que el sistema se ajusta a sus necesidades.

Asimismo, se llevarán a cabo las tareas necesarias para la preparación del **mantenimiento,** siempre y cuando se haya decidido que los sistemas de información implicados en la implantación van a ser objeto de mantenimiento. En cualquier caso, es necesario que la persona que vaya a asumir el mantenimiento **conozca el sistema** antes de su incorporación al entorno de producción.

Además, se determinan los **servicios** que requiere el sistema que se va a implantar, especificando los niveles de servicio y el acuerdo que se adquiere una vez que se inicia la producción. Se distinguen:

- Los **servicios de gestión de operaciones** (servicios por lotes, seguridad, comunicaciones, etc.).

- Los **servicios al cliente** (servicio de atención al usuario, mantenimiento, etc.), que se deberán negociar en cuanto a recursos, horarios, coste, etc. Se fija el nivel con el que se prestará el servicio como indicador de su calidad.

Conviene señalar que la implantación puede ser un **proceso iterativo** que se realice de acuerdo con el plan que se establezca para el comienzo de la producción del sistema en su entorno de operación.

Para establecer este plan se tendrá en cuenta:

- El cumplimiento de los requisitos de implantación definidos en la actividad «establecimiento de requisitos» (ASI 2) y especificados en la actividad «establecimiento de requisitos de implantación» (DSI 11).

- La estrategia de transición del sistema antiguo al nuevo.

Finalmente, se efectúan las acciones que sean necesarias para el inicio de la producción.

Mantenimiento de sistemas de información (MSI)

El objetivo de este proceso es la obtención de una **nueva versión del SI,** a partir de las peticiones de mantenimiento que los usuarios realizan con motivo de un problema detectado en el sistema o por la necesidad de una mejora del mismo.

En métrica v.3 se tiene en cuenta un **mantenimiento correctivo,** es decir, cambios precisos para corregir errores del producto *software*, y un **mantenimiento evolutivo,** es decir, incorporaciones, modificaciones y eliminaciones necesarias en un producto *software* para cubrir la expansión o cambio en las necesidades del usuario.

Ante la petición de **cambio de un SI ya en producción,** se realiza un registro de las peticiones, se diagnostica el tipo de mantenimiento y se decide si se le da respuesta o no (en función del plan de mantenimiento asociado al sistema afectado por la petición), y se establece con qué prioridad.

La definición de la **solución al problema** o necesidad planteada por el usuario, que realiza la labor de responsable de mantenimiento, incluye un estudio del impacto, la valoración del esfuerzo y coste, las actividades y tareas del proceso de desarrollo a realizar (ver Figura 11) y el plan de pruebas de regresión.

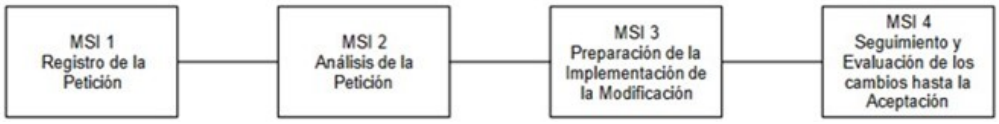


Figura 11. Secuencia de actividades del MSI. Fuente: Consejo Superior de Informática, 2001.

2.6. Microsoft solution framework (MSF)

Se trata de una técnica flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso para el desarrollo de software, que controla la planificación, el desarrollo y la gestión de proyectos tecnológicos.

Se centra en el **modelo de procesos y de equipo,** dejando en un segundo plano las elecciones tecnológicas (Turner, 2006).

A diferencia de una metodología prescriptiva, MSF proporciona un **marco flexible y escalable** para satisfacer las necesidades de cualquier organización de tamaño o equipo de proyecto. Consta de principios, modelos y disciplinas para gestionar a las personas, los procesos y los elementos tecnológicos que se encuentran la mayoría de los proyectos.

MSF fue desarrollado para **maximizar el éxito** de los proyectos de desarrollo de *software* a lo largo de todo su ciclo de vida. Fue creado en base a la experiencia adquirida en Microsoft en proyectos de desarrollo de *software* y a las mejores prácticas de la industria informática (Microsoft, 2007).

MSF se basa en ocho principios fundamentales que sirven de guía para desarrollar una solución de calidad:

- Fomentar una comunicación abierta.
- Trabajar hacia una visión compartida.
- Empoderar a los miembros del equipo.
- Establecer responsabilidades claras y compartidas.
- Centrarse en entregas que den valor al negocio.

- Mantenerse ágil, esperar cambios y adaptarse.
- Invertir en calidad.
- Aprender de todas las experiencias.

MSF se compone de varios modelos para el desarrollo de un proyecto *software:*

- **Modelo de equipo de trabajo:** tiene como finalidad mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto, asignando roles y responsabilidades a cada miembro del equipo con el objetivo de satisfacer los objetivos del proyecto. Puede ser escalado, dependiendo del tamaño del proyecto y del equipo de personas disponibles. Se espera que los miembros del equipo compartan una visión común del proyecto, un enfoque en la implementación del proyecto, altos estándares de calidad y comunicación, y una buena disposición para aprender. En la Figura 12 se muestran los grupos de roles del modelo de equipo de MSF.

Figura 12. Roles del modelo de equipo de MSF. Fuente: Microsoft, 2007.

- **Modelo de procesos:** cubre el ciclo de vida de una solución. Define una secuencia de actividades para construir e implementar soluciones. Diseñado para mejorar el control del proyecto minimizando el riesgo y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto describiendo las fases, las actividades, la liberación de versiones y

explicando su relación con el modelo de equipo. Combina el modelo en cascada, que enfatiza el logro de hitos, y el modelo en espiral, que se centra en la necesidad continua de refinar los requisitos y estimaciones de un proyecto. Ayuda a los equipos de proyecto a enfocarse en el valor comercial del cliente, porque no se obtiene ningún valor hasta que la solución se implementa y se pone en funcionamiento.

- **Modelo de arquitectura:** tiene como finalidad reducir la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

- **Modelo de diseño de proceso:** proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

- **Modelo de aplicación:** tiene como finalidad mejorar el desarrollo, el mantenimiento y el soporte, proporcionando un modelo de tres niveles para diseñar y desarrollar aplicaciones *software.* Los servicios utilizados en este modelo son escalables y pueden ser usados en un solo ordenador o incluso en varios servidores.

Junto a los modelos MSF utiliza **tres disciplinas clásicas,** que son: la gestión de riesgos, la gestión de preparación y la gestión de proyectos. Estas disciplinas se reflejan tanto en el modelo de proceso como en las responsabilidades de roles definidas en el modelo de equipo.

- **Gestión del riesgo:** aboga por la gestión proactiva de riesgos, la evaluación continua de riesgos y la integración en la toma de decisiones durante todo el proyecto y el ciclo de vida. Define seis pasos lógicos que el equipo utiliza para gestionar los riesgos actuales, planificar y ejecutar estrategias de gestión de riesgos, y capturar conocimiento para la empresa: identificar, analizar y priorizar, planificar y programar, rastrear e informar, controlar y aprender.

- **Gestión de preparación:** define la preparación como una medida del estado actual frente al estado deseado de conocimiento, habilidades y capacidades de las personas en una organización. Esta medición es la capacidad real o percibida de los individuos en cualquier momento durante el proceso continuo de planificación, construcción y administración de soluciones.

- **Gestión de proyectos:** para entregar una solución dentro de las limitaciones del proyecto, es esencial contar con fuertes habilidades de gestión de proyectos. El modelo de equipo de MSF no contiene una función conocida como gestor de proyectos, pero la mayoría de las funciones de gestión de proyectos las realiza el rol de gestión de programas de MSF. Esta disciplina es un conjunto de habilidades y técnicas que incluyen la planificación integrada para cada aspecto del proyecto, la realización de control de cambios, la definición y gestión del alcance del proyecto, la preparación de un presupuesto y gestión de costes, la preparación y seguimiento de una planificación, la obtención de los recursos correctos asignados al proyecto, la gestión de contratos y proveedores, la facilitación de la gestión de riesgos, y la documentación y rastreo de la gestión de calidad del equipo.

Por último, las fases para el desarrollo de un proyecto en MSF son (ver Figura 13):

Figura 13. Fases de desarrollo en MSF. Fuente: Microsoft, 2007.

- **Previsión:** en esta fase el equipo identifica la visión y el alcance del proyecto mediante la preparación de un documento de visión y alcance. Durante esta fase se forman los objetivos para el proyecto y se crea una declaración de visión que define todo el proyecto. Esta visión compartida ayuda al equipo a trabajar hacia un objetivo común. El equipo del proyecto se reúne y los miembros del equipo se empoderan mediante la asignación de roles y responsabilidades. Otra actividad importante en esta fase es la identificación de riesgos y la preparación de planes de mitigación y contingencia.

- **Planificación:** durante esta fase se evalúa el entorno existente para formar un diseño de solución. La actividad principal es hacer una planificación detallada para garantizar el éxito del proyecto. Para ello, se crea un entregable del plan maestro del proyecto que consta de todos los subplanes, como el plan de prueba y el plan de desarrollo. El plan se centra en los presupuestos, la calidad, el cronograma y la implementación técnica de la solución.

- **Desarrollo:** durante esta fase los componentes de la aplicación se transforman en función de los planes de desarrollo. Se desarrollan componentes que necesitan una reescritura. En los componentes se llevan a cabo cambios basados en el cumplimiento de la sintaxis, el rendimiento y la seguridad. El código fuente y los ejecutables ayudan a crear los entregables de esta etapa.

- **Estabilización:** durante esta fase todos los componentes se prueban colectivamente para garantizar que toda la solución funcione correctamente. Los criterios de prueba incluyen el logro de los requisitos deseados de funcionalidad, seguridad y rendimiento. La base de datos y los componentes probados forman los entregables de esta etapa. Una vez que se completan las pruebas, la solución debe estabilizarse para garantizar que cumpla con los niveles de calidad definidos.

- **Implantación:** en esta fase el equipo implanta la solución y se asegura de que sea estable y se pueda usar. La solución probada se traslada a producción y se transfiere a operaciones. Durante esta fase los componentes de la solución se

ajustan en el entorno de producción. Se obtiene una aprobación de la solución de todas las partes interesadas que confirma que la aplicación cumple con los requisitos desarrollados durante las fases de previsión y planificación.

2.7. Referencias bibliográficas

Booch, G., Rumbaugh, J. y Jacobson, I. (2005). *Unified Modeling Language User Guide* (2ª ed.). Addison-Wesley.

Consejo Superior de Informática. (2001). *Métrica versión 3.* PAE. https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.W1IuUNgzbUI.

Hanssen, G. K., Westerheim, H. y Bjørson, F. O. (2005). Tailoring RUP to a defined project type: A case study. Actas de 6th International Conference on Product-Focused Software Process Improvement, PROFES, Oulu, Finland. *Springer-Verlag*, 3547, 314-327.

IEEE. (1998). IEEE 830-1998 - IEEE *Recommended practice for software requirements specifications* [Standard]. The Institute of Electrical and Electronics Engineers. https://standards.ieee.org/standard/830-1998.html.

Jacobson, I., Booch, G. y Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.* Addison Wesley.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction* (3ª ed.). Addison-Wesley.

Microsoft. (2007). Chapter 1 - Introduction to the Microsoft Solutions Framework. En *Oracle on UNIX to SQL Server on Windows Migration Guide.* https://docs.microsoft.com/en-us/previous-versions/tn-archive/bb497060(v=technet.10)?redirectedfrom=MSDN.

Sommerville, I. (2005). *Ingeniería del Software* (7ª ed.). Pearson Addison-Wesley.

Turner, M. S. (2006). *Microsoft Solutions Framework Essentials*. Microsoft Press.

UCM. (2008). *Especificación de requisitos según el estándar IEEE 830-1998*. Universidad Complutense de Madrid. https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.

El proceso unificado de desarrollo del software

Jacobson, I., Booch, G. y Rumbaugh, J. (2000). E*l proceso unificado de desarrollo de software.* Addison Wesley. https://dialnet.unirioja.es/servlet/libro?codigo=198782.

El libro de referencia sobre el proceso unificado, especialmente para miembros del equipo de desarrollo. Este presupone un conocimiento básico de diseño orientado a objetos.

Métrica v.3

Consejo Superior de Informática. (2001). *Métrica versión 3.* PAE. https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.W1IuUNgzbUI.

Métrica v.3 es una metodología desarrollada por el Ministerio de Hacienda y Administraciones Públicas, que proporciona a las empresas y organizaciones unas directrices para sistematizar las actividades que dan soporte al ciclo de vida del *software.*

Ivar Jacobson International

Página de Ivar Jacobson International.

Sitio comercial de uno de los creadores del proceso unificado de desarrollo de *software,* en el que se incluye bastante información relacionada con el desarrollo de *software.*

Test

1. ¿Cuál de estas no es una característica principal de las técnicas tradicionales?

A. Son técnicas predictivas.

B. Se basan en una planificación detallada y en un contrato estricto.

C. Se adaptan rápidamente al cambio.

D. Se basan en poca documentación y en un trabajo colectivo.

2. ¿Cuál de las siguientes es una característica clave del proceso unificado?

A. Está centrado en la arquitectura.

B. Sigue un modelo iterativo e incremental.

C. Está dirigido por casos de uso.

D. Todas las respuestas anteriores son correctas.

3. Uno de los modelos más importantes en la fase de inicio del proceso unificado es:

A. El modelo de dominio.

B. El modelo de clases de diseño.

C. El modelo de componentes.

D. El modelo de interacción.

4. ¿Bajo qué perspectivas se trabaja en RUP?

A. Estática, dinámica y estética.

B. Estática, práctica e incremental.

C. Estática, práctica y dinámica.

D. Dinámica, práctica, estética e incremental.

5. El proceso unificado de rational:

A. Únicamente tiene en cuenta una perspectiva estática y otra dinámica.

B. Es una mejora con respecto al *essential unified process* (EssUP).

C. Implementa el *software* en base a artefactos.

D. Implementa el *software* reutilizando componentes.

6. El objetivo del proceso de diseño del sistema de información, en métrica v.3, es la obtención de un marco de referencia para el desarrollo de sistemas de información que responda a los objetivos estratégicos de la organización:

A. Verdadero.

B. Falso.

C. Verdadero, en el caso de que no fuera preciso realizar el estudio de viabilidad.

D. Falso, solo en el caso en el que no haya objetivos estratégicos.

7. ¿Cuál es el objetivo principal del proceso de implantación y aceptación del sistema en métrica v.3?

A. La entrega y aceptación del sistema de una parte concreta.

B. Estudiar la visión de los usuarios previamente a la entrega del sistema.

C. La entrega y aceptación del sistema en su totalidad y realizar el paso a producción.

D. Entregar el producto para que el usuario lo pruebe.

8. Métrica v.3 contempla tanto desarrollos estructurados como orientados a objetos:

A. Verdadero.

B. Falso.

C. Falso, es la versión 2 la que permite el doble enfoque.

D. Verdadero, pudiendo variar la orientación de las técnicas en el análisis y el diseño.

9. ¿Cuál es un principio fundamental de MSF?

A. Establecer responsabilidades claras y compartidas.

B. Adaptarse al cambio no es una prioridad.

C. Centrarse en entregas que no den valor al cliente.

D. Trabajar de forma individual.

10. ¿Cuál de las siguientes afirmaciones es correcta?

A. Las fases para el desarrollo de un proyecto en MSF son previsión, planificación, desarrollo, estabilización e implementación.

B. El modelo de equipo de MSF proporciona una estructura flexible para organizar los equipos de un proyecto, asignando roles y responsabilidades a cada miembro del equipo con el objetivo de satisfacer los objetivos del proyecto.

C. La disciplina de gestión de riesgos de MSF aboga por la gestión proactiva de riesgos, la evaluación continua de riesgos y la integración en la toma de decisiones durante todo el proyecto y el ciclo de vida.

D. Todas las respuestas anteriores son correctas.