

PROCESOS EN LA INGENIERÍA DEL SOFTWARE

© José Ramón Hilera González, 2006



Índice:

1. Objetivos	2
2. Definición de proceso	3
2.1. Proceso definido, metodología y ciclo de vida	3
2.2. Tipos de procesos en la Ingeniería del Software	4
3. Procesos principales	5
3.1. Adquisición del software	6
3.2. Suministro del software	7
3.3. Desarrollo del software	7
3.4. Operación del software	15
3.5. Mantenimiento del software	16
4. Procesos de apoyo	17
4.1. Documentación del software	17
4.2. Gestión de la configuración	18
4.3. Aseguramiento de la calidad	18
4.4. Verificación del software	19
4.5. Validación del software	19
4.6. Revisiones conjuntas	20
4.7. Auditoría del software	20
4.8. Resolución de problemas	21
5. Procesos organizacionales	21
5.1. Gestión de proyectos software	21
5.2. Infraestructura del software	23
5.3. Mejora del software	23
5.4. Formación	24
6. Estándares sobre procesos en la Ingeniería del Software	24
6.1. ISO/IEC/IEEE 12207	24
6.2. IEEE 1074	25
7. Madurez del proceso software	29
7.1. CMMI (Capability Maturity Model Integration)	29
7.2. ISO 14504 (SPICE)	33
7.3. ISO 9001/ISO 9000-3	36
8. Metodologías de Ingeniería del Software	37
8.1. Tipos de metodologías	38
8.2. Metodología “Métrica”	43
8.3. Metodología “Proceso Unificado”	49
9. Modelado y automatización del proceso software	52
9.1. Modelado del proceso software	55
9.2. Entornos de automatización del proceso software	57

Procesos en la Ingeniería del Software (José R. Hilera, 2006)

UNIVERSIDAD DE ALCALÁ

Processes are like habits: hard to establish and even harder to break.
Humphrey, W (Humphrey 1995).

1 Objetivos

El objetivo general de este documento es poner de manifiesto la necesidad de aplicar un enfoque basado en procesos a la Ingeniería del Software, concibiendo cada una de las actividades implicadas en el ciclo de vida del software como parte de procesos definidos que son descritos y comunicados para su aplicación disciplinada en cada uno de los proyectos de desarrollo o mantenimiento de software que vaya a llevar a cabo una organización.

Más concretamente, este documento pretende que el lector sea capaz de lo siguiente:

- Entender los conceptos de proceso y proceso definido, y su relación con otros conceptos fundamentales de la Ingeniería del Software, como Ciclo de Vida y Metodología.
- Conocer los diferentes tipos de procesos que pueden establecerse en los proyectos de Ingeniería del Software.
- Conocer la existencia de estándares y recomendaciones sobre los procesos a considerar en la Ingeniería del Software, y determinar la relación entre ellos.

2 PROCESOS EN LA INGENIERÍA DEL SOFTWARE (JOSÉ R. HILERA, 2006)

- Diferenciar los posibles niveles de madurez que pueden atribuirse a los procesos definidos en las organizaciones de desarrollo de software.
- Conocer la existencia de diferentes tipos de metodologías de Ingeniería del Software que permiten aplicar el enfoque de proceso en los proyectos de desarrollo y mantenimiento que llevan a cabo las organizaciones de software.
- Entender el concepto de Ingeniería del Software Asistida por Computador (CASE) y saber diferenciar las diferentes categorías de herramientas y entornos de desarrollo automatizados.
- Interpretar modelos de procesos descritos con técnicas estándar de modelado y realizar modelos básicos.

2 Definición de proceso

El concepto de proceso aplicado a la Industria no es exclusivo del ámbito de la Ingeniería del Software (o Industria del software), es un término que empezó a utilizarse a mediados de la década de los 80 del siglo XX en el ámbito empresarial, para diferenciar este nuevo enfoque de negocio del enfoque tradicional orientado a las tareas. Los principales precursores de esta nueva forma de pensar llamada "*Thinking process*" fueron Hammer y Champy, que establecieron diferentes pautas para mejorar la forma de trabajo de las organizaciones aplicando lo que denominaron "reingeniería de procesos de negocio" (Hammer 1990).

Por tanto, en el ámbito que nos ocupa, el concepto que realmente se utiliza no es el de proceso en general, sino el de proceso de negocio, teniendo en cuenta, en este caso que el negocio es la producción de software, por lo que habitualmente el término que más aparece en la bibliografía especializada es el de "proceso software" (abreviatura de lo que debería ser "proceso de negocio de producción de software").

Un proceso de negocio es una colección de actividades que recibe unas entradas y genera unas salidas con valor para el cliente. A diferencia del enfoque de tareas, el enfoque de procesos considera que las tareas que se desarrollan en una organización forman parte de un proceso definido e institucionalizado, y han de estar integradas y sincronizadas para conseguir un objetivo común: el del proceso del que forman parte. La integración de las tareas en procesos evita situaciones de solapamiento o descoordinación entre las tareas si éstas se realizarán de forma independiente.

Estas ideas fueron recogidas y aplicadas por Humphrey en el ámbito del software. En opinión de este importante autor, para resolver los problemas

relacionados con la producción de software, un primer paso debería ser tratar las tareas como un proceso que pueda ser controlado, medido y mejorado. (Humphrey 1987). Para este autor, un proceso software es "el conjunto de actividades, métodos y prácticas usadas en la producción y evolución del software".

El concepto de proceso no puede considerarse de forma independiente, sino vinculado a otros conceptos, como actividad, ciclo de vida, etc. Se han creado ontologías para conceptualizar el conocimiento relacionado con la Ingeniería del software en general, y sobre el procesos software en particular, con el objetivo de compartir un vocabulario común (Ruiz, Calero & Piattini 2005). Estas ontologías incluyen la definición de términos como "Actividad", "Modelo de Ciclo de Vida", "Paradigma", "Tecnología de Desarrollo", "Recurso", "Artefacto", o "Procedimiento"; así como la formalización de las relaciones entre ellos, estableciendo, por ejemplo, que un "proceso software" se compone de "Actividades", se ajusta a una "Tecnología", o hace referencia a un "modelo de Ciclo de Vida".

Las ontologías sobre procesos también se pueden complementar con otras sobre las habilidades de los Ingenieros de Software, o sobre conocimiento extraído de la experiencia de proyectos de desarrollo concretos, para obtener una representación del conocimiento en este campo lo más completa posible y que pueda ser de utilidad para el estudio de esta disciplina o para su aplicación en futuros proyectos.

2.1 Proceso definido, metodología y ciclo de vida

Aunque está asumido implícitamente en la propia definición de proceso software, es conveniente aclarar que para que un proceso software sea de utilidad debe tratarse realmente de un "proceso software definido" (también denominado "proceso estándar"), aunque habitualmente no se añade esta última palabra al término. Ello quiere decir que el proceso ha de estar documentado, es decir, descrito en forma de documento o modelo, para que pueda ser transmitido a todos los participantes en los proyectos de desarrollo de software en los que se aplicará dicho proceso.

En este sentido, el término proceso software definido podría equivaler semánticamente al de metodología de desarrollo de software, siempre que la descripción del proceso establezca también las técnicas a aplicar en las diferentes actividades contempladas por el mismo, ya que todas las metodologías sí lo hacen.

Otro término muy vinculado al de proceso es el de ciclo de vida, en este caso es necesario considerar que todo proceso puede descomponerse en otros procesos que serían subprocesos de aquél. Teniendo esto en cuenta, el concepto de ciclo de vida del software es un término que abarca todos los procesos implicados en el desarrollo, gestión, mantenimiento y, en general, todas las acciones que ocurren

sobre el software desde su concepción hasta el momento que deja de utilizarse. Por ese motivo, se han creado estándares, como ISO 12207 (ISO 1995a), que sugieren qué tipo de procesos han de considerarse en el ciclo de vida, como el proceso de adquisición, proceso de desarrollo, proceso de mantenimiento, o proceso de gestión, entre otros. En este caso, estos procesos podrían suponerse que son subprocesos del proceso software en general, que abarcaría a todos. También pueden definirse procesos software que sólo consideren parte de los sub-procesos recomendados. Por ejemplo, si consideramos el proceso software definido por la metodología de desarrollo Métrica (MAP 2001), o por la metodología conocida como "Proceso Unificado" (Jacobson, Booch & Rumbaugh 2000), en ambos casos no se consideran las actividades previstas en algunos sub-procesos del estándar ISO 12207, como el denominado "proceso de adquisición". En el caso de Métrica, sin embargo, sí se considera un proceso explícito de gestión de la configuración, mientras que en el Proceso Unificado no.

2.2 Tipos de procesos en la Ingeniería del Software

En relación con las actividades a realizar con el software, el desarrollo del mismo no es la única que debe considerarse, ya que existen otras tareas como la negociación del contrato con el cliente, la gestión de versiones, la formación de los usuarios, etc., que no son propiamente de desarrollo en el sentido de diseño, programación y prueba. En este apartado se describirán brevemente todos los tipos de procesos que deberían definirse en las organizaciones dedicadas a la Ingeniería del software, para lo cual nos basaremos en las recomendaciones establecidas en el más importante estándar publicado hasta la fecha en este sentido, se trata del estándar ISO 12207, denominado "Software life cycle processes", el cual establece "el conjunto de los procesos de la ingeniería del software que son fundamentales para una buena ingeniería del software" (ISO 1995a).

No todos los procesos que se van a describir deben ser definidos en todas las organizaciones, si bien cuantos más de ellos estén estandarizados, mayor será el nivel de madurez de la organización. Por este motivo, se va a realizar una primera división de los procesos en tres tipos: principales, de apoyo y organizacionales. Los primeros se refieren sobre todo a las actividades de desarrollo, los segundos a otras actividades no fundamentales pero que son importantes para conseguir software de calidad y fácil de mantener, y los últimos son los que deberían llevarse a cabo en relación con la gestión de los proyectos de software. En la figura 1.1 se representan todos estos procesos.

Los procesos que se van a describir están evidentemente relacionados entre sí, pero no existe una ordenación en su realización, ni en las de las actividades que los componen. El objetivo es describir todas las actividades relacionadas con la ingeniería del software, pero agrupadas en conjuntos coherentes que serían los

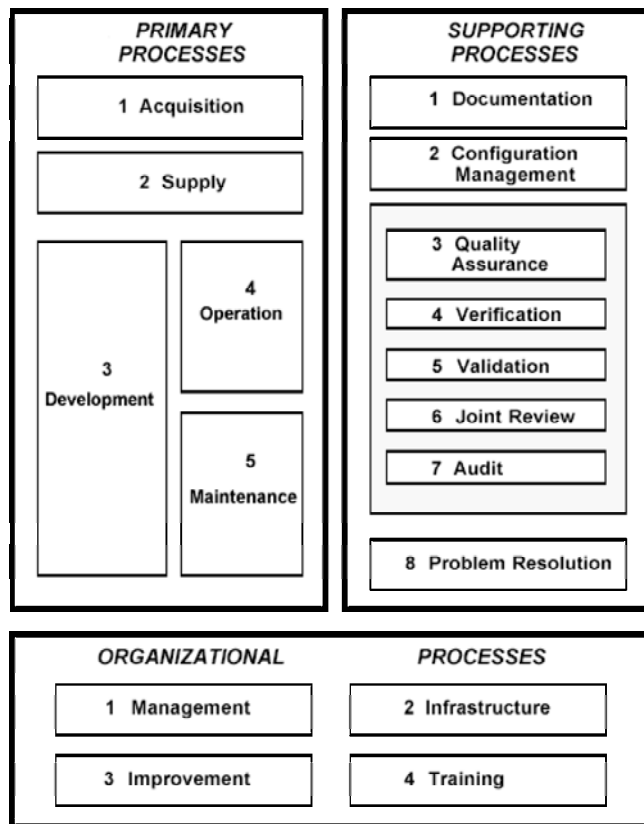


Figura 1.1: Procesos en la Ingeniería del Software (ISO 12207)

procesos. Es misión de las organizaciones establecer metodologías que definan con detalle cada uno de los procesos y su ordenación, así como las actividades que contienen, en un modelo de ciclo de vida.

3 Procesos principales

Los procesos principales son los que al menos habría que definir en una organización para poder desarrollar y mantener software con un enfoque de ingeniería del software. Se trata de procesos que incluirían las actividades relacionadas con la negociación de la adquisición y entrega del software entre cliente y proveedor, las actividades para desarrollar y mantener el software, así como las que se llevan a cabo durante su operación o funcionamiento. Según ISO, son los procesos siguientes: proceso de adquisición, proceso de suministro, proceso de desarrollo, proceso de operación, y proceso de mantenimiento. Aunque son los procesos fundamentales, su ejecución sería complicada sin la definición de otros procesos de

apoyo y organizacionales, que se describirán en los siguientes apartados.

3.1 Adquisición del software

El objetivo del proceso de adquisición es obtener el producto software que satisfaga las necesidades expresadas por el cliente. El proceso debe comenzar con la identificación de las necesidades de un cliente y la selección de un proveedor, y finalizar con la aceptación del producto.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Desarrollar un contrato que claramente exprese las expectativas, responsabilidades y compromisos del adquirente (cliente) y del suministrador (proveedor) del software.
- Obtener productos y/o servicios que satisfagan las necesidades del cliente.
- Gestionar la adquisición para que se cumplan las restricciones (ej. coste, calendario, calidad) y objetivos (ej., grado de reutilización de software).
- Establecer un compromiso de trabajo para ser realizado bajo contrato.
- Cualificar a los posibles suministradores mediante una valoración de su capacidad para realizar el software requerido.
- Seleccionar los suministradores cualificados para realizar las partes definidas en el contrato.
- Establecer y gestionar las obligaciones de y para el suministrador.
- Intercambiar regularmente información de progreso con el suministrador.
- Valorar el cumplimiento por parte del suministrador de los planes, estándares y procedimientos acordados.
- Valorar la calidad de los productos y servicios desarrollados por el suministrador.
- Establecer y ejecutar la estrategia y las condiciones o criterios de aceptación de los productos software o servicios que están siendo adquiridos.
- Establecer un medio por el cual el adquirente asumirá la responsabilidad sobre el producto software o servicio adquirido.

3.2 Suministro del software

El objetivo del proceso de suministro es proporcionar un producto al cliente que esté de acuerdo con los requisitos acordados. Las actividades de este proceso debería realizarlas principalmente la organización que actúa como suministrador del software. Este proceso puede ser iniciado bien por la decisión de preparar una oferta como respuesta a una solicitud del cliente, o por la firma del contrato para suministrar el producto o servicio software. El proceso ha de determinar los procedimientos y recursos necesarios para gestionar y asegurar la realización del proyecto, incluyendo la elaboración de un plan de desarrollo del proyecto.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Establecer una comunicación clara y permanente con el cliente.
- Definir de forma documentada y pactada los requisitos del cliente, así como los cambios necesarios.
- Establecer un mecanismo para detectar permanentemente las necesidades del cliente.
- Establecer un mecanismo para asegurar que los clientes puedan determinar fácilmente el estado y disposición de sus requisitos.
- Determinar los requisitos para la replicación, distribución, instalación y prueba del sistema que contiene el software, o el producto software.
- Empaquetar el sistema que contiene el software, o el producto software, de forma que facilite su eficiente y efectiva replicación, distribución, instalación, prueba y operación.
- Entregar al cliente el sistema que contiene el software, o el producto software, tal y como se ha establecido en los requisitos, e instalar de acuerdo con los requisitos.

3.3 Desarrollo del software

El objetivo del proceso de desarrollo es transformar un conjunto de requisitos en un producto software que cumpla las necesidades del cliente establecidas en un contrato. Este proceso contiene las actividades relacionadas con el análisis de requisitos, diseño, codificación, integración, prueba, e instalación y aceptación de los productos software.

Para llevar a cabo el desarrollo de software en un proyecto, se debe establecer previamente la metodología o estrategia de desarrollo que se va a seguir, la cual

a su vez establece un determinado modelo de ciclo de vida para el desarrollo del software (en cascada, incremental, evolutivo, etc.). Aunque las actividades implicadas en el desarrollo y su orden de realización depende de la metodología a seguir, ya que las define la propia metodología, a continuación se van a describir algunas de las actividades que deberían considerarse. En algunas metodologías estas actividades tienen un nombre diferente al que se va a utilizar, y también puede ocurrir que otras metodologías consideren unas integradas en otras. En cualquier caso, el enfoque de la ingeniería del software exige, al menos, que el desarrollo implique actividades de análisis de requisitos, diseño, codificación y prueba, e instalación. En las siguientes figuras, adaptadas de (IEEE 1998c), se representa la ordenación de estas actividades si se aplicasen las estrategias de desarrollo en cascada (figura 1.2), incremental (figura 1.3) o evolutiva (figura 1.4). En estas figuras *R* representa la actividad de *análisis de requisitos*, *D* el *diseño*, *C/T* la *codificación y prueba*, y *I/AS* la *instalación y aceptación* del software.

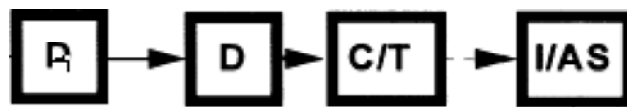


Figura 1.2: Estrategia de desarrollo en cascada

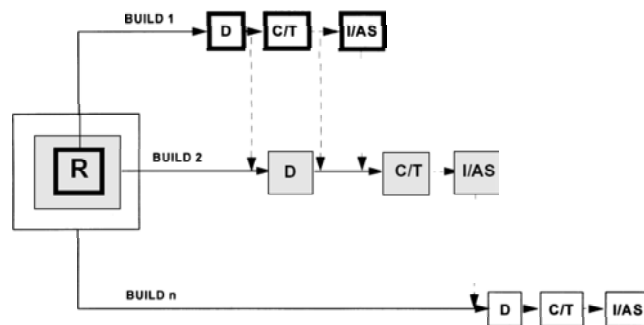


Figura 1.3: Estrategia de desarrollo incremental

De forma más detallada, si consideramos las recomendaciones de ISO e IEEE (ISO 1995a), las actividades en el proceso de desarrollo serían las siguientes:

- Análisis de requisitos del sistema.
- Diseño de la arquitectura del sistema.

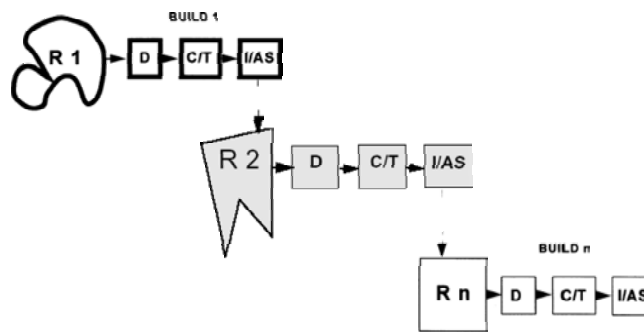


Figura 1.4: Estrategia de desarrollo evolutivo

- Análisis de requisitos del software.
- Diseño de la arquitectura del software.
- Diseño detallado del software.
- Codificación y prueba del software.
- Integración del software.
- Pruebas de cualificación del software.
- Integración del sistema.
- Prueba de cualificación del sistema.
- Instalación del software.
- Aceptación del software.

Como puede observarse, existen actividades que se refieren al software y otras al sistema; esto es así porque en la mayoría de los casos no se desarrolla únicamente software, sino software que va a formar parte de un sistema que, además de software, también se compone de hardware y de usuarios que realizan actividades manuales. Una definición ampliamente aceptada del término sistema¹ en el contexto de la Ingeniería del Software es la que se indica a continuación (ISO 1995a):

¹En este contexto, el término "Sistema" puede considerarse sinónimo de "Sistema de Información"

Definición
Un Sistema es an integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective.

Existen autores que consideran que la Ingeniería del Software implica todas las actividades anteriores excepto las que se referieren a la entidad denominada "sistema", que formarían parte de lo que se consideraría "Ingeniería de sistemas"; así, actividades como el análisis, diseño, integración y prueba del sistema serían de Ingeniería de sistemas y no de Ingeniería del software (Thayer 2002).

A continuación se realizará una breve descripción de cada una de estas actividades sugeridas por ISO y asumidas por IEEE, para tener una visión general de ellas, si bien será a lo largo de este libro donde se estudiarán con detalle tanto éstas como el resto de actividades del proceso software.

Análisis de requisitos del sistema

El desarrollo de un sistema basado en software debe comenzar con la realización de una especificación documentada de los requisitos que ha de satisfacer el sistema, de tal forma que una vez finalizado el mismo pueda utilizarse tal especificación para comprobar que todos y cada uno de los requisitos establecidos han sido implementados en el sistema. Una definición del término requisito es la incluida en el glosario del IEEE (IEEE 1990):

Definición
Por Requisito se entiende (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).

Los requisitos del sistema se han de definir para que reflejen las necesidades del cliente, y deben ser aprobados por todas las partes implicadas. Los requisitos que deben formularse pueden ser de varios tipos. En primer lugar se encontrarían los requisitos funcionales, que expresan las funciones que el usuario precisa del sistema a desarrollar; pero también pueden establecerse requisitos relacionados con la seguridad del sistema, con la ergonomía del mismo (incluidos los requisitos de usabilidad de las interfaces de usuario), con su mantenimiento, o con restricciones de diseño, entre otros.

Diseño de la arquitectura del sistema

Otra de las actividades que deben realizarse durante el desarrollo es el establecimiento de una arquitectura de alto nivel del sistema, que identifique los elementos hardware, software y de operación manual. Estos elementos constituyen la arquitectura del sistema, entendiendo como arquitectura la siguiente definición (IEEE 2000)

Definición
Arquitectura es the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

La arquitectura del sistema debe establecerse teniendo en cuenta los requisitos del sistema, asignando los requisitos definidos a los principales elementos de la arquitectura del sistema.

Análisis de requisitos del software

Los requisitos del software son los correspondientes a los elementos software que constituyen la arquitectura de un sistema basado en software. En los proyectos de desarrollo es necesario establecer y documentar con el mayor detalle posible los requisitos del software para garantizar que el software es el que realmente el usuario necesita y está perfectamente integrado con el resto de elementos del sistema. Se trata, en definitiva, de definir los requisitos asociados a los componentes software del sistema y a sus interfaces para cubrir las necesidades del cliente. Se deben desarrollar requisitos software que sean correctos y verificables, teniendo en cuenta su impacto en el entorno operativo. Además, es conveniente asignar prioridades a los requisitos y establecer relaciones de dependencia entre ellos, lo cual afectará a la estrategia de liberación o entrega del software que se vaya a considerar en un proyecto determinado.

Diseño de la arquitectura del software

2

El Diseño del software es la actividad fundamental del proceso de desarrollo, y suele descomponerse en dos componentes o niveles: **diseño de alto nivel o arquitectónico** (o preliminar, que se describe aquí) y el **diseño detallado** (que se describe en la siguiente sección). En un proyecto de desarrollo de software

²No hay una definición universalmente aceptada del término "software architecture"; el SEI dispone de un sitio Web en el que se publica un gran número de definiciones diferentes para el término, y en la que se pueden proponer otras: <http://www.sei.cmu.edu/architecture/definitions.html>

con enfoque de ingeniería es fundamental realizar un diseño arquitectónico del software, que supone decidir "la organización del software del sistema, la selección de los elementos estructurales (y sus interfaces) de los que se compone el software, junto con su comportamiento tal y como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas cada vez mayores y el estilo arquitectónico que orienta esta organización (esos elementos y sus interfaces, sus colaboraciones y su composición)" (Booch, Rumbaugh & Jacobson 1999).

Durante el diseño, se debe, en definitiva, seleccionar y documentar los componentes más eficaces y eficientes que juntos implementarán los requisitos del software, por lo que es importante establecer y documentar la asignación de los requisitos del software definidos a los componentes de la arquitectura del software. Además se deben especificar las interfaces de cada agrupación de componentes con el exterior y entre los componentes que forman la agrupación. También se debe proceder al diseño de alto nivel de los recursos de información que utilizará el software, como bases de datos o ficheros. Finalmente, es importante también definir y documentar los requisitos de las pruebas y planificar la integración del software.

Diseño detallado del software

A diferencia del Diseño arquitectónico anterior (o diseño de alto nivel o preliminar), que establece los componentes del software, el diseño detallado se refiere precisamente al diseño documentado de cada uno de estos componentes, y su descomposición en unidades software de menor nivel que pueden ser codificadas, compiladas y probadas. Durante el diseño detallado hay que asegurarse de que todos los requisitos software asignados a los componentes software pasan a las unidades software de nivel inferior.

El diseño detallado también implica la definición detallada de las interfaces entre los componentes y entre las unidades software de menor nivel, así como el de las estructuras de datos (por ejemplo, bases de datos, archivos). También es necesario establecer los criterios de prueba de las unidades software. La idea fundamental que debe guiar a un buen diseñador de un componente software es que la documentación de diseño que genere ha de ser suficiente para la codificación y prueba de dicho componente.

Codificación y prueba del software

Aunque las actividades anteriores son importantes en el proceso de desarrollo de software, si duda, la actividad fundamental es la codificación y prueba del software, ya que se trata de la actividad cuyo objetivo es obtener el producto

software a desarrollar. Se trata de programar cada una de las unidades software identificadas en el diseño. También se deben crear los recursos de información que utilizará el software, como bases de datos o ficheros. Además de la construcción, es importante desarrollar también los procedimientos y datos de prueba de cada unidad software y de los recursos de información, para asegurar que cumplen los requisitos previstos para ellos.

La programación es una actividad fundamental para la obtención del producto final, sin embargo, no suele considerarse como una de las actividades que deba llevar a cabo lo que habitualmente se considera como Ingeniero del software, que estaría más vinculado a las actividades de análisis de requisitos y diseño. En las organizaciones, habitualmente existen tres categorías de desarrolladores asociadas a estas tres actividades: el análisis de requisitos lo llevan a cabo ingenieros con la categoría de Analista, el diseño lo realizan los Arquitectos software, y la codificación los Programadores³. Sin embargo, suele ser bastante habitual que en pequeñas y medianas empresas, estas tres funciones las desempeñe la misma persona con la categoría de Analista-Programador.

Integración del software

En los proyectos de desarrollo de software normalmente participan varios programadores, cada uno de los cuales se ha encargado de codificar varias unidades o componentes software. Por ello es muy importante considerar en el proceso software la necesidad de llevar a cabo una actividad específica dedicada a la integración de estas porciones del software que formarán parte del software final. La forma de llevar a cabo esta integración ha de ser definida, aunque las unidades de software hayan sido codificadas por un mismo programador. Se debe establecer, por tanto en cualquier proyecto la estrategia de integración que se seguirá, que debe ser consistente con la estrategia de liberación del software (por ejemplo, en sucesivas versiones con incremento de la funcionalidad en cada una de ellas).

La integración supone integrar las unidades software identificadas en el diseño detallado, en los componentes software establecidos en el diseño arquitectónico. Estos componentes pueden empaquetarse a su vez en agrupaciones o agregaciones de un mayor nivel, que deberían estar sometidas a un control de configuración (ver apartado 2.4.2). Hay que planificar, realizar y documentar las pruebas de integración que sean necesarias para asegurar el correcto funcionamiento de las interfaces entre los componentes que constituyen cada agrupación o paquete de software.

³En España, tradicionalmente estos tres perfiles se han denominado Analista Funcional, Analista Orgánico y Programador.

Pruebas de cualificación del software

Las pruebas de cualificación son aquellas que permiten determinar si el software integrado cumple cada uno de los requisitos inicialmente previstos para su aceptación por parte del usuario. Por tanto, hay que verificar cada entidad de software integrado⁴ usando los criterios de aceptación definidos, registrando y documentando los resultados de las pruebas.

Puesto que en la mayoría de los casos será necesario realizar modificaciones en el software debido a la no superación de alguna de las pruebas, o como consecuencia de operaciones de mantenimiento, es conveniente desarrollar una estrategia de regresión para la repetición de las pruebas de las agregaciones de software cuando se realizan cambios en los componentes que las integran.

Integración del sistema

Como ya se ha indicado anteriormente, cuando se desarrolla software, se hace para que éste forme parte de una entidad de nivel superior (sistema) junto con otros elementos (como hardware y operaciones manuales). Por ello, el proceso de desarrollo debe contemplar la definición de las acciones a llevar a cabo para la integración de los paquetes o items software con los elementos hardware, con las operaciones manuales del sistema y, si es necesario, con otros sistemas. Esta integración da lugar a agrupaciones a nivel de sistema, que han de cumplir los requisitos del sistema inicialmente establecidos. Hay que planificar, realizar y documentar las pruebas de integración a nivel de sistema que sean necesarias para asegurar el correcto funcionamiento de las interfaces entre los componentes que constituyen cada agrupación en el sistema.

Pruebas de cualificación del sistema

Al igual que ocurre con el software, también es necesario realizar pruebas del sistema del que forma parte, con el objetivo de verificar los componentes del sistema desarrollado usando los criterios de aceptación definidos al inicio de un proyecto. Se trata de demostrar el cumplimiento de los requisitos del sistema (funcionales, no funcionales, de operación, de mantenimiento, etc.). Como ocurría con el software, estas pruebas deben repetirse cuando se realizan cambios en alguno de los componentes software o hardware del sistema.

⁴Una entidad de software integrado es un artefacto software sometido a control de configuración. Se utilizan diferentes términos para referirse a este concepto, como "paquete", "agrupación", "agregación" o simplemente "item de configuración".

Instalación del software

El desarrollo de software suele hacerse normalmente en un entorno diferente (la organización desarrolladora) al entorno en el que el estará operativo (la organización usuaria). La ingeniería del software también se debe ocupar de las acciones a llevar a cabo para la instalación del software, una vez probado, en el entorno de operación. Para ello se debe elaborar un plan de instalación que establezca cómo ésta se llevará a cabo y que recursos se necesitarán para ello.

La instalación del software en algunos casos es una tarea sencilla, cuando se trata de implantar un único paquete de software en uno o varios equipos informáticos. Sin embargo, en los casos en los que el software deba reemplazar a otras aplicaciones ya existentes, debería establecerse una estrategia para realizar la instalación en paralelo con el sistema en funcionamiento; por ejemplo, dejando en funcionamiento ambos sistemas durante un periodo de tiempo antes de hacer efectiva la substitución de uno por otro. Por otra parte, cuando se trata de un software a instalar en diferentes ubicaciones físicas (por ejemplo, sucursales de una entidad bancaria), puede ser conveniente establecer una instalación progresiva, comenzando por una instalación piloto en una ubicación, para continuar con las ubicaciones de una localidad, después las de toda una región, etc.

Aceptación del software

En los proyectos de ingeniería del software no suele ser suficiente con la instalación del software para dar por concluido el desarrollo. Normalmente el cliente o usuario exige la realización de sus propias pruebas antes de aceptar el producto y liberar el pago de las cantidades que se adeuden al desarrollador. Durante las pruebas de aceptación el usuario puede decidir que el sistema no cumple algunos requisitos o encontrar algún error de ejecución, lo cual supone la realización de modificaciones en el software y la ejecución de las actividades que sean necesarias del proceso de desarrollo, incluida la repetición de todas las pruebas que se estimen conveniente para verificar que el software ya cumple todos los requisitos. Además, también hay que comprobar que, como consecuencia de los cambios, no se hayan dejado de cumplir otros, realizando lo que se conoce como "pruebas de regresión".

3.4 Operación del software

El objetivo del proceso de operación es establecer cómo debe operar el usuario (cliente) con el producto software en su entorno operativo, y cómo el suministrador va a proporcionar soporte operativo a los usuarios.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar y mitigar los riesgos operacionales (de uso) relacionados con la introducción del software y su operación.
- Operar con el software en su entorno de acuerdo con los procedimientos documentados.
- Proporcionar soporte operacional para resolver problemas de utilización y atender las preguntas y peticiones del usuario.
- Proporcionar la seguridad de que las capacidades del software (y del sistema) son adecuadas para resolver las necesidades del usuario.
- Identificar las necesidades de servicio de soporte del cliente.
- Valorar la satisfacción del cliente con los servicios de soporte que se le están proporcionando y con el propio producto.
- Ofrecer los servicios que el cliente necesita.

3.5 Mantenimiento del software

El objetivo del proceso de mantenimiento es modificar un producto software operativo después de su entrega, para corregir fallos, mejorar su rendimiento u otros atributos, o adaptarlo a los cambios del entorno. Este proceso también incluye la migración y retirada final del producto; de hecho, puede decirse que la realización de este proceso realmente sólo termina con la retirada del producto software.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Definir el impacto de las modificaciones sobre la organización, operaciones e interfaces del sistema en operación.
- Identificar y actualizar la información correspondiente del ciclo de vida.
- Desarrollar los componentes del sistema a modificar con su correspondiente documentación y pruebas que demuestren que los requisitos del sistema siguen cumpliéndose.
- Migrar las actualizaciones del software y del sistema al entorno de usuario.
- Asegurar que la incorporación de nuevos sistemas o versiones no afecta de forma adversa a las operaciones habituales.
- Mantener la capacidad de recuperar las versiones previas.

El proceso de mantenimiento está estrechamente ligado al de desarrollo, ya que el mantenimiento normalmente precisará la ejecución de las actividades establecidas en ese proceso, si bien sobre el producto operativo y no sobre un nuevo producto. En ese caso, las actividades serían las de desarrollo, pero la figura del "desarrollador" sería sustituida por la del "mantenedor".

4 Procesos de apoyo

Aunque en la sección anterior se han descrito los procesos fundamentales que la Ingeniería del Software establece que deben definirse y ejecutarse en los proyectos de desarrollo, existen otros procesos que pueden denominarse "de apoyo", o "de soporte", sin los cuales sería muy difícil que aquellos llegaran a buen término. Son procesos relacionados, entre otros, con la elaboración y gestión de la documentación asociada a los proyectos; con la implantación de un sistema de aseguramiento de la calidad, tanto de los procesos como de los productos; o con el control de las versiones de los cambios que sufran los diferentes elementos que forman parte del software a lo largo de su "vida". En los siguientes apartados se tratarán brevemente estos procesos auxiliares.

4.1 Documentación del software

El objetivo del proceso de documentación es registrar la información producida por las actividades del resto de procesos de la Ingeniería del Software. Esto incluye la planificación, diseño, desarrollo, producción, edición, distribución y mantenimiento de los documentos que necesitan todos los implicados en los proyectos, como gestores, ingenieros y usuarios.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar todos los documentos a producir por cada proceso del ciclo de vida.
- Especificar el contenido y propósito de todos los documentos y planificar su producción.
- Identificar los estándares que se deben aplicar en el desarrollo de los documentos.
- Desarrollar y publicar todos los documentos de acuerdo con los estándares identificados y con la planificación establecida.
- Mantener todos los documentos de acuerdo con los criterios especificados.

4.2 Gestión de la configuración

El objetivo del proceso de gestión de la configuración es establecer y mantener la integridad de todos los productos creados en un proceso o proyecto, y hacer que estén disponibles para las partes implicadas. Esto supone la necesidad de aplicar procedimientos administrativos y técnicos a través del ciclo de vida del software para gestionar los cambios de versión de los diferentes elementos del software.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar, definir y controlar la línea de base⁵ de todos los elementos de un proyecto.
- Controlar las modificaciones de los elementos.
- Registrar e informar sobre el estado de los elementos y las peticiones de modificación.
- Asegurar la completitud, consistencia y corrección de los elementos.
- Controlar el almacenamiento, manipulación, liberación (*release*) y entrega de los elementos.

4.3 Aseguramiento de la calidad

El objetivo del proceso de aseguramiento de la calidad es proporcionar la seguridad de que los productos software y los procesos del ciclo de vida del software cumplen con sus requisitos y se ajustan a la planificación establecida; para lo cual utiliza principalmente los resultados de otros procesos de apoyo, como los de verificación, validación, revisiones conjuntas, auditoría y resolución de problemas. Para ser imparcial, el aseguramiento de la calidad necesita tener libertad organizacional y autoridad de las personas directamente responsables del desarrollo de los productos software o de la ejecución de los procesos en un proyecto. El aseguramiento de la calidad puede ser interno, llevado a cabo por personal vinculado a la organización de desarrollo, o externo, en este caso llevado a cabo por un tercero independiente del desarrollo.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar, planificar y temporizar las actividades de aseguramiento de la calidad para un proceso o producto.

⁵Línea de base (*Baseline*): "Especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, y que en adelante sirve como base para un desarrollo posterior y que puede cambiarse sólo a través de procedimientos formales de control de cambios" (IEEE 1990)

- Identificar estándares de calidad, metodologías, procedimientos y herramientas para realizar las actividades de aseguramiento de la calidad y adaptarlo a un proyecto.
- Identificar recursos y responsabilidades para realizar las actividades de aseguramiento de la calidad.
- Establecer y garantizar la independencia de los responsables de realizar las actividades de aseguramiento de la calidad.
- Realizar las actividades de aseguramiento de la calidad de acuerdo con planes relevantes, procedimientos y temporizaciones.
- Aplicar sistemas de gestión de calidad al proyecto.

4.4 Verificación del software

El objetivo del proceso de verificación es determinar si los requisitos de un sistema o software son completos y correctos, y si los productos software obtenidos en cada actividad del ciclo de vida cumplen los requisitos o condiciones impuestos sobre ellos en las actividades previas ⁶. Para un coste y rendimiento más eficiente, las actividades de verificación deberían ser integradas lo antes posible con los procesos que las emplean (por ejemplo, los procesos de suministro, desarrollo, operación o mantenimiento). Este proceso puede ser ejecutado con varios grados de independencia, pudiendo ser llevado a cabo por personal vinculado al suministrador, al cliente o a una tercera entidad independiente.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar los criterios para la verificación de todos los productos requeridos.
- Realizar las actividades de verificación de requisitos.
- Encontrar y eliminar defectos de los productos producidos por un proyecto.

4.5 Validación del software

El objetivo del proceso de validación es determinar si los requisitos de un sistema o software, y el propio sistema o software final, cumplen con su uso previsto ⁷. Como en el caso de la verificación, este proceso puede ser ejecutado con varios

⁶Se trata de preguntarse: "¿Se está desarrollando el producto correctamente?"

⁷Se trata de preguntarse: "¿Se está desarrollando el producto correcto?"

grados de independencia, pudiendo ser llevado a cabo por personal vinculado al suministrador, al cliente o a una tercera entidad independiente.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar los criterios para la validación de todos los productos requeridos.
- Realizar las actividades de validación requeridas.
- Comprobar que los productos desarrollados son apropiados para su uso previsto.

4.6 Revisiones conjuntas

El objetivo del proceso de revisiones conjuntas es evaluar el estado y los productos software de una actividad del ciclo de vida, tanto a nivel de gestión como a nivel técnico.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Evaluar el estado y productos de una actividad de un proceso a través de actividades de revisión conjunta entre las partes del contrato.
- Establecer mecanismos para asegurar que las decisiones alcanzadas sean registradas para llevarlas a cabo.

4.7 Auditoría del software

El objetivo del proceso de auditoría es determinar el grado en que se cumplen los requisitos, planes y contrato, a petición de una de las partes contratantes.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Determinar el cumplimiento de requisitos, planes y contrato, según sea apropiado.
- Organizar la forma de llevar a cabo las auditorías de los productos (código fuente, informes, documentos, estimaciones de costes, etc.) o procesos por una entidad independiente y cualificada.
- Llevar a cabo las auditorías para determinar acciones correctivas.

4.8 Resolución de problemas

El objetivo del proceso de resolución de problemas es analizar y resolver los problemas que surgen durante la ejecución de los procesos del ciclo de vida del software, especialmente los de desarrollo, operación y mantenimiento.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Proporcionar un medio oportuno, responsable y documentado de asegurar que todos los problemas descubiertos se analizan y resuelven.
- Proporcionar un mecanismo para reconocer y actuar sobre las tendencias en los problemas identificados.

5 Procesos organizacionales

La última categoría de procesos que se pueden identificar en la Ingeniería del software está constituida por aquellos que han de definir y ejecutar las organizaciones de desarrollo de software para establecer una estructura organizativa que permita la implantación del resto de procesos y del personal necesario para llevarlos a cabo, y facilite la mejora continua de su proceso software. Son procesos relacionados con actividades de gestión, de establecimiento de la infraestructura para el resto de procesos, de mejora de los procesos y de preparación del personal de la organización.

5.1 Gestión de proyectos software

Además de actividades específicas de ingeniería, como las consideradas en el resto de procesos, son necesarias actividades de planificación y seguimiento de los proyectos software que garanticen que finalmente el software será desarrollado en un plazo y con un coste determinados. Estas actividades forman parte de lo que se puede denominar proceso de gestión, cuyo objetivo es realizar las actividades genéricas que ha de llevar a cabo cualquiera de las partes implicadas en un proyecto y que tenga que gestionar o dirigir los procesos de los que sea responsable. Además de los procesos, deben gestionarse también los productos generados a lo largo del ciclo de vida del software.

Entre los objetivos específicos de la gestión de un proyecto software se encontrarían los siguientes:

- Definir el alcance del trabajo del proyecto.
- Identificar, medir, estimar, planificar, controlar y medir las tareas y recursos necesarios para completar el trabajo.

- Identificar y gestionar las interfaces entre elementos en el proyecto y con otros proyectos y unidades organizacionales.
- Realizar las acciones correctivas cuando no se alcanzan las metas del proyecto.
- Establecer objetivos de calidad, basados en los requisitos de calidad del cliente, para varios puntos de control en el ciclo de vida del proyecto.
- Establecer objetivos de rendimiento del producto (memoria, procesamiento, comunicaciones), para varios puntos de control en el ciclo de vida del proyecto.
- Definir y utilizar métricas para cuantificar los resultados de las actividades del proyecto, en puntos de control del ciclo de vida del proyecto, para valorar si se han alcanzado los objetivos de rendimiento.
- Establecer criterios, métricas y procedimientos para identificar prácticas de la ingeniería del software, e integrar prácticas mejoradas en los procesos adecuados del ciclo de vida del software.
- Realizar las actividades de calidad y confirmar su realización.
- Llevar a cabo las acciones correctivas necesarias cuando no se alcancen los objetivos técnicos, de calidad, de rendimiento del producto.
- Determinar el alcance de la gestión de riesgos que se llevará a cabo en el proyecto.
- Identificar riesgos del proyecto.
- Analizar riesgos y determinar la prioridad para aplicar recursos para mitigar esos riesgos.
- Definir, implementar y valorar las estrategias de mitigación de riesgos mas adecuadas.
- Definir, aplicar y valorar las métricas de riesgo para medir el cambio en el estado de riesgo y el progreso de las actividades de mitigación.
- Establecer un entorno que soporte la interacción efectiva entre individuos y grupos.
- Llevar a cabo las acciones correctivas necesarias cuando no se alcance el progreso esperado.

5.2 Infraestructura del software

El objetivo del proceso de infraestructura es establecer y mantener la infraestructura necesaria para realizar cualquier otro proceso. La infraestructura puede incluir hardware, software, herramientas, técnicas, estándares, y facilidades para el desarrollo, operación o mantenimiento.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Establecer y mantener un entorno de ingeniería del software, consistente con, y soporte de, un conjunto de procesos estándar, métodos y técnicas organizacionales.
- Adaptar el entorno de ingeniería del software a las necesidades del proyecto y del equipo del proyecto.
- Desarrollar un entorno de ingeniería del software que de soporte a los miembros del equipo del proyecto independientemente de la localización donde se realicen las actividades.
- Implementar una estrategia definida para la reutilización.

5.3 Mejora del software

El objetivo del proceso de mejora es establecer, valorar, medir, controlar y mejorar los procesos del ciclo de vida.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Establecer un conjunto bien definido de procesos estándar, junto con una descripción de la aplicabilidad de cada proceso.
- Identificar las tareas, actividades y productos para cada proceso estándar.
- Adaptar los procesos a las necesidades de un proyecto.
- Establecer y mantener información y datos relacionados con la realización de los procesos.
- Entender los puntos fuertes y débiles de los procesos.
- Hacer cambios en la definición de los procesos de forma controlada.
- Implementar las actividades, planificadas y monitorizadas, de mejora de los procesos de forma coordinada con el resto de la organización.

5.4 Formación

El objetivo del proceso de formación es proporcionar y mantener un personal formado. Es importante tener en cuenta que la mayoría de las actividades previstas por los procesos principales dependen en gran medida del conocimiento y habilidad del personal implicado.

Entre los objetivos específicos de este proceso se encontrarían los siguientes:

- Identificar los papeles (perfiles) y habilidades requeridas para la operaciones de la organización y de un proyecto.
- Establecer procedimientos formales para reclutar y seleccionar personal y su asignación a la organización.
- Diseñar y conducir la formación para asegurar que todas los individuos tienen las habilidades requeridas para su trabajo.
- Identificar y reclutar o formar, según sea apropiado, individuos con las habilidades y competencias requeridas para llevar a cabo su trabajo
- Establecer una fuerza de trabajo con las habilidades para compartir información y coordinar sus actividades eficientemente.
- Definir los criterios objetivos de medición de los resultados de la formación para su mejora.

6 Estándares sobre procesos en la Ingeniería del Software

6.1 ISO/IEC/IEEE 12207

Esta norma de ISO, denominada "*Software Life-Cycle Processes*", establece un "marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso" (ISO 1995a). Aunque no se trata del primer estándar sobre este tema, ya que en 1991 se publicó un estándar similar por IEEE (IEEE 1074), que se describe en el siguiente apartado, sí ha sido la que más repercusión ha tenido en el ámbito de la Ingeniería del Software a nivel mundial, siendo en la actualidad la referencia obligada en este contexto, como lo demuestra el hecho de que IEEE finalmente decidió asumirla en 1998, en detrimento de su norma IEEE

6. ESTÁNDARES SOBRE PROCESOS EN LA INGENIERÍA DEL SOFTWARE 25

1074, con el nombre de "IEEE/EIA 12207.0 Industry Implementation of International Standard ISO/IEC 12207:1995" (IEEE 1998a). Esta organización también ha publicado otras dos guías para la aplicación de dicho estándar (IEEE 1998b) (IEEE 1998c).

Como ocurre con IEEE 1074, este estándar no prescribe una metodología concreta con un ciclo de vida de desarrollo determinado, sino que establece un marco de referencia que puede ser utilizado por las organizaciones para situar en él su propia metodología. ISO/IEC 12207 recomienda considerar los diecisiete procesos descritos en los apartados 1.3 a 1.5 de este documento: cinco de los cuales son esenciales ("principales") para la realización del software. El resto son procesos "de apoyo" (un total de ocho) y "organizacionales" (cuatro) que facilitan la realización de los principales. Su descripción detallada se llevó a cabo en los apartados citados. El estándar 12207 y las guías publicadas por IEEE, incluyen recomendaciones sobre como aplicar esta norma en las organizaciones y como adaptarla a diferentes modelos de ciclo de vida del software. En la figura 1.5, incluida en el estándar original, se esquematiza el procedimiento de aplicación de esta norma a un proyecto de desarrollo en una organización (ISO 1995a). Como puede observarse en la figura, la organización debe considerar qué restricciones son relevantes y aplicables al proyecto, como los requisitos legales, de seguridad o de carácter técnico (por ejemplo, los lenguajes de programación a utilizar); y las restricciones de tiempo, económicas, de cumplimiento de otros estándares (por ejemplo, de calidad, como ISO 9001, en la figura). También se debe determinar que metodología, modelo de ciclo de vida, y entorno de desarrollo son los más convenientes para el proyecto. La adaptación del estándar se debería plasmar en una serie de documentos, como un contrato de desarrollo con el cliente, un plan de calidad, un plan del proyecto, una asignación de responsabilidades (en la figura, en forma de matriz), antes de proceder al inicio del proyecto.

6.2 IEEE 1074

Esta norma internacional, publicada en 1991 (y revisada en 1997) por la *IEEE Computer Society* y asumida por ANSI (*American National Standards Institute*), se denomina "*Standard for Developing Software Life Cycle Processes*" y "define los procesos que deberían considerarse en el desarrollo y mantenimiento de software, ofreciendo un conjunto de actividades que constituyen tales procesos, junto con su información de entrada y salida asociada" (IEEE 1997).

Al igual que ISO 12207 no establece ningún modelo concreto de ciclo de vida o método de desarrollo, ni describe cómo realizar ninguna actividad. Para no perder generalidad, se limita a presentar los procesos y actividades que deberían contemplarse y lo que debería hacerse en cada caso.

IEEE 1074 recomienda considerar un total de dieciséis procesos a lo largo

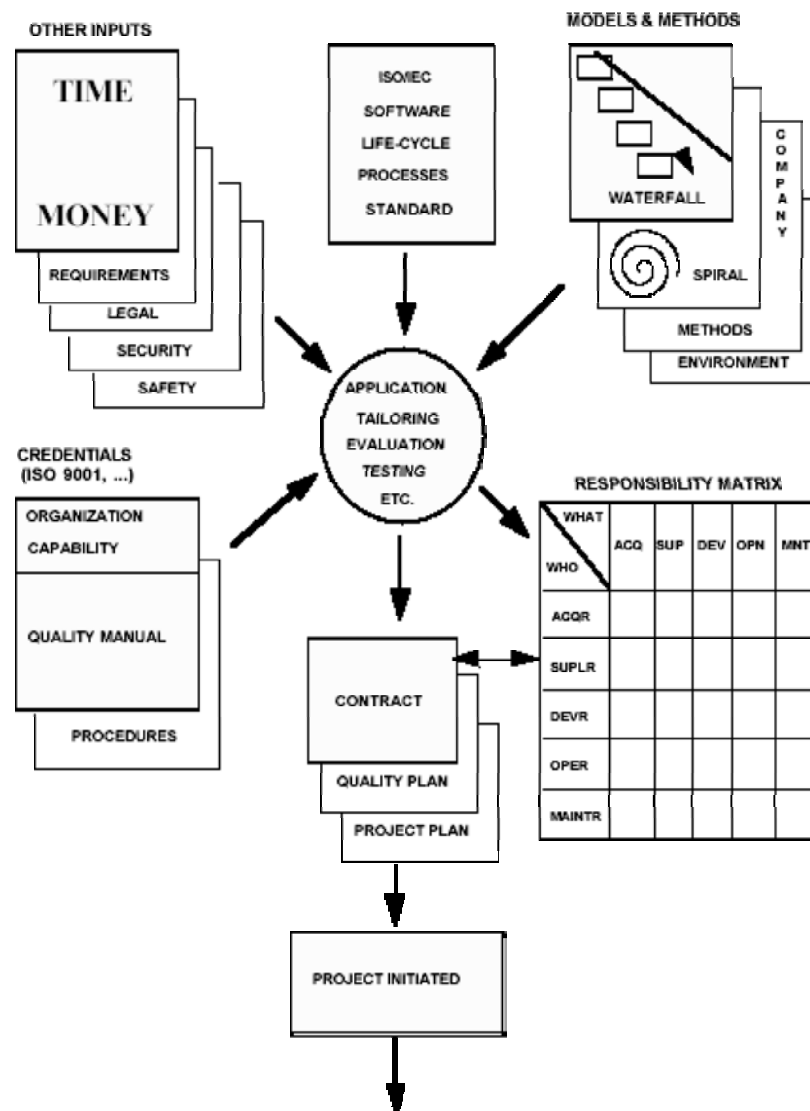


Figura 1.5: Ejemplo de aplicación del estándar 12207

de un proyecto de desarrollo de software. Un primer proceso para establecer previamente la metodología que se va a emplear ("proceso de modelo de ciclo de vida del software"), un conjunto de otros tres procesos para iniciar y controlar el proyecto ("procesos de gestión del proyecto"), dos procesos previos al comienzo del desarrollo ("procesos de pre-desarrollo"), tres propiamente de desarrollo ("procesos de desarrollo") y otros tres posteriores al mismo ("procesos de post-desarrollo"). Finalmente, se consideran también cuatro denominados "procesos integrales", necesarios para completar con éxito todas las actividades

6. ESTÁNDARES SOBRE PROCESOS EN LA INGENIERÍA DEL SOFTWARE²⁷

del proyecto garantizando un adecuado nivel de calidad.

Además de la descripción detallada del objetivo, información de entrada e información de salida de todas las actividades incluidas en los procesos anteriores, para facilitar su aplicación, también incluye un anexo que muestra su adaptación a diferentes modelos concretos de ciclo de vida del software.

Aunque la norma no establece la secuencia de ejecución de estos procesos, pues dependerá en gran medida de la metodología que se adopte en cada caso, sí define los flujos de información entre ellos, es decir qué entradas y salidas tiene asociadas cada proceso y su origen o destino. Todos estos procesos y sus correspondientes actividades son los siguientes:

Proceso de modelo de ciclo de vida del software

Como esta norma no especifica una metodología ni, por tanto, un modelo de ciclo de vida del software concreto, se ha previsto este proceso para proceder al establecimiento de la que se aplicará en cada proyecto de desarrollo. La lista de actividades asociadas a este proceso es la siguiente: Identificar modelos de ciclo de vida del software candidatos, Seleccionar el modelo para el proyecto.

Procesos de gestión de proyecto

Se establecen tres procesos relacionados con la gestión de los proyectos: **Proceso de iniciación del proyecto**, que define las actividades que crean el marco para el proyecto, preparando el plan para la gestión del proyecto e identificando los estándares, metodologías y herramientas necesarias para gestionar y ejecutar el proyecto; **Proceso de control de proyecto**, con actividades iterativas relacionadas con el seguimiento, documentación y gestión de costes, tiempos, problemas y rendimiento de un proyecto a través de su ciclo de vida; y **Proceso de gestión de calidad del software**, que incluye la planificación y administración de las acciones necesarias (definición de métricas de medición de la calidad, gestión de la calidad del software, identificación de las necesidades de mejora de la calidad) para poder ofrecer un adecuado nivel de confianza en el software.

Procesos de pre-desarrollo

El estándar considera la necesidad de definir dos procesos previos al desarrollo del software: **Proceso de exploración**, destinado al análisis previo de las necesidades a las que dará respuesta el software a desarrollar; y **Proceso de asignación**, a considerar sólo en los proyectos que incluyan una parte hardware

además del software, destinado a analizar los requisitos generales del proyecto y separar los correspondientes al software de los del hardware.

Procesos de desarrollo

Según la norma, existen tres procesos estrictamente vinculados al desarrollo del software: **Proceso de requisitos**, cuyo objetivo es la especificación de los requisitos del software, de las interfaces de usuario y del hardware; **Proceso de diseño**, para determinar la estructura del software, pasando de la descripción del "qué hacer" de la especificación de requisitos al "cómo hacerlo" de las especificaciones de diseño, que incluyen el diseño de la arquitectura del software, el análisis de los flujos de información, el diseño de bases de datos, el diseño de interfaces, el diseño detallado y la selección o desarrollo de algoritmos; y **Proceso de implementación**, con las actividades para la transformación del diseño detallado de un producto software a su representación mediante un lenguaje de programación, considerando tanto la codificación e integración de componentes software como la realización de las pruebas necesarias.

Procesos de post-desarrollo

Como complemento a los procesos anteriores, el estándar 1074 establece otros cuatro procesos a considerar una vez finalizado el desarrollo de software: **Proceso de instalación**, dedicado a la implantación del software en el entorno donde funcionará habitualmente; **Proceso de operación y soporte**, que considera el funcionamiento del sistema en el entorno del usuario y el soporte de asistencia técnica; **Proceso de mantenimiento**, para definir tanto el mantenimiento correctivo, perfectivo como adaptativo; y **Proceso de retirada**, con las actividades implicadas en la retirada del software y, en su caso, substitución por otro o por una nueva versión.

Procesos integrales

Para dar soporte al resto de procesos del ciclo de vida, son necesario otros cuatro procesos que la norma denomina "integrales": **Proceso de verificación y validación**, que incluye las tareas relacionadas con revisiones, auditorías y pruebas realizadas sobre los diferentes procesos y productos generados a lo largo del ciclo de vida del software; **Proceso de gestión de configuración del software**, con las actividades para controlar los cambios producidos en los elementos software durante el ciclo de vida; **Proceso de desarrollo de la documentación**, que considera las actividades de planificación, diseño, implementación, edición, producción, distribución y mantenimiento de aquellos documentos que necesitan

desarrolladores y usuarios; y **Proceso de formación**, dedicado a las actividades para la formación del personal usuario y de explotación del software.

7 Madurez del proceso software

Estándares como las normas ISO 12207 o IEEE 1074 pueden ayudar a las organizaciones a mejorar su proceso software y, así, estar en condiciones de obtener diferentes certificaciones de calidad, como ISO 9001 (ISO 2000), o de madurez, según, por ejemplo, la escala CMMI (SEI 2002a) o SPICE (ISO 2004a). Esto es así porque estas normas ofrecen un marco de referencia en el que poder insertar todos los elementos necesarios para obtener el reconocimiento de calidad o madurez, como es la aplicación de metodologías de desarrollo, gestión de calidad, gestión de configuración, formación, etc.

Por ello, además de los estándares sobre los procesos (o subprocesos) que han de formar parte del proceso software, es conveniente conocer la existencia de otros estándares relacionados con la calidad y la madurez del proceso, que también hacen referencia a los subprocesos que deberían definir las organizaciones para alcanzar un determinado nivel de capacidad y madurez. Entre los estándares más importantes en este ámbito se encuentran CMMI, ISO 15504 (SPICE) e ISO 9001. Los dos primeros ofrecen un enfoque distinto de calidad respecto ISO 9001, en cuanto a que mientras esta norma establece un objetivo a conseguir para obtener la correspondiente certificación de calidad sin indicar la mejor manera de conseguirlo, CMMI y SPICE determinan las acciones a seguir que permitan acceder a una organización al siguiente nivel de madurez para mejorar la calidad del proceso y, por tanto, del software.

7.1 CMMI (Capability Maturity Model Integration)

El Modelo de Capacidad de Madurez para el Software (CMM: Capability Maturity Model for Software) empezó a ser desarrollado en 1986 por el SEI (Software Engineering Institute) de la Universidad de Carnegie Mellon en EE.UU. como respuesta a un encargo del Departamento de Defensa americano (DoD) para desarrollar una guía que le permitiera seleccionar a los contratistas de software más capacitados y evitar los problemas que se estaban encontrando con proyectos que no terminaban en el plazo previsto y cuyo coste duplicaba el presupuesto establecido inicialmente.

Aunque aparecieron borradores desde 1987, es en 1991 cuando se publica la primera versión del modelo CMM, el cual establecía una escala de cinco niveles de madurez en los que pueden encontrarse las organizaciones de desarrollo según su forma de trabajo. Cada nivel determina la calidad de los productos obtenidos

por las organizaciones situadas en dicho nivel ya que, como afirma uno de sus autores, el principio que inspira al modelo CMM es que "la calidad de un producto software es determinada fundamentalmente por la calidad del proceso de desarrollo y mantenimiento utilizado para construirlo" (Paulk, Curtis, Chrissis & Weber 1987).

Después del éxito de CMM para el caso de la Ingeniería del Software (también conocido como SW-CMM), el SEI desarrolló modelos similares para otras disciplinas, como, entre otras, la Ingeniería de Sistemas (SE-CMM: System Engineering CMM), el desarrollo integrado de productos (IPD-CMM: Integrated Product Development CMM), o la adquisición de software (SA-CMM: Software Acquisition CMM); que culminaron en 2002 con la unificación de todos ellos en un modelo integrado conocido como CMMI (Capability Maturity Model Integration) (SEI 2002a), siendo el modelo CMMI-SW el correspondiente al proceso software (SEI 2002b).

El modelo CMMI está basado en los principios de "Gestión de la Calidad Total" (TQM: Total Quality Management) cuyo objetivo es implicar a todos los servicios, personal, procesos y actividades de una organización en un proyecto común de calidad. El resultado es una reacción en cadena, como la enunciada por (Deming 1986), en la que a medida que una organización va madurando en este sentido, el proceso de desarrollo de software estará mejor definido y mas consistentemente implantado en el seno de la organización, lo que al final redunda en un proceso maduro que, debido a la disciplina que implica su aplicación, posibilita una mejora constante de la productividad y, en definitiva, de la calidad.

Estos principios de la TQM han sido adaptados en el CMMI, ofreciendo una guía para incrementar el control sobre el proceso de desarrollo y mantenimiento de software mediante la evaluación del nivel de madurez del proceso y la selección de estrategias de mejora del mismo. Este modelo, que se inspira en el modelo de madurez para la gestión de la calidad de (Crosby 1979), establece una escala de valoración dividida en cinco niveles de madurez. También, excepto en el caso del nivel básico, cada nivel se descompone en aquellas áreas críticas de la organización en las que habría que concentrar el esfuerzo para mejorar el proceso de software. Estos niveles son los siguientes (figura 1.6):

- (1) Inicial (*Initial*): Las organizaciones con este nivel de madurez se caracterizan porque su proceso de desarrollo software normalmente se reduce a la codificación y pruebas. No es una actividad de ingeniería, resultando impredecibles los costes y calendario de los proyectos y la calidad del software, dependiendo por completo de la capacidad individual del personal implicado.

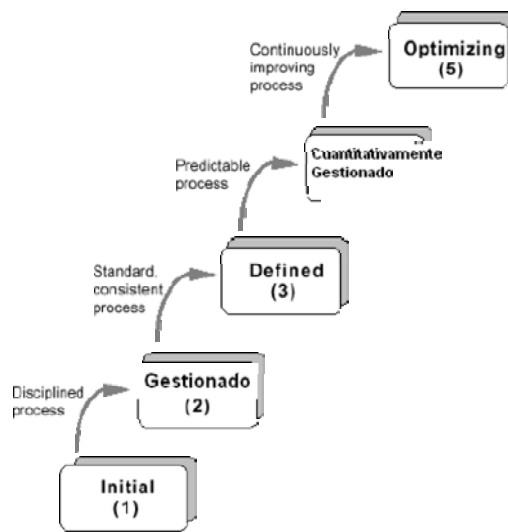


Figura 1.6: Niveles de madurez del proceso software según CMMI

- (2) Gestionado⁸ (*Managed*): En este nivel, el proceso se basa en la repetición de las prácticas de gestión experimentadas en proyectos similares anteriores. Existen mecanismos básicos de gestión para controlar el coste, calendario y la funcionalidad del software. El control del calendario es razonable pero los costes y la calidad pueden estar sujetos a grandes fluctuaciones.
- (3) Definido (*Defined*): Las actividades de gestión y de ingeniería (diseño, programación, etc.) están documentadas y normalizadas según una metodología de desarrollo definida en la organización. Los costes y los plazos son fiables y se garantiza la calidad del software, cuando menos, a nivel cualitativo.
- (4) Cuantitativamente gestionado⁹ (*Quantitatively managed*): Se establecen objetivos de calidad y se realizan mediciones del proceso de desarrollo y de los productos obtenidos en todos los proyectos que se registran para evaluar los procesos en el futuro. Se garantiza la calidad del software a nivel cuantitativo.
- (5) Optimizado (*Optimizing*): El objetivo principal de la organización es la mejora continuada del proceso de desarrollo, incorporando aquellas nuevas tecnologías que lo optimicen. La calidad es total, así como la productividad, con un riesgo nulo.

⁸Denominado "Repetible" en el modelo CMM original

⁹Denominado "Gestionado" en el modelo CMM original

La decisión del gobierno de EE.UU. de exigir a partir de 1998 el nivel 3 a sus proveedores de software supuso una auténtica revolución en las organizaciones de desarrollo y desencadenó un enorme interés en todo el mundo respecto a la mejora del proceso software.

En relación con los procesos tratados anteriormente en este documento, hay que indicar que el estándar CMMI establece lo que denomina "áreas de proceso" (*Process Areas*), cada una de las cuales representa un conjunto de actividades interrelacionadas, que al realizarse conjuntamente consiguen satisfacer unos objetivos de mejora de la capacidad del proceso software en ese área. Estas áreas de proceso pueden asimilarse al concepto de subproceso manejado en los apartados anteriores de este documento. Las áreas de proceso están organizadas por niveles de madurez, de tal forma que para alcanzar un determinado nivel, una organización debe implementar las áreas de proceso establecidas para ese nivel y así satisfacer los objetivos establecidos para las mismas. En la siguiente tabla se muestran las 22 áreas de proceso establecidas por CMMI para cada nivel de capacidad de madurez, así como la categoría o tipo de proceso a la que pertenece cada área, de entre las cuatro establecidas por CMMI: Ingeniería, Gestión de proyecto, Gestión de proceso, y Soporte.

Nivel CMMI	Área de proceso	Categoría
2	Requirements Management	Ingeniería
	Project Planning	Gestión proyecto
	Project Monitoring and Control	Gestión proyecto
	Supplier Agreement Management	Gestión proyecto
	Measurement and Analysis	Soporte
	Process and Product Quality Assurance	Soporte
	Configuration Management	Soporte
3	Requirements Development	Ingeniería
	Technical Solutions	Ingeniería
	Product Integration	Ingeniería
	Verification	Ingeniería
	Validation	Ingeniería
	Organizational Process Focus	Gestión proceso
	Organizational Process Definition	Gestión proceso
	Organizational Training	Gestión proceso
	Integrated Project Management	Gestión proyecto
	Risk Management	Gestión proyecto
	Decision Analysis and Resolution	Soporte
	Organizational Process Performance	Gestión proceso
	Quantitative Project Management	Gestión proyecto
5	Organizational Innovation and Deployment	Gestión proceso
	Causal Analysis and Resolution	Soporte

Tabla. Áreas de proceso según CMMI

7.2 ISO 14504 (SPICE)

El estándar ISO 14504, también conocido como SPICE ("*Software Process Improvement and Capability dEtermination*") es una norma similar a CMMI pero con origen en el ámbito europeo, concretamente en el *European Software Institute* (ESI), creado en 1993 a iniciativa de la Comisión Europea para promover la mejora de la competitividad de las empresas de software europeas a través de su proceso de producción (ISO 2004a).

ISO 14504 se basa en el estándar ISO 12207 y en CMMI. Del primero recoge la necesidad de descomponer el proceso software en subprocesos, asumiendo y ampliando el número de procesos establecidos por ISO 12207, y aumentando también las categorías en las que se pueden enmarcar estos procesos. De CMMI asume la idea de los niveles de madurez, aunque en el caso ISO 14504 se convierten en niveles de capacidad de los procesos, se establece un nuevo nivel previo, y cambian los nombres de los cinco restante respecto a CMMI (figura 1.7). Es

decir, mientras CMMI describe la madurez organizacional de una empresa, ISO 15504 describe la capacidad de su proceso software¹⁰. Los niveles de capacidad identifican si los procesos son incompletos, realizados, gestionados, establecidos, predecibles u optimizados, como se indica a continuación:

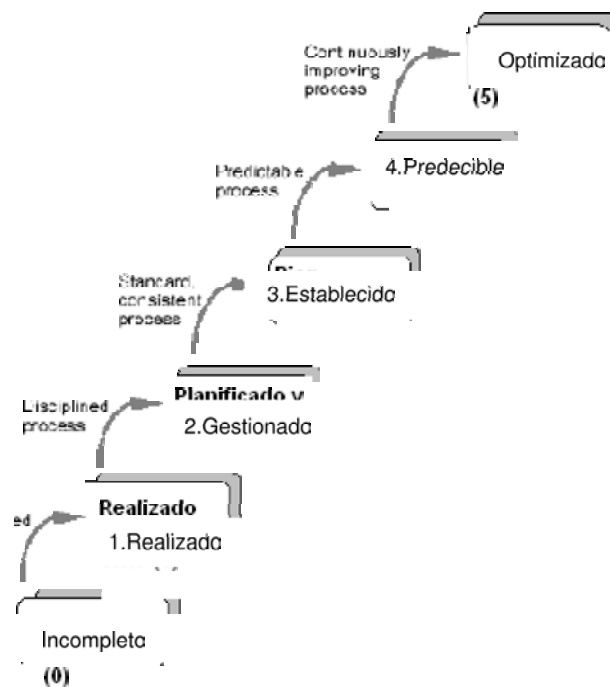


Figura 1.7: Niveles de capacidad del proceso software según ISO 15504

- (0) Incompleto (*Incomplete*): Un proceso es incompleto si no está implementado o falla y no puede alcanzar su propósito. Es, en definitiva, caótico.
- (1) Realizado (*Performed*): El proceso se realiza de manera intuitiva y alcanza su propósito. Es un proceso porque se dispone de productos de entrada y se obtienen productos de trabajo de salida.
- (2) Gestionado (*Managed*): El proceso está gestionado y se han establecido los productos a obtener, que son controlados y mantenidos. Se identifican las responsabilidades de los participantes.

¹⁰CMMI certifica un nivel final para una organización, ISO 15504 certifica un nivel final para cada proceso en una organización.

- (3) Establecido (*Established*): Se utiliza un proceso definido adaptado a un uso específico. Los recursos son gestionados.
- (4) Predecible (*Predictable*): Se manejan métricas, que hacen controlable la realización del proceso y sus resultados.
- (5) Optimizado (*Optimizing*): Se realiza una mejora continua del proceso, para adaptarse a los objetivos de negocio actuales y futuros. Se innova y optimiza constantemente el proceso.

En cuanto a los procesos establecidos por esta norma, están agrupados en 5 categorías diferentes. En la siguiente tabla se puede observar la equivalencia de categorías respecto a las áreas de proceso de CMMI. Los procesos enmarcados en cada categoría son los siguientes:

CMMI Process Area Categories	ISO 15504 Process Categories
Process Management	Organization
Project Management	Management
Engineering	Engineering
Support	Support
	Customer-Supplier

Figura 1.8: Comparación de categorías de procesos en CMMI e ISO 15504

- Cliente-Proveedor: Esta categoría incluye diez procesos que afectan directamente a las relaciones entre el cliente y el proveedor o suministrador. Son los procesos siguientes: preparación de la adquisición, selección del proveedor, control del proveedor, aceptación por el cliente, preparación de la entrega, entrega del producto, negociación de los requisitos, uso operacional del software, soporte al cliente, mantenimiento del contrato.
- Ingeniería: Son procesos orientados a la producción del software: análisis y diseño de los requisitos del sistema, análisis de los requisitos del software, diseño del software, construcción del software, prueba de integración del software, prueba del software, integración del sistema, mantenimiento del software y del sistema.
- Soporte: Categoría compuesta por procesos que dan soporte al resto de procesos: documentación, gestión de la configuración, aseguramiento de la calidad, verificación, validación, revisión conjunta, auditoría, resolución de problemas.

- **Gestión:** Son procesos relacionados con las actividades de administración de los recursos necesarios para producir el software. Son los siguientes: gestión, gestión del proyecto, gestión de calidad, gestión de riesgos, gestión de la información.
- **Organización:** Esta categoría engloba procesos que proporcionan a la organización la infraestructura en la que se integran el resto de procesos: preparación de la organización, establecimiento del proceso, evaluación del proceso, mejora del proceso, gestión de recursos humanos, infraestructura, medición, reutilización.

7.3 ISO 9001/ISO 9000-3

Estos estándares forman parte de una familia o serie de normas denominadas ISO 9000, que empezaron a publicarse en 1987 con el objetivo de definir los requisitos de los sistemas de calidad que pueden utilizarse para el aseguramiento externo de la calidad, es decir, para que un suministrador demuestre su capacidad y para la evaluación de la capacidad de un suministrador por partes externas.

ISO 9001 es la norma principal de la serie, que define con detalle cómo debe ser un sistema de calidad mediante un modelo para el aseguramiento de la calidad en el diseño, desarrollo, producción, instalación y mantenimiento de productos y servicios (ISO 2000). Los certificados que se expiden a las organizaciones llevan precisamente el nombre de esta norma. Para el caso de la industria del software, se ha elaborado la guía ISO 9000-3 para la adaptación de ISO 9001 a este campo (ISO 2004b); de tal forma que siguiendo las recomendaciones de la guía 9000-3, las organizaciones de desarrollo de software estarían en disposición de obtener la preciada certificación "ISO 9001" que ayude a superar la natural desconfianza de sus potenciales clientes.

El objetivo conjunto de estas normas es establecer un modelo genérico de sistema de calidad aplicable al ciclo completo del desarrollo de software, desde el diseño hasta la instalación y mantenimiento del producto final. Las directrices que se establecen están orientadas hacia tres aspectos del sistema de calidad que se pretende implantar en las organizaciones de desarrollo:

- **Marco general para el establecimiento del Sistema de calidad:** en el que se describen las responsabilidades de la dirección de la organización de desarrollo de software, entre las que se encuentran la definición y documentación de aspectos tales como: su política de calidad; las responsabilidades, competencias y relaciones entre el personal cuyo trabajo incida en la calidad, y las necesidades de recursos para la dirección, ejecución del trabajo y actividades de verificación, incluyendo las auditorías internas de la calidad. En

este marco también se ha de describir el sistema de calidad, las auditorías internas del sistema de calidad, y las acciones correctivas.

- Actividades del ciclo de vida del software en el sistema de calidad: como las revisiones del contrato, la especificación de los requisitos del cliente, la planificación del desarrollo, la planificación de la calidad, el diseño e implantación del software, las pruebas y validación, la aceptación del producto final, y el mantenimiento.
- Actividades de apoyo en el sistema de calidad: como la gestión de la configuración, el control de documentos, el registro de calidad, la mediciones, y las reglas, prácticas y convenios.

En relación con los procesos de la ingeniería del software y con los estándares descritos anteriormente en este documento, hay que indicar que la implantación de los procesos establecidos en las normas indicadas facilitaría a las organizaciones el cumplimiento de los requisitos exigidos por ISO 9001, ya que la mayor parte de estos requisitos, que son muy generales, debe dar lugar a acciones (concretas) en las organizaciones que deberían formar parte de los procesos establecidos en la misma. Por ejemplo, entre los requisitos de ISO 9001 se encuentra la exigencia de "controlar los cambios del diseño y desarrollo", lo cual puede ser fácilmente realizable en una organización que haya implantado el proceso "Gestión de la configuración" previsto por ISO 15504 en la categoría de procesos de soporte; o el proceso CMMI con el mismo nombre y categoría, necesario para situarse en el nivel 2 de madurez (gestionado).

Normalmente una organización con nivel de madurez 3 de CMMI (o con procesos con capacidad 3 de ISO 15504) tendrá facilidad para obtener el certificado ISO 9001. Sin embargo, no se puede afirmar que una empresa con certificado ISO 9001 esté en un nivel superior al 1, ya que el nivel de detalle establecido por la norma ISO 9001 es mucho menor que el que se incluye en CMMI y SPICE para obtener las correspondientes certificaciones.

8 Metodologías de Ingeniería del Software

La forma de hacer realidad todo lo descrito hasta ahora en relación con los procesos de la ingeniería del software, es substanciar la idea genérica de "proceso software" en el concepto práctico de "metodología". Las organizaciones interesadas en el enfoque de proceso, deben definir y documentar su proceso software en forma de metodología.

Por definición, el concepto "metodología" hace referencia a "un conjunto de métodos, siendo un método un modo de hacer con orden" (DRAE 2001). Una

metodología es, por tanto, una forma explícita de estructurar el pensamiento y las acciones; una metodología debe señalar qué pasos tomar y cómo llevarlos a cabo.

Según Maddison, una metodología es un "conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores del software. Una metodología representa el camino para desarrollar software de una manera sistemática" (Maddison 1983).

De una forma más práctica, podríamos decir que una metodología de ingeniería del software debe establecer cómo dividir los proyectos en etapas, qué trabajos se llevan a cabo en cada etapa, qué artefactos (documentos, software) se producen en cada etapa y cuándo se deben producir, qué técnicas y herramientas se van a utilizar en cada etapa, qué restricciones se aplican (de tiempo, coste, objetivos, etc.), y cómo se gestiona y controla un proyecto.

Una metodología puede definirse para seguir uno o varios modelos de ciclo de vida, de tal forma que puede ser adaptada cuando se aplica a un proyecto de desarrollo de software concreto, seleccionando el modelo de ciclo de vida más apropiado, y las actividades, técnicas y herramientas más adecuadas para dicho proyecto.

8.1 Tipos de metodologías

En esta sección se presenta una clasificación de las diferentes metodologías que se han propuesto desde los años 70 en el ámbito de la ingeniería del software, en cinco categorías: metodologías estructuradas, orientadas a objetos, basadas en componentes, metodologías web y metodologías ágiles. A continuación, y en este mismo apartado, se dedican otras dos secciones a una descripción breve de dos de las metodologías mas completas publicadas, como son la metodología "oficial" española Métrica, y el "Proceso Unificado".

Metodologías de Ingeniería del Software estructurada

Las metodologías estructuradas, también denominadas "clásicas" u "orientadas a funciones", son las primeras que se aplicaron en el ámbito de la ingeniería del software. Estas metodologías tienen en común estar basadas en teorías sobre el modelado de sistemas establecidas en los años setenta por autores como Yourdon, DeMarco, Gane, Sarson, Constantine, Jackson, Martin, Warnier o Chen; que proponen la creación de modelos que representen las funciones de un sistema (también llamados procesos en este contexto); los flujos de datos de entrada, salida e internos de cada función del sistema; y la estructura de datos procesados para llevar a cabo las funciones. Y todo ello de una forma descendente, desde un nivel alto de abstracción (nivel de análisis o conceptual) hasta un nivel cercano

al entorno físico de implementación del sistema (nivel de diseño o lógico, y nivel de programación o físico).

Estas metodologías utilizan técnicas de análisis estructurado: Diagrama de Flujo de Datos, Diagrama de Entidad Relación, Diagrama de Historia de la Vida de una Entidad, Matriz de funciones-entidades; técnicas de diseño estructurado: Diagrama de Estructura Modular (o "de Constantine"), Diagrama de Módulos (o "de Jackson"), Diagrama de Estructura Lógica de Programa (o "de Warnier"), Diagrama de Arbol Programático (o "de Bertini"), "Diagrama de estructura de datos, Diagrama de tablas (de una Base de Datos), Diagrama de flujo de control (flowchart, organigrama, ordinograma), pseudocódigo; y técnicas de programación estructurada: Estructuras de control (secuencia, decisión (if), repetición (loop, for, while, repeat,...), subprogramas (que dan origen al tipo de programación estructurada conocida como programación modular), lenguajes de programación estructurada (Cobol, Fortran, C, Pascal, Natural, PL/SQL, etc.)).

Se han elaborado metodologías estructuradas por parte de organismos públicos y por organizaciones privadas, así como por autores particulares. Entre las primeras destacan: MÉTRICA, la metodología pública española; MERISE, metodología pública francesa; SSADM (*Structured System Analysis and Design Method*), metodología pública británica y V-MODEL, metodología pública alemana. Entre las privadas, algunas son las siguientes: INFORMATION ENGINEERING (IE), ORACLE METHOD, SAG PROJECT MODEL y SUMMIT-D.

Metodologías de Ingeniería del Software orientada a Objetos

Estas metodologías se empiezan a aplicar a finales de los años 80. Se basan en la idea de que un sistema software se compone de objetos software que interactúan entre sí. La funcionalidad del sistema se reparte entre los objetos, asignando a cada objeto funciones específicas. El objetivo final de la Ingeniería del Software Orientada a Objetos es construir software de la misma forma como se construye el hardware: mediante el ensamblaje de componentes. Las ventajas de la orientación a objetos son: la facilidad para la reutilización del software, la simplificación del mantenimiento del software, y la mejora de la calidad del software.

Las metodologías orientadas a objetos utilizan técnicas de análisis orientado a objetos: diagramas UML (Diagrama de Casos de Uso, Diagrama de Actividades, Diagrama de Secuencia, Diagrama de Colaboración, Diagrama de Estados, Diagrama de Clases), tarjetas CRC (*Class, Responsibility and Collaboration*); técnicas de diseño orientado a objetos: diagramas UML (los ya citados y también Diagrama de Objetos, Diagrama de Componentes, Diagrama de Despliegue), patrones de diseño; y técnicas de la programación orientada a objetos: herencia, polimorfismo, agregación, y lenguajes como Java, C++, C sharp, VB.Net,

Smalltalk, Eiffel.

Entre las metodologías orientadas a objetos más conocidas se encuentran: Proceso unificado, Fusion, OPEN, OMT, Booch method, Objectory y Métrica Versión 3.

Metodologías de Ingeniería del Software basada en componentes

Las metodologías de ingeniería del software basada en componentes están orientadas a la reutilización de componentes software ya desarrollados en el diseño y construcción de nuevos sistemas. Un componente software es "una parte reemplazable, casi independiente y no trivial de un sistema que cumple una función clara en el contexto de una arquitectura software bien definida" (Brown & Wallnau 1996).

La ingeniería del software basada en componentes surge a partir de la ingeniería del software orientada a objetos. Como en ésta, las metodologías prevén en los proyectos actividades de análisis de requisitos y de diseño arquitectónico, pero, en lugar de entrar inmediatamente en tareas de diseño detalladas, el equipo de desarrollo debe examinar los requisitos y la arquitectura para determinar qué elementos del software son susceptibles de composición (reutilización) y no de construcción.

Las metodologías basadas en componentes incluyen en el ciclo de vida actividades relacionadas con la **cualificación de componentes**, para determinar si es posible reutilizar componentes ya desarrollados por la misma organización, o comercializados por terceros¹¹, para implementar determinados requisitos; y con la **adaptación de los componentes reutilizados** al resto de elementos de la arquitectura del software.

Las técnicas utilizadas por este tipo de metodologías son las mismas que en el caso de la ingeniería del software orientada a objetos, si bien en este caso ampliadas con otras técnicas basadas en metadatos (ontologías, tesauros, diccionarios de términos, etc.) para la descripción de los componentes susceptibles de reutilización que se registran en repositorios, sobre los que se implementan operaciones de búsqueda de los componentes más adecuados en cada proyecto de desarrollo que pretenda la reutilización.

Metodologías de Ingeniería Web

La Ingeniería Web (*Web Engineering*) es un tipo de ingeniería del software orientada a la naturaleza dinámica y multidimensional de las aplicaciones web, que implican, además de programación, la definición de estructuras complejas de

¹¹Los componentes que se comercializan se denominan componentes COTS: *Commercial Off-The Shelf*.

información (normalmente basadas en XML); el diseño de estructuras de navegación, la gestión de contenidos; el diseño gráfico y multimedia; la gestión de aspectos de seguridad, legales, sociales y éticos; la gestión de diferentes perfiles de usuario; o el mantenimiento continuo que requieren este tipo de aplicaciones.

Las metodologías de ingeniería web suelen basarse en metodologías de ingeniería del software orientada a objetos, siendo habitual que se hayan publicado ampliaciones de metodologías orientadas a objetos ya existentes para adaptarlas a proyectos Web (Henderson-Sellers, Haire & Lowe 2001). En este caso, las ampliaciones necesarias se refieren a la inclusión, en el ciclo de vida, de actividades relacionadas con el diseño gráfico y multimedia, la generación de contenidos para las páginas web, el diseño de la navegación, o el diseño de las interfaces de usuario siguiendo estándares de usabilidad y teniendo en cuenta su personalización ante diferentes tipos de usuario. Además de nuevas actividades, las metodologías web también amplían los perfiles de los participantes en proyectos de desarrollo, incorporando las figuras de diseñador web, diseñador gráfico o editor de contenidos.

Las técnicas que exigen utilizar este tipo de metodologías también suelen ser extensiones de técnicas ya existentes, como UML, aunque también se han creado nuevas técnicas específicas para este tipo de aplicaciones¹². Entre las técnicas utilizadas para las actividades relacionadas con el análisis se encuentran las siguientes: Diagrama de Entidad Relación, Diagrama de Slices (Entidad Relación extendido), Diagrama de clases. Algunas técnicas de diseño web son: Diagrama de Presentación, Diagrama de Navegación, Diagrama de Sincronización, Diagrama de Composición, Diagrama de Personalización. Finalmente, entre las técnicas de programación web estarías los lenguajes de marcado (HTML, XML, ASP.NET, etc), los lenguajes de script (JavaScript, VBScript), los lenguajes para programación CGI (Perl, C, C++, etc.), y los lenguajes para programación de servidor (Java, VB.Net, C sharp, PHP, etc.).

En cuanto a las metodologías concretas que se han propuesto para el desarrollo de sistemas web con un enfoque de ingeniería, ha aparecido un gran número de ellas desde mediados de los 90, algunas creadas por autores provenientes del ámbito de la producción multimedia, como las metodologías HDM (Hypermedia Design Methodology), OOHDM (Object Oriented HDM), RMM (Relationship Management Methodology) o EORM (Enhanced Object Relationship Model). Otras son adaptaciones de métodos orientados a objetos, como OPEN o el Proceso Unificado. Y otras han sido creadas de forma específica para la Web, como el *WebML development process*.

¹²Una de las propuestas más completa de técnicas específicas para el modelado de aplicaciones web es WebML (*Web Modelling Language*): www.webml.org

Metodologías ágiles

Desde mediados de los años 90 ha empezado a surgir un tipo de metodologías denominadas inicialmente "ligeras", y posteriormente "ágiles", como alternativa a las metodologías más tradicionales, que los defensores del enfoque ágil denominan "monumentales" o "burocráticas", es decir, aquellas en las que hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda. Aunque en principio estas metodologías surgen como reacción a la burocracia de las metodologías monumentales, realmente las propuestas existentes buscan un punto intermedio entre no definir "ningún proceso" y establecer "demasiado proceso", proporcionando simplemente "suficiente proceso" para que el esfuerzo valga la pena (Fowler 2003).

Los principios que guían a este tipo de metodologías han quedado recogidos en el "*Agile Manifesto*" elaborado y suscrito en 2001 por sus principales creadores¹³. Son los doce principios (o "mandamientos") siguientes:

- La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software que le reporten valor.
- Hay que dar la bienvenida a los cambios en el software, para que el cliente tenga una ventaja competitiva.
- Es necesario entregar frecuentemente software que funcione, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente.
- Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo de los proyectos.
- Se deben crear los proyectos en torno a individuos motivados, ofreciéndoles el entorno y el apoyo que necesiten y confiar en ellos para la realización del trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona ha de ser la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible: los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

¹³El manifiesto se encuentra publicado en el sitio Web *agilemanifesto.org*.

- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- A intervalos regulares, el equipo debe reflexionar respecto a cómo llegar a ser más efectivo, y según esto ajustar su comportamiento.

Entre las metodologías ágiles mas conocidas se encuentran: Programación Extrema (XP: eXtreme Programming), Crystal, ASD (Adaptable Software Development), Scrum y DSDM (Dynamic Systems Development Method).

8.2 Metodología "Métrica"

MÉTRICA es la metodología oficial española que debe utilizarse en el desarrollo de Sistemas de Información para la Administración del Estado¹⁴; aunque son muchas las empresas la han asumido como metodología propia para el desarrollo de todos sus proyectos, sean o no para la Administración. La primera versión data de 1989, apareciendo una segunda versión en 1995, y la tercera, denominada Versión 3, en 2001 (MAP 2001). Está patrocinada por el Consejo Superior de Informática del Ministerio para las Administraciones Públicas, y ha sido elaborado a partir de la colaboración entre empresas de desarrollo y de consultoría, universidades, y el propio Consejo.

La metodología Métrica es una de las metodologías de desarrollo de software más completa que se han publicado. Como se pone de manifiesto en el propio documento que describe la metodología, su objetivo es ofrecer a las organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software dentro del marco que permite alcanzar los siguientes objetivos:

- Proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la Organización mediante la definición de un marco estratégico para el desarrollo de los mismos.
- Dotar a la Organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos
- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad

¹⁴Métrica está enfocada al desarrollo del software que forma parte de los sistemas de información, aunque esta metodología también establece actividades relacionadas con el resto de elementos de este tipo de sistemas, como el hardware y las operaciones manuales.

de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible

- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos.
- Facilitar la operación, mantenimiento y uso de los productos software obtenidos.

La metodología Métrica contempla el desarrollo de Sistemas de Información para las distintas tecnologías que actualmente están conviviendo y los aspectos de gestión que aseguran que un proyecto cumple sus objetivos en términos de calidad, coste y plazos. En la Versión 3 de esta metodología, el ciclo de vida de un proyecto se divide en una serie de fases que se denominan "procesos", atendiendo a la definición de este concepto como transformación de unas entradas (provenientes de otros procesos) en unas salidas (que se dirigen, como entrada, a otros procesos).

Los procesos se descomponen en actividades y éstas en tareas. Para cada tarea se detallan los participantes que intervienen, los productos de entrada y de salida así como las técnicas y prácticas a emplear para su obtención. En la elaboración de Métrica se han tenido en cuenta los métodos de desarrollo más extendidos, así como los estándares ISO 12207 e ISO 15504 de ingeniería del software y el estándar ISO 9001 sobre calidad; además de referencias específicas en cuanto a seguridad y gestión de proyectos. También se ha tenido en cuenta la experiencia de los usuarios de las versiones anteriores para solventar los problemas o deficiencias detectados.

En una única estructura, la metodología Métrica cubre distintos tipos de desarrollo: estructurado y orientado a objetos, facilitando a través de lo que denomina "interfaces" la realización de procesos de apoyo u organizativos, como los de gestión de proyectos, gestión de configuración, aseguramiento de la calidad, y de seguridad. La automatización de las actividades propuestas en la estructura de Métrica Versión 3 es posible ya que las técnicas que sugiere aplicar a lo largo del ciclo de vida del software son estándar (la mayoría de las indicadas en el apartado anterior de este documento, como DFD, ERD, UML, etc.), y están soportadas por una amplia variedad de herramientas CASE de ayuda al desarrollo.

Procesos principales de Métrica

Métrica tiene un enfoque orientado al proceso, en consonancia con la tendencia general en los estándares en este campo, y por ello se ha enmarcado dentro de

la norma ISO 12207, ya tratada en este documento. Como punto de partida y atendiendo a dicha norma, Métrica cubre el Proceso de Desarrollo y el Proceso de Mantenimiento. Esta metodología ha sido concebida para abarcar el desarrollo

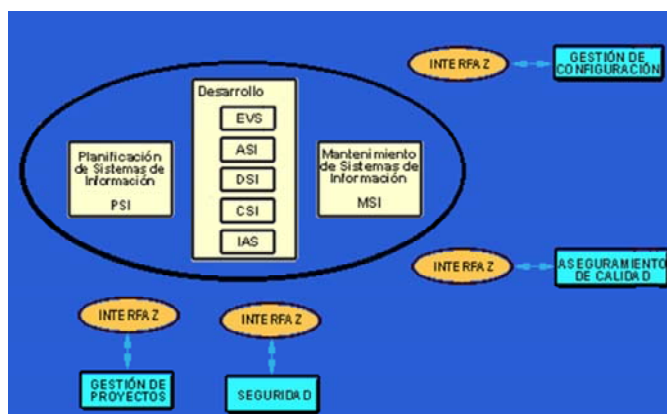


Figura 1.9: Procesos e Interfaces en la metodología Métrica

completo de Sistemas de Información sea cual sea su complejidad y magnitud, por lo cual su estructura responde a desarrollos máximos y debe adaptarse y dimensionarse en cada momento de acuerdo a las características particulares de cada proyecto.

Como ya se ha indicado, la metodología descompone cada uno de los procesos en actividades, y éstas a su vez en tareas. Para cada tarea se describe su contenido haciendo referencia a sus principales acciones, productos, técnicas, prácticas y participantes. El orden numérico que Métrica asigna a las actividades no debe interpretarse como secuencia en su realización, ya que éstas pueden realizarse en orden diferente a su numeración o bien en paralelo, pero no se debe dar por acabado un proceso hasta no haber finalizado todas las actividades del mismo determinadas al inicio del proyecto.

Los procesos de la estructura principal de Métrica Versión 3 son los siguientes:

- **Proceso de Planificación de Sistemas de Información (PSI):** Al no estar dentro del ámbito de la norma ISO 12207, los autores de Métrica han determinado este proceso a partir del estudio de otras propuestas en este campo. El objetivo de un Plan de Sistemas de Información es proporcionar un marco estratégico de referencia para los Sistemas de Información de un determinado ámbito de la Organización. El resultado del Plan de Sistemas debe, por tanto, orientar las actuaciones en materia de desarrollo de Sistemas de Información con el objetivo básico de apoyar la estrategia corporativa, elaborando una arquitectura de información y un plan de proyectos informáticos para dar apoyo a los objetivos estratégicos.

- **Proceso de desarrollo de sistemas de información:** Para facilitar la comprensión y dada su amplitud y complejidad, este proceso de ISO 12207 se ha subdividido en cinco procesos:
 - **Estudio de Viabilidad del Sistema (EVS).** El propósito de este proceso es analizar un conjunto concreto de necesidades, con la idea de proponer una solución a corto plazo. Los criterios con los que se hace esta propuesta no deben ser estratégicos sino tácticos, y han de estar relacionados con aspectos económicos, técnicos, legales y operativos.
 - **Análisis del Sistema de Información (ASI).** El propósito de este proceso es conseguir la especificación detallada del sistema de información, a través de un catálogo de requisitos y una serie de modelos que cubran las necesidades de información de los usuarios para los que se desarrollará el sistema de información y que serán la entrada para el proceso de Diseño del Sistema de Información.
 - **Diseño del Sistema de Información (DSI).** El propósito de este proceso es obtener 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. A partir de dicha información, se generan todas las especificaciones de construcción relativas al propio sistema, así como la especificación técnica del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial, éstos últimos cuando proceda.
 - **Construcción del Sistema de Información (CSI).** Su objetivo es la construcción y prueba de los distintos componentes del sistema de información, a partir del conjunto de especificaciones lógicas y físicas del mismo obtenido en el Proceso de Diseño del Sistema de Información (DSI). Se desarrollan los procedimientos de operación y seguridad y se elaboran los manuales de usuario final y de explotación, estos últimos cuando proceda.
 - **Implantación y Aceptación del Sistema (IAS).** Este proceso tiene como objetivo principal, la entrega y aceptación del sistema en su totalidad, que puede comprender varios sistemas de información desarrollados de manera independiente, según se haya establecido en el proceso de Estudio de Viabilidad del Sistema (EVS), y un segundo objetivo que es llevar a cabo las actividades oportunas para el paso a producción del sistema.

- **Proceso de Mantenimiento de Sistemas de Información (MSI):** Comprende actividades y tareas de modificación o retirada de todos los componentes de un sistema de información (hardware, software, software de base, operaciones manuales, redes, etc.). Este marco de actuación no es el objetivo de Métrica, ya que esta metodología está dirigida principalmente al proceso de desarrollo del software. Por lo tanto, Métrica refleja los aspectos del Mantenimiento, correctivo y evolutivo, que tienen relación con el Proceso de Desarrollo.

Procesos organizacionales y de apoyo de Métrica

La metodología incluye también un conjunto de interfaces que definen una serie de actividades de tipo organizativo o de soporte al proceso de desarrollo y a los productos, que en el caso de existir en la organización se deben aplicar para enriquecer o influir en la ejecución de las actividades de los procesos principales de la metodología, y que si no existen hay que realizar para complementar y garantizar el éxito del proyecto desarrollado con Métrica. Dos de las cuatro interfaces descritas en la metodología facilitan la realización de los procesos considerados "de apoyo" por el estándar ISO 12207 (aseguramiento de la calidad, gestión de la configuración), otra facilita la realización de uno de los procesos organizacionales de ISO 12207 (Gestión de proyectos), y la cuarta (Seguridad) no está contemplada en el estándar.

- **Gestión de Proyectos (GPS):** La Gestión de Proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información. 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. La Interfaz de Gestión de Proyectos de Métrica contempla tanto los proyectos de desarrollo de nuevos Sistemas de Información como los proyectos de ampliación y mejora de los ya existentes.
- **Seguridad (SEG):** El objetivo de la interfaz de seguridad es incorporar en los sistemas de información mecanismos de seguridad adicionales a los que se proponen en la propia metodología, asegurando el desarrollo de cualquier tipo de sistema a lo largo de los procesos que se realicen para su obtención. La interfaz de Seguridad hace posible incorporar durante la fase de desarrollo las funciones y mecanismos que refuerzan la seguridad del nuevo sistema y del propio proceso de desarrollo, asegurando su consistencia y seguridad, completando el plan de seguridad vigente en la organización o desarrollándolo desde el principio.

- **Gestión de la Configuración (GC):** La interfaz de gestión de la configuración consiste en la aplicación de procedimientos administrativos y técnicos durante el desarrollo del sistema de información y su posterior mantenimiento. 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. Este proceso permite conocer el estado de cada uno de los productos que se hayan definido como elementos de configuración, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan.
- **Aseguramiento de la Calidad (CAL):** El objetivo de la interfaz de Aseguramiento de la Calidad 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. Las actividades propias de la interfaz de Calidad están orientadas a verificar la calidad de los productos. Son actividades que evalúan la calidad, y deben ser realizadas por un grupo de asesoramiento de la calidad independiente de los responsables de la obtención de los productos.

Comparación con otras metodologías

Para concluir la sección dedicada a Métrica se va a realizar una comparación de esta metodología con otras metodologías "oficiales" también del ámbito europeo, como son MERISE y SSADM.

Merise Es la metodología patrocinada por el gobierno francés, con una gran difusión en Francia, Bélgica, España (hasta la aparición de Métrica), Suiza, Italia, América del Norte y el Magreb. La primera versión de esta metodología data de 1982 (Tardieu, Rochfeld, Colletti, Panet & Vahée 1985). La metodología Merise considera el proceso de desarrollo de software estructurado en seis etapas que se subdividen en un cierto número de fases, las cuales a su vez se llevan a cabo a través de una serie de subfases o tareas, que constituyen la menor unidad de trabajo controlable por la metodología. En la siguiente tabla se muestra la relación de estas etapas con Métrica, así como las equivalencias terminológicas, entre ambas. En la tabla también se indican las técnicas que Merise establece que deben utilizarse en un proyecto, comparándalas con las técnicas que podrían considerarse equivalentes en Métrica.

SSADM (*Structured Systems Analysis and Design Method*) es la metodología patrocinada por el gobierno británico, que también es oficial en lugares como Irlanda, Israel, Hong Kong o Malta. La primera versión de SSADM se publicó en 1980. No se trata de una metodología de ciclo completo como Merise o Métrica,

METRICA	MERISE
COMPARACIÓN DE ACTIVIDADES DEL CICLO DE VIDA	
Las fases de un proyecto se denominan PROCESOS	Las fases de un proyecto se denominan ETAPAS
Los PROCESOS se dividen en ACTIVIDADES	Las ETAPAS se dividen en FASES
Las ACTIVIDADES se dividen en TAREAS	Las FASES se dividen en TAREAS
PROCESO DE PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN (PSI)	ETAPA DE ESQUEMA DIRECTOR
PROCESO DE ESTUDIO DE VIABILIDAD DEL SISTEMA DE INFORMACIÓN (EVS)	ETAPA DE ESTUDIO PREVIO
PROCESO DE ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI)	ETAPA DE ESTUDIO DETALLADO
PROCESO DE DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI)	ETAPA DE ESTUDIO TÉCNICO
PROCESO DE CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	
PROCESO DE IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA (IAS)	ETAPA DE REALIZACIÓN Y PUESTA EN MARCHA
PROCESO DE MANTENIMIENTO DEL SISTEMA DE INFORMACIÓN (MSI)	ETAPA DE MANTENIMIENTO
COMPARACIÓN DE TÉCNICAS UTILIZADAS	
DIAGRAMA DE FLUJO DE DATOS (DFD)	MODELO CONCEPTUAL DE TRATAMIENTOS (MCT)
DIAGRAMA DE ESTRUCTURA DE CUADROS	MODELO ORGANIZATIVO DE TRATAMIENTOS (MOT)
	MODELO OPERATIVO DE TRATAMIENTOS (MOPT)
DIAGRAMA ENTIDAD RELACION (ERD)	MODELO CONCEPTUAL DE DATOS
	MODELO LÓGICO DE DATOS
DIAGRAMA FÍSICO DE DATOS	MODELO FÍSICO DE DATOS
COMPARACIÓN DE DOCUMENTOS GENERADOS	
PLANIFICACIÓN	PLAN DE DESARROLLO
ESTUDIO DE VIABILIDAD DEL SISTEMA	DOSIER DE OPCIIONES
ANÁLISIS DEL SISTEMA DE INFORMACIÓN	DOCUMENTO DE ESTUDIO DETALLADO
DISEÑO DEL SISTEMA DE INFORMACIÓN	DOCUMENTO DE ESTUDIO TÉCNICO
	CUADERNO DE CARGAS

Figura 1.10: Comparación de las metodologías Métrica y Merise

ya que sólo considera aspectos de análisis, incluido el estudio de viabilidad, y diseño; sin tener en cuenta las tareas previas de planificación o las posteriores de construcción e implantación del software desarrollado. Las técnicas que establece SSADM son estructuradas y permiten el modelado de las funciones y los datos de los Sistemas a desarrollar. En la siguiente tabla se realiza una comparación de esta metodología con Métrica.

8.3 Metodología "Proceso Unificado"

El Proceso Unificado es una metodología de desarrollo de software orientado a objetos, obtenida como combinación de otros métodos definidos previamente por sus tres autores: Objetary (I. Jacobson), OMT (J. Rumgaugh) y Booch Method (G. Booch). El Proceso Unificado¹⁵ es un proceso de desarrollo de software, y como tal, estructurado en un conjunto de actividades para transformar los requisitos de usuario en un sistema de software. Pero también puede considerarse un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El Proceso Unificado está basado en componentes, lo que quiere decir que el sistema software en construcción está formado por componentes software inter-

¹⁵Inicialmente conocido como RUP: *Rational Unified Process*

METRICA	SSADM (Structured Systems Analysis and Design Method)
COMPARACIÓN DE ACTIVIDADES DEL CICLO DE VIDA	
Las fases de un proyecto se denominan PROCESOS Los PROCESOS se dividen en ACTIVIDADES Las ACTIVIDADES se dividen en TAREAS	Las fases de un proyecto se denominan MODULOS Los MODULOS se dividen en ETAPAS Las ETAPAS se dividen en PASOS
PROCESO DE PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN (PSI)	La metodología SSADM no contempla este proceso, sólo los de análisis y diseño.
PROCESO DE ESTUDIO DE VIABILIDAD DEL SISTEMA DE INFORMACIÓN (EVS)	MODULO DE ESTUDIO DE VIABILIDAD (FS: FEASIBILITY STUDY)
PROCESO DE ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI)	MODULO DE ANÁLISIS DE REQUISITOS (RA: REQUIREMENTS ANALYSIS) + MODULO DE ESPECIFICACIÓN DE REQUISITOS (RS: REQUIREMENTS SPECIFICATION) + MODULO DE ESPECIFICACIÓN DEL SISTEMA LÓGICO (LS: LOGICAL SYSTEM SPECIFICATION)
PROCESO DE DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI)	MODULO DE DISEÑO FÍSICO (PD: PHYSICAL DESIGN)
PROCESO DE CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	La metodología SSADM no contempla este proceso, sólo los de análisis y diseño.
PROCESO DE IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA (IAS)	La metodología SSADM no contempla este proceso, sólo los de análisis y diseño.
PROCESO DE MANTENIMIENTO DEL SISTEMA DE INFORMACIÓN (MSI)	La metodología SSADM no contempla este proceso, sólo los de análisis y diseño.
COMPARACIÓN DE TÉCNICAS UTILIZADAS	
DIAGRAMA DE FLUJO DE DATOS (DFD)	DIAGRAMA DE FLUJO DE DATOS (DFD)
DIAGRAMA ENTIDAD RELACION (ERD)	ANÁLISIS RELACIONAL DE DATOS (RDA)
	MODELO LÓGICO DE DATOS (LDW)
DIAGRAMA DE ESTRUCTURA DE CUADROS	DIAGRAMA DE ESTRUCTURA DE CUADROS
DIAGRAMA FÍSICO DE DATOS	DIAGRAMA DE ESTRUCTURA DE DATOS (DSD)
No existe un diagrama equivalente en METRICA.	HISTORIA DE LA VIDA DE LAS ENTIDADES (ELH)
COMPARACIÓN DE DOCUMENTOS GENERADOS	
ESTUDIO DE VIABILIDAD DEL SISTEMA	INFORME DE VIABILIDAD
ANÁLISIS DEL SISTEMA DE INFORMACIÓN	DOCUMENTO DE ANÁLISIS DE REQUISITOS DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS DOCUMENTO DE ESPECIFICACIÓN DEL SISTEMA LÓGICO
DISEÑO DEL SISTEMA DE INFORMACIÓN	DOCUMENTO DE DISEÑO FÍSICO

Figura 1.11: Comparación de las metodologías Métrica y SSADM

conectados a través de interfases bien definidas. Utiliza el Lenguaje Unificado de Modelado (UML) para modelar todos los aspectos de un sistema software. El ciclo de vida establecido por esta metodología para los proyectos es bidimensional, y se compone de fases, flujos de trabajo e iteraciones (figura 1.12).

Según esta metodología, un proyecto se divide en fases (Inicio, Elaboración, Construcción y Transición), y cada fase se realiza mediante iteraciones. Cada iteración es como un miniproyecto que pasa por las siguientes actividades (llamadas flujos de trabajo): Requisitos-Análisis-Diseño-Implementación-Prueba, en la que se obtiene una nueva versión interna del software. La diferencia entre la versión de una iteración y la versión de la siguiente es un incremento, que representa el crecimiento del producto software.

Los objetivos de las fases establecidas por la metodología son los siguientes:

- **Inicio.** La fase de inicio establece la viabilidad. Se define el alcance del proyecto y se desarrollan los casos de negocio. Se deben establecer los objetivos del ciclo de vida para el proyecto.
- **Elaboración.** La fase de elaboración se centra en la factibilidad. Se planifica el proyecto, se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura del sistema. Se obtiene la línea base de

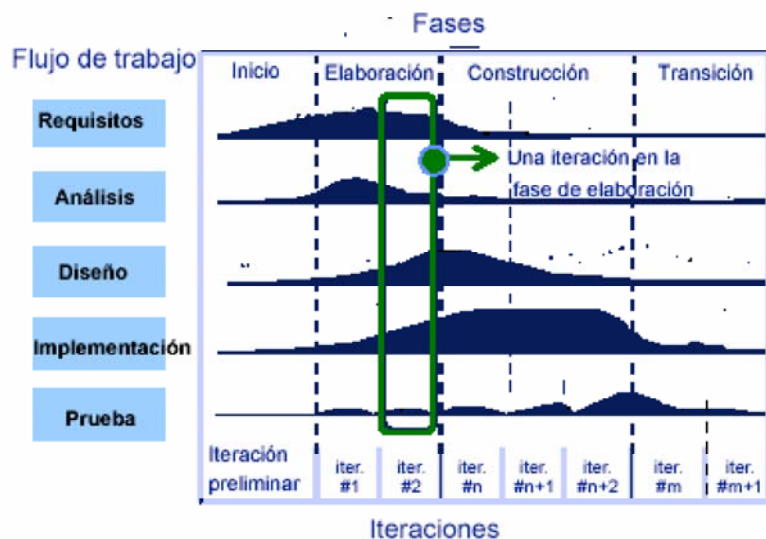


Figura 1.12: Ciclo de vida del Proceso Unificado

la arquitectura, capturando la mayoría de los requisitos y reduciendo los peores riesgos, pudiendo estimar costes y plazos.

- **Construcción.** En la fase de construcción se construye el sistema. Al final de esta fase se debe garantizar que el producto puede comenzar su transición a los clientes, es decir, que tiene una funcionalidad operativa inicial.
- **Transición.** La fase de transición se centra en el entorno del usuario. El producto se convierte en versión beta. Se corrigen problemas y se incorporan mejoras sugeridas en la revisión. Se debe garantizar que hay un producto preparado para entregar a los usuarios y se debe formar a estos en su utilización.

Los objetivos de los flujos de trabajo de cada iteración son:

- **Captura de requisitos.** El objetivo de este flujo de trabajo es guiar el desarrollo hacia el sistema correcto, de forma que el cliente debe ser capaz de leer y comprender el resultado y que este resultado ayude al jefe de proyecto a planificar las iteraciones. Se consigue, además, reducir los riesgos.
- **Análisis.** Durante este flujo de trabajo se analizan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. Para ello se utilizan modelos de análisis, que ayudan a refinar los requisitos

y permiten razonar sobre los aspectos internos del sistema, facilitando la identificación y planificación de incrementos. Finalmente proporciona una visión general del sistema.

- **Diseño.** Se modela el sistema y se encuentra su forma para que soporte todos los requisitos que se le suponen. Una entrada esencial en el diseño es la salida del análisis. El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción.
- **Implementación.** Se parte del diseño y se implementa el sistema en términos de componentes, planificando las integraciones y probando los componentes individualmente e integrados.
- **Prueba.** Se verifica el resultado de la implementación probando cada construcción. Se llevan a cabo sobre todo cuando una construcción es sometida a pruebas de integración y de sistema. Se deben planificar, diseñar, implementar y realizar las pruebas, manejando los resultados de forma que los defectos importantes puedan ser corregidos.

Comparación con la metodología Métrica

Como ya se ha indicado anteriormente, el Proceso Unificado es una metodología orientada a objetos, mientras que Métrica Versión 3 es una metodología que puede adaptarse tanto al enfoque estructurado como al orientado a objetos, mediante la selección de unas determinadas actividades u otras de las establecidas para cada uno de los procesos de desarrollo. Por ejemplo, en el proceso de Análisis, si se trata de un proyecto estructurado, establece la realización de las actividades "Elaboración del modelo de datos" y "Elaboración del modelo de procesos", que serían substituidas por las actividades "Análisis de casos de uso" y "Análisis de Clases", si se tratara de un proyecto orientado a objetos.

En la siguiente tabla se muestra una comparación entre ambas metodologías, en lo que respecta a terminología, actividades y técnicas.

9 Modelado y automatización del proceso software

La definición de un proceso software y de sus correspondientes subprocesos debe concretar y ordenar las actividades y tareas que se han de realizar para alcanzar los objetivos establecidos para ese proceso. Esto puede hacerse a través de una especificación informal, por ejemplo, en lenguaje natural; como ocurre, por ejemplo, en el documento original de descripción de una metodología. Sin embargo,

METRICA	PROCESO UNIFICADO (ORIENTADO A OBJETOS)
COMPARACIÓN DE ACTIVIDADES DEL CICLO DE VIDA	
Las fases de un proyecto se denominan PROCESOS Los PROCESOS se dividen en ACTIVIDADES Las ACTIVIDADES se dividen en TAREAS	Las fases de un proyecto se denominan FASES Las FASES se realizan a base de ITERACIONES (repetiendo cada fase varias veces, cada vez más detalle, antes de pasar a la siguiente fase) Cada FASE se divide en los siguientes FLUJOS DE TRABAJO: Requisitos, Análisis, Diseño, Implementación, Prueba
PROCESO DE PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN (PSI)	La metodología PROCESO UNIFICADO no contempla este proceso.
PROCESO DE ESTUDIO DE VIABILIDAD DEL SISTEMA DE INFORMACIÓN (EVS)	FASE DE INICIO
PROCESO DE ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI)	FASE DE ELABORACIÓN
PROCESO DE DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI)	
PROCESO DE CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	FASE DE CONSTRUCCIÓN
PROCESO DE IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA (IAS)	FASE DE TRANSICIÓN
PROCESO DE MANTENIMIENTO DEL SISTEMA DE INFORMACIÓN (MSI)	La metodología PROCESO UNIFICADO no contempla este proceso.
COMPARACIÓN DE TÉCNICAS UTILIZADAS	
DIAGRAMA DE CASOS DE USO	DIAGRAMA DE CASOS DE USO
DIAGRAMA DE CLASES	DIAGRAMA DE CLASES
DIAGRAMA DE SECUENCIA	DIAGRAMA DE SECUENCIA
DIAGRAMA DE COLABORACIÓN	DIAGRAMA DE COLABORACIÓN
DIAGRAMA DE TRANSICIÓN DE ESTADOS	DIAGRAMA DE TRANSICIÓN DE ESTADOS
DIAGRAMA DE COMPONENTES	DIAGRAMA DE COMPONENTES
DIAGRAMA DE DESPLIEGUE	DIAGRAMA DE DESPLIEGUE
COMPARACIÓN DE DOCUMENTOS GENERADOS	
METRICA establece documentos concretos para todos los procesos del ciclo de vida de los proyectos.	El PROCESO UNIFICADO básicamente sólo establece un único documento, denominado DESCRIPCIÓN DE LA ARQUITECTURA, en el que se registra primero el resultado del Análisis y después es sustituido por el resultado del Diseño.
ANÁLISIS DEL SISTEMA DE INFORMACIÓN	DESCRIPCIÓN DE LA ARQUITECTURA DEL SISTEMA (VISTA DEL MODELO DE ANÁLISIS)
DISEÑO DEL SISTEMA DE INFORMACIÓN	DESCRIPCIÓN DE LA ARQUITECTURA DEL SISTEMA (VISTA DEL MODELO DE DISEÑO)

Figura 1.13: Comparación de las metodologías Métrica y Proceso Unificado

también es posible utilizar una técnica de especificación formal para describir el proceso, lo que facilitaría el control de su ejecución mediante herramientas informáticas.

Como se indicó al principio de este documento, el proceso software puede considerarse un tipo de proceso de negocio; y puesto que desde principios de los 90 se están utilizando tecnologías para el modelado y ejecución del trabajo implicado en los procesos de negocio, no es extraño que estas ideas se hayan trasladado al ámbito de la ingeniería del software para tratar de conseguir que los procesos software se realicen de una forma más productiva.

El origen del modelado y automatización del proceso se encuentra en los sistemas de flujo de trabajo (*workflow systems*). En 1993 se creó la *Workflow Management Coalition* (WfMC), formada por compañías que comercializan productos de workflow, usuarios y analistas, con el fin de desarrollar y promocionar estándares sobre terminología, conectividad e interoperabilidad entre este tipo de productos (www.wfmc.org)¹⁶. Un sistema de flujo de trabajo es aquel que

¹⁶En España, el Ministerio para las Administraciones Públicas, ha publicado, basándose principalmente en el trabajo de esta coalición, un conjunto de especificaciones, bajo la denominación de ESTROFA (ESpecificaciones para el TRatamiento de Flujos Automatizados), que exige la Administración española a los productos (sistemas) de workflow que se adquieren desde 1996 (MAP 1995).

permite definir, ejecutar y gestionar procesos y tareas en base a unas reglas. En este contexto, por proceso se entiende un conjunto de tareas ordenadas, bien temporalmente, bien cumpliendo condiciones contenidas en reglas, que son realizadas bien por sujetos competentes (usuarios o grupos de usuarios organizados jerárquicamente con capacidad para realizar las tareas), o bien de forma automatizada (por autorización expresa del sujeto competente). Un proceso puede componerse de uno o varios subprocesos, que a su vez pueden descomponerse en tareas, admitiéndose todo tipo de conjunciones y disyunciones en la composición.

Los procesos pueden ser reglamentados (estructurados), si están bien definidos a través de un flujo de actividades que normalmente siempre se cumplirá; abiertos (no reglamentados o no estructurados), en los que cada actividad da origen a la siguiente, sin que sea posible definir "a priori" su flujo, con lo que los sujetos competentes pueden integrar y ordenar tareas, y asignar reglas de forma dinámica. Como caso intermedio se encuentran los procesos semi-reglamentados, aquellos reglamentados en los que se prevén excepciones que pueden alterarlo dinámicamente a voluntad de los sujetos autorizados para ello.

En cuanto al concepto de tarea, se trata de la unidad mínima de trabajo que, combinada con otras tareas, constituye un proceso. Las tareas pueden ser manuales, semiautomatizadas y automatizadas. Las automatizadas son tareas que el sistema pone a disposición de los sujetos competentes para que sean realizadas por ejecución de una regla. Las semiautomatizadas, por el contrario, se realizan a petición expresa y manual.

Los sistemas de workflow normalmente están formados por tres entornos de trabajo¹⁷: 1) un entorno de modelado y diseño, en el que, mediante un lenguaje gráfico, se puedan representar flujos de trabajo (flujogramas), y se puedan definir, entre otros, los atributos de tareas, procesos, sujetos, reglas, tratamiento de excepciones, objetos (documentos) y encaminamiento de objetos y tiempos; 2) un entorno de administración, supervisión y simulación, con facilidades para la monitorización de los estados de ejecución los procesos y tareas ("en espera", "cancelada", etc.) en cualquier momento, facilidades de reasignación dinámica de tareas, reglas prioridades y permisos, facilidades de información, proporcionando estadísticas parametrizables por tiempo y coste, y facilidades de gestión de la seguridad, para el control de accesos, para la recuperación automática en caso de error, y para asegurar la coherencia interna de los procesos; y 3) un entorno de ejecución y usuario final, con un interface personalizable de acuerdo con las labores que debe realizar cada usuario, a través del cual conozca el trabajo que tiene asignado, y un sistema de alertas y notificación de problemas, como vencimiento de plazos, fechas límite, entregas rechazadas, tareas automatizadas no disponibles, etc.

¹⁷En España así lo exige la norma ESTROFA.

Toda esta tecnología relacionada con el workflow ha sido incorporada al ámbito de la ingeniería del software, como vía para conseguir la automatización del proceso de desarrollo de software. En los siguientes apartados se analizan las propuestas relativas al modelado de los flujos de trabajo correspondientes a los procesos software, y los entornos y herramientas informáticas que dan soporte a la ejecución de esos procesos.

9.1 Modelado del proceso software

Al igual que ha ocurrido en el ámbito de los sistemas de workflow, también en el campo de la ingeniería del software se ha propuesto técnicas para el modelado formal de las actividades implicadas en un proceso software. Existe un gran número de los denominados Lenguajes de Modelado de Procesos (PML: *Process Modeling Languages*), y de entornos de automatización del proceso software que son capaces de "ejecutar" los modelos realizados con estos lenguajes.

Entre los intentos por normalizar este tipo de lenguajes para el modelado, una de las propuestas mas importantes es el estándar SPEM (*Software Process Engineering Metamodel*) del Object Management Group (OMG 2005), cuyo objetivo es ofrecer un "metamodelo que pueda ser utilizado para describir procesos de desarrollo de software concretos, o una familia de procesos relacionados". La ventaja de este estándar es que se basa en UML, ampliamente conocido en el ámbito de la ingeniería del software.

SPEM ofrece a los "ingenieros de procesos" elementos gráficos (iconos), similares a los esterotipos de UML, para crear los modelos que representen los procesos software. Algunos de estos elementos son: *Process*, *ProcessPerformer*, *ProcessRole*, *Activity*, *Step*, *WorkProduct*, *Document*, etc. En la figura 1.14 se representa un ejemplo simplificado de modelo de proceso utilizando la notación de SPEM, en el que aparecen, entre otros elementos, actividades (*Define Requirements*, *Refine User Interface*", *Build Application*, etc.), realizadores de procesos (*Functional Analyst*, *Interface Designer*, *Technical Designer*), y documentos (*User Requirements*, *User Interface*, etc.).

Una vez realizado un modelo de proceso con un lenguaje formal, por ejemplo, basado en SPEM, el siguiente paso sería la ejecución del proceso representado a través del modelo. Si se dispusiera de un entorno de ingeniería del software orientado a procesos, como los denominados PSEE que se describen en el siguiente apartado, éste podría tomar el modelo y obligar a los participantes en un proyecto de desarrollo en particular a seguir el proceso según se ha definido, es decir controlando el orden de ejecución de las actividades; automatizando el intercambio de documentos a través de correo electrónico; o activando las herramientas CASE apropiadas para realizar determinadas actividades (por ejemplo, una herramienta de prototipado para realizar la actividad *Draft User Interface* del

diagrama de la figura 1.14). Normalmente estos entornos requieren que antes de que se proceda a la ejecución del proceso, se realice lo que se conoce en este ámbito como la "reificación"¹⁸ del modelo creado, es decir la particularización e instanciación de los elementos que aparecen en el modelo para la preparación de su ejecución en el entorno PSEE.

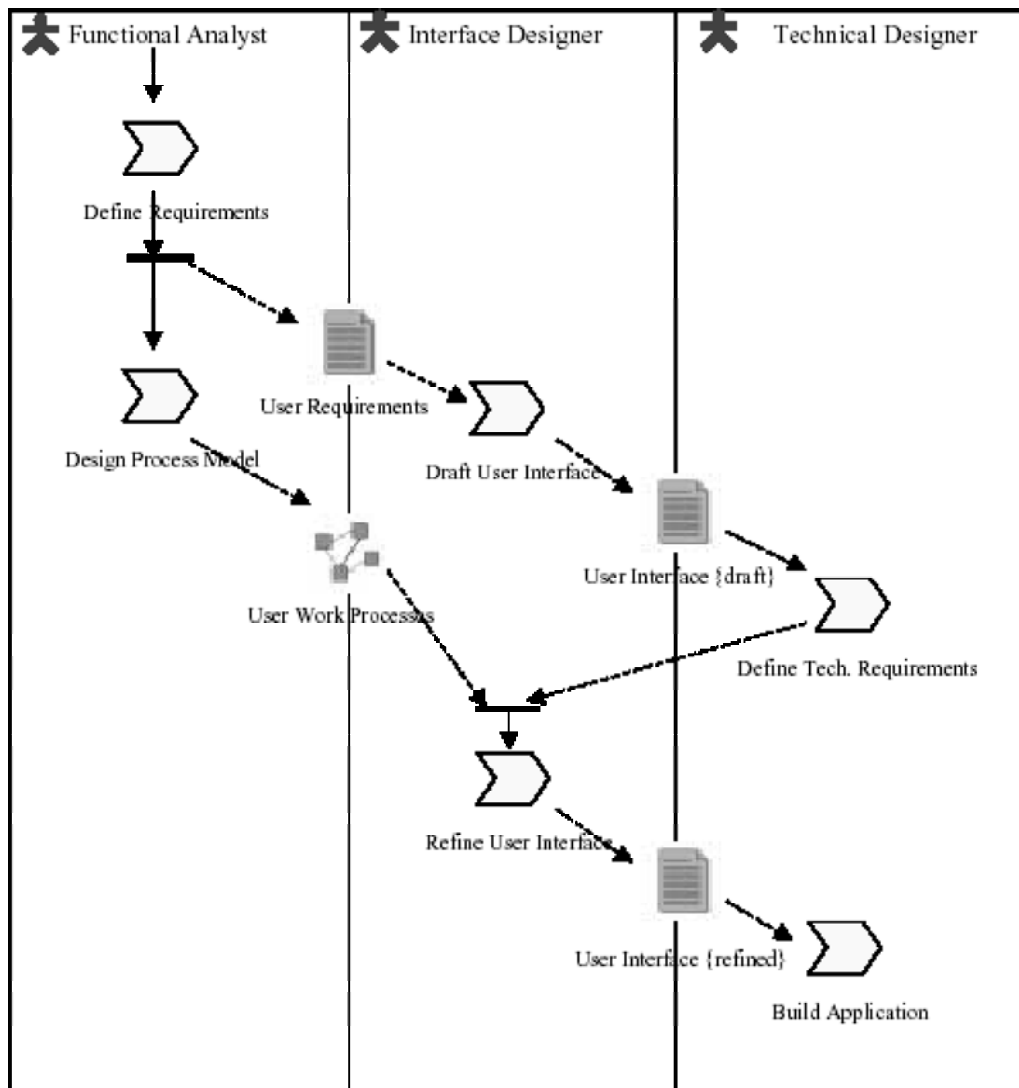


Figura 1.14: Ejemplo de modelo de proceso (OMG, 2005)

¹⁸Es la traducción que suele hacerse en este contexto del término *enactment*.

9.2 Entornos de automatización del proceso software

El concepto de herramienta CASE normalmente se asocia a los entornos que proporcionan un soporte basado en computador para las diferentes actividades de un proyecto de desarrollo de software. Sin embargo, fruto de un intenso trabajo de investigación que comenzó a mediados de la década de los 80 han surgido entornos que consideran el propio desarrollo de software como la ejecución de un proceso que puede ser definido explícitamente, se trata de los denominados entornos de ingeniería del software basados en el proceso o, en inglés, *Process-Centered Software Engineering Environments* (PSEEs). En este apartado se describirán éstas y otras tecnologías que dan soporte a la automatización de los procesos de la Ingeniería del Software.

Herramientas CASE

En la actualidad ya no se concibe el desarrollo de software sin utilizar herramientas de ayuda para automatizar gran parte de las tareas del ciclo de vida del software. Esto dió lugar en la década de los 80 al nacimiento de la "Ingeniería del software Asistida por Ordenador" o tecnología CASE (*Computer Aided Software Engineering*), que consiste en una combinación de metodología de desarrollo (MERISE, SSADM, MÉTRICA, Proceso Unificado, etc.) y herramientas orientadas a facilitar dicho desarrollo basado en esas metodologías concretas (Oracle Designer, Rational Rose, System Architect, MS Visio, etc.).

La tecnología CASE reemplaza el papel y el lápiz por el ordenador para hacer del desarrollo del software un proceso más productivo. El objetivo fundamental de CASE es proporcionar un conjunto de herramientas que automaticen los trabajos de desarrollo y mantenimiento del software. Aunque de esta forma se facilita la implantación de soluciones, su principal aportación se refiere a las mejoras en la productividad y calidad en todos los aspectos del desarrollo, abarcando todas las fases del ciclo de vida, permitiendo la aplicación real de las técnicas recomendadas por la metodología utilizada; algo que, de forma manual, sería muy difícil de llevarse a cabo debido a la gran cantidad de información, esquemas y documentación que se debe generar.

La evolución de las herramientas CASE ha sido considerable. Originalmente (principios de los 80) las herramientas de desarrollo software se dirigieron principalmente a la edición de diagramas y generación de informes. Posteriormente (mediados de los 80) se mejoraron para permitir la comprobación automática de diagramas de análisis y diseño y la gestión automática del repositorio. El paso siguiente (a partir de finales de los 80) fue la unión del diseño automático con la programación automática, apareciendo los generadores de aplicaciones (programación automática con lenguajes de cuarta generación) y los generadores de

código (en lenguaje de programación convencional a partir de las especificaciones de diseño)¹⁹.

Debido a una deficiente información por parte de las organizaciones de desarrollo respecto a las posibilidades de las herramientas CASE, desde su aparición se han producido muchas decepciones respecto a esta tecnología, al no satisfacer las expectativas de los adquirientes de las mismas. Para evitar estas situaciones, ISO ha publicado el estándar ISO 14102 (asumido por IEEE como norma IEEE 1462) para la evaluación y selección de herramientas CASE (ISO 1995b).

Las herramientas CASE más avanzadas incorporan funciones de reingeniería y trabajo en grupo. Las herramientas CASE con funciones de reingeniería permiten realizar un proceso de ingeniería inversa sobre sistemas compuestos por programas en código fuente (miles de líneas, por ejemplo, en COBOL, C o Java), creando una representación del programa a un nivel superior de abstracción, el nivel de diseño (de datos, arquitectónico y procedimental) y de análisis. Estas herramientas, además de recuperar la información de diseño del software, permiten su alteración y reconstrucción para mejorar la calidad global del sistema existente. Las herramientas CASE con funciones de trabajo en grupo facilitan la integración de diferentes grupos humanos, pertenecientes incluso a empresas diferentes, trabajando conjuntamente en un gran proyecto. Este tipo de herramientas deben incorporar facilidades clásicas de ofimática: correo electrónico, calendarios on-line, planificación de actividades, preparación de documentos, actas de reuniones, etc.

Otro problema que surge en relación con las herramientas CASE es la compartición de información entre herramientas de diferentes fabricantes, que, en muchos casos, hace imposible su integración y coordinación, no pudiendo, en general, una herramienta aprovechar los productos (modelos, diagramas, especificaciones, etc.) obtenidos con otra. Por este motivo se han desarrollado diferentes estándares internacionales (CDIF, PCTE, XMI) para facilitar este intercambio. En el caso de las herramientas CASE para desarrollo orientado a objetos basado en modelos de análisis y diseño realizados con la notación UML, prácticamente la totalidad de las herramientas existentes permiten la importación y exportación de modelos con el formato estándar XMI (*XML Metadata Interchange*) (OMG 2003b).

Independientemente del grado de automatización introducido en el desarrollo de software en cualquier organización, la mayoría de los productos generados durante un proyecto son documentos (en papel o electrónicos, listados de programas, manuales de usuario, etc.). En un entorno de desarrollo con enfoque de ingeniería del software (CASE), debería estar predefinido el estilo, contenido y

¹⁹Estas herramientas se conocen como *LowerCASE*, para diferenciarlas de la que sólo permiten el modelado, llamadas *UpperCASE*.

mecanismos de elaboración de todos los tipos de documentos que serán generados.

Normalmente las facilidades de documentación ofrecidas por las herramientas CASE tradicionales se limitan, por una parte, a la generación de informes que recogen los modelos gráficos (diagramas de análisis o de diseño) exigidos por la metodología aplicada y creados con la herramienta. Por otro lado, suelen ofrecer editores de texto propios o el acceso a procesadores externos para la creación de los documentos que deben ser producidos durante el proyecto, así como la descripción de los diferentes elementos que aparecen en los modelos gráficos (funciones, objetos, atributos, etc.). Los entornos CASE más evolucionados ofrecen además facilidades para el control de la documentación generada, proporcionando, entre otros, mecanismos de protección del contenido de los documentos para evitar accesos no autorizados, de almacenamiento y distribución de documentos, y de gestión de versiones (Mair 1996).

Desde el punto de vista metodológico, la evolución de la tecnología CASE desde las herramientas que permitían la preparación y mantenimiento de los modelos gráficos establecidos por las metodologías para cada fase del ciclo de vida de desarrollo, hasta los entornos CASE integrados que cubren completamente el ciclo de desarrollo, ha supuesto la aparición de lo que (Welsh & Han 1994) denominan "lagunas de integración y de seguridad", ya que, aunque estos sistemas consiguen la generación parcial del código fuente (fase de construcción) del software a partir de la documentación de especificaciones de diseño (fases de análisis y diseño, anteriores a la construcción), no han considerado la gestión de las necesarias relaciones existentes entre los documentos producidos en las sucesivas fases.

Entornos PSEE

En un entorno PSEE se puede modelar el desarrollo de software como tres tipos de actividades que pueden ser llevadas a cabo por grupos de personas diferentes. Estas tres actividades son: ingeniería del proceso, gestión del proyecto e ingeniería del software (Garg & Jazayeri 1996). La ingeniería del proceso permite definir y evaluar modelos del proceso de software; la actividad de gestión del proyecto selecciona un modelo de proceso producido por la ingeniería de procesos y lo adapta a un proyecto en particular y supervisa la ejecución del proceso. La actividad de ingeniería del software es la encargada de ejecutar el proceso especificado por la gestión de proyecto.

Estas herramientas PSEE se diferencian del enfoque CASE tradicional en que contemplan el proceso software como una entidad dinámica con su propio ciclo de vida, ofreciendo la posibilidad de adaptarlo en función del tipo de software que se vaya a construir y de supervisar el desarrollo para detectar las desviaciones

respecto del proceso establecido como modelo a seguir en ese caso.

Las principales características de estos entornos para la automatización del desarrollo de software son las siguientes:

- Permiten la definición de procesos mediante técnicas gráficas o formales de modelado.
- Ofrecen la posibilidad de analizar automáticamente los modelos de procesos para comprobar su consistencia, completitud y corrección de acuerdo con las normas de calidad preestablecidas. Por ejemplo, para detectar actividades redundantes, si se ha sobrepasado un número máximo de actividades concurrentes establecido o si se han incluido determinadas actividades de inspección y revisión.
- Se puede visualizar o imprimir la estructura de los procesos, con los flujos de actividades y productos implicados en su realización. Algunas PSEE también consideran la posibilidad de proporcionar diferentes puntos de vista de un mismo proceso, adaptándolo al tipo de usuario (gestor, analista, etc.) que lo está utilizando en cada momento.
- Además de la visualización, un PSEE debe servir de guía al personal de desarrollo, mostrando en todo momento las actividades siguientes que se deben realizar o los productos que se están desarrollando y el acceso a los mismos.
- En algunos PSEE también puede simularse la ejecución de los procesos modelados, para ayudar a decidir su idoneidad antes de adquirir los recursos necesarios para llevarlos a cabo.
- Cuando un proceso de software incluye actividades que podrían no requerir necesariamente la intervención humana en su ejecución, como la realización de pruebas del software o la notificación al personal pertinente de los cambios realizados en el software o en un documento; si tales actividades se han descrito ("programado") de una manera formal en un PSEE, utilizando un cierto lenguaje de modelado de procesos, podría ser ejecutadas automáticamente por el propio PSEE.
- Otro aspecto importante de un PSEE es su capacidad de supervisar la ejecución de un proceso y registrar información sobre las actividades realizadas que pueda utilizarse para posteriores mejoras del proceso.
- Un PSEE debe incluir las herramientas necesarias para el desarrollo de los proyectos software (de análisis, diseño, programación, pruebas, etc.) o el acceso a ellas.

- Puesto que en todo proyecto software están implicadas muchas personas, debe incorporar facilidades para el trabajo en grupo, garantizando la sincronización en el acceso concurrente a las actividades y productos gestionados por el PSEE.
- Finalmente una característica útil que se debe exigir a estos sistemas es la posibilidad de cambiar un modelo de proceso mientras ese proceso está siendo ejecutado.

Aunque ya se han creado entornos PSEE que funcionan y satisfacen muchos de los requisitos anteriores (Adele, SPADE, IPSE, ProcessWeaver, etc), todavía es necesario un esfuerzo investigador en este campo, y no sólo de expertos en ingeniería del software, sino también por parte de investigadores del ámbito de la psicología, que ayuden a aquellos a modelar el comportamiento no determinista del personal implicado en un proyecto; e ingenieros del conocimiento, si al final se pretende llegar al caso ideal aventurado ya en 1989 por McClure de herramientas "inteligentes" que realicen la mayoría de las tareas de forma automática y que, además, "enseñen" al ingeniero de software y "aprendan" de él, incrementando su habilidad para desarrollar software según aumenta su experiencia adquirida en proyectos anteriores (McClure 1992).

Factorías de software

Además de las herramientas CASE y los entornos PSEE, la automatización de la producción de software también se puede basar en la idea de que el desarrollo del software se asemeja a una cadena o línea de montaje especializada en la producción de un determinado tipo de productos software, por ejemplo, software para empresas aeronáuticas, para instituciones educativas, etc., reutilizando gran parte de los componentes software incluidos ("ensamblados") en las aplicaciones de una misma categoría. Esta idea ha dado lugar al concepto de Factoría de software, que puede definirse como "una línea de producción de software que proporciona facilidades para el desarrollo de una familia de productos mediante la configuración de herramientas extensibles utilizando pantallas basadas en un esquema de software" (Greenfield 2004). En la figura 1.15 se muestra un esquema del funcionamiento de este tipo de factorías para la producción de software.

Las factorías de software se basan en la reutilización sistemática del software, y en la existencia de entornos de desarrollo que se puedan configurar o adaptar al método de producción de software que sea más adecuado en cada tipo de proyecto. Herramientas como MS Visual Studio .Net o IBM WebSphere Studio permiten esta adaptación, y podrían, por tanto, dar soporte a una factoría de software.

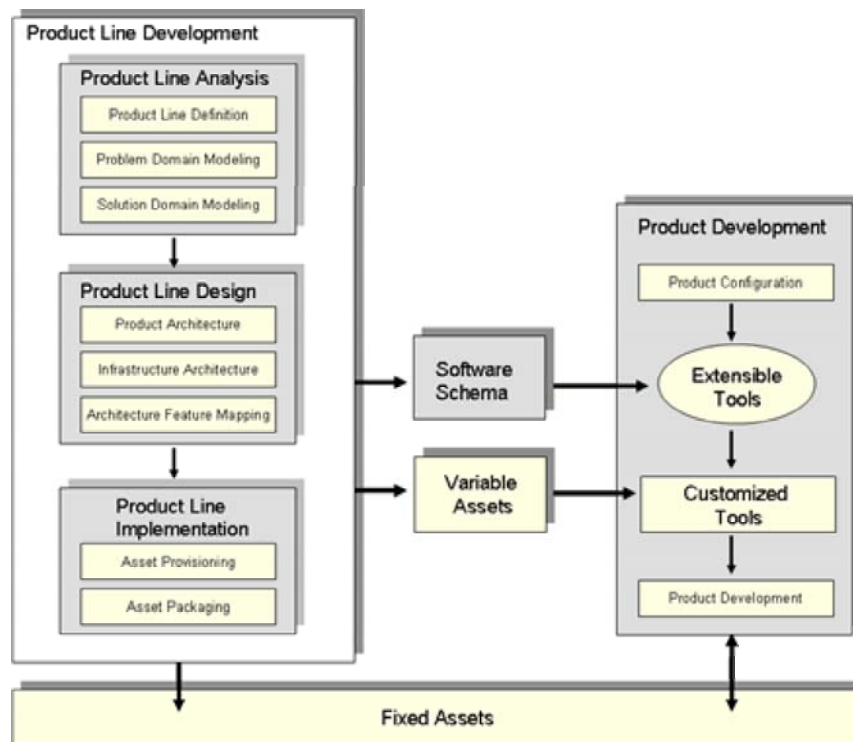


Figura 1.15: Factoría de software (Greenfield 2004)

Las factorías de software son una extensión del enfoque de desarrollo de software dirigido por modelos (MDD: *Model Driven Development*), en el que a través de modelos de diferentes niveles de abstracción, es posible generar de forma automática gran parte o la totalidad del código fuente de una aplicación (OMG 2003a). Por ello, las herramientas que dan soporte a una factoría de software trabajan a través de modelos, y permiten definir lenguajes de dominio específico (DSL: *Domain Specific Languages*) para modelar las necesidades particulares de los sistemas software de una familia o categoría determinada; por ejemplo, podría crearse un lenguaje de modelado visual orientado a aplicaciones bancarias, con elementos de modelado como "Tarjeta de Crédito", "Transacción", etc., y el entorno disponer de utilidades para transformar estos elementos en componentes software reutilizables.

10 Cuestiones de auto-evaluación

- 1 ¿Qué es un "proceso software definido"? ¿Qué relación tiene con el término "metodología de desarrollo de software"? **Respuesta:** Proceso (conjunto

de actividades, métodos, prácticas y transformaciones que se utilizan para desarrollar y mantener software y sus productos asociados) que cualquiera puede comunicar acerca de cómo avanzar en un proyecto de software. La definición del proceso (que ha de estar documentada) es la que guía a los ingenieros de software sobre la manera de trabajar. También se conoce como "proceso estándar". En cuanto a su relación con el término "metodología de desarrollo de software", pueden considerarse como sinónimos, si se considera que el "proceso de software definido" en una organización establece también las técnicas que hay que aplicar en cada una de las actividades en las que se descompone el proceso.

- 2 ¿Qué organismo creó CMMI (Capability Maturity Model Integration)? ¿Para qué?
Respuesta: Fue creado por el SEI (Software Engineering Institute), que pertenece a la Carnegie Mellon University (EEUU), para ayudar a las organizaciones a mejorar su proceso de producción de software
- 3 ¿Definen las normas CMMI e ISO-15504(SPICE) el mismo número de niveles de madurez para el proceso software? ¿Cómo se denominan los niveles en ambos casos?
Respuesta: 8. CMMI define 5 niveles: 1) Inicial, 2) Gestionado, 3) Definido, 4) Cuantitativamente Gestionado y 5) Optimizado. ISO-15504(SPICE) define 6 niveles: 0) Incompleto, 1) Realizado, 2) Gestionado, 3) Establecido, 4) Predecible, y 5) Optimizado. Los niveles 1 a 5 de ambas normas son equivalentes, si bien en SPICE también se considera un nivel 0 para identificar aquel proceso software que no puede llevarse a cabo por existir un fallo general de funcionamiento en la organización
- 4 ¿Qué nivel de madurez, según CMMI y según SPICE, tendría una empresa de software que realiza una mejora continua de su proceso software? **Respuesta:** Nivel 5 (Optimizado) según CMMI, y nivel 5 (Optimizado) según SPICE
- 5 ¿Qué relación puede establecerse entre CMMI, ISO-15504(SPICE) e ISO-9001? **Respuesta:** 11. CMMI y SPICE tienen el mismo objetivo, fijar el marco de trabajo que permita establecer y mejorar la madurez del proceso software, sirviendo de referencia para definir los procesos que se necesitan y cómo se deben implantar en las organizaciones que desarrollan software, y para determinar la capacidad de los procesos que están utilizando las organizaciones y los aspectos que deben mejorar. La norma ISO-9001 especifica los requisitos para un sistema de control de calidad en las empresas de producción, entre ellas la de desarrollo de software. Esta norma promueve un enfoque orientado a procesos, como CMMI y SPICE, por lo que

gran parte de los diferentes requisitos establecidos, pueden satisfacerse a través de los procesos definidos por CMMI y SPICE.

- 6 ¿Puede afirmarse que una organización que ha obtenido el certificado ISO-9001 se encuentra en el nivel 3 (Definido) de CMMI? Y al revés? ¿Puede afirmarse que una organización de nivel 3 cumple con ISO-9001? **Respuesta:** 12. No, porque podría encontrarse en el nivel 2, ya que la norma ISO es menos específica en el caso de la producción de software que CMMI, ya que se aplica en otros ámbitos de la industria. Una organización de nivel 3 podría conseguir fácilmente el certificado ISO-9001, pero debería aplicar esta norma de calidad también en el resto de la empresa (dpto. de recursos humanos, marketing, etc.), además de hacerlo en la producción.
- 7 ¿Qué es la capacidad de un proceso software? ¿Es lo mismo que el rendimiento del proceso, o la madurez del proceso? **Respuesta:** La capacidad de un proceso software es el rango de resultados esperado, que pueden ser conseguidos siguiendo el proceso software. El rendimiento hace referencia a los resultados actuales que se han conseguido siguiendo el proceso software. La madurez es el alcance con el que el proceso específico ha sido explícitamente definido, gestionado, medido, controlado y efectivo. La madurez indica la riqueza del proceso y la consistencia con la cual se aplica.
- 8 ¿Cuántos procesos para el ciclo de vida del software define la norma ISO 12207? **Respuesta:** 17.

11 Notas bibliográficas

Es muy recomendable la consulta de los estándares tratados en este documento, especialmente ISO 12207, ISO 14504, ISO 9001 (www.iso.ch), IEEE 1074 (www.ieee.org) y CMMI, este último publicado íntegramente en la Web del *Software Engineering Institute* (www.sei.cmu.edu/cmmi). La metodología de desarrollo Métrica Versión 3 también se encuentra disponible en Internet, en la Web del *Consejo Superior de Informática* (www.csi.map.es). Pueden encontrarse documentos sobre otras metodologías utilizando los buscadores de Internet, recomendándose especialmente las descripciones incluidas en la enciclopedia on-line *Wikipedia* (en.wikipedia.org). En relación con las metodologías para Ingeniería Web, el artículo (Escalona, Mejías & Torres 2002) realiza un completo recorrido por las metodologías más conocidas. Sobre las metodologías ágiles, hay interesantes documentos, incluido el "manifiesto ágil", en la dirección www.programacionextrema.org.

Respecto al modelado y automatización del proceso software, es muy recomendable la lectura del número monográfico dedicado a este tema de la revista Novática (ATI 2004). El estándar SPEM para el modelado de procesos está disponible en la Web del Object Management Group (www.omg.org). En la dirección www.objectbydesign.com se publica y actualiza periódicamente una lista con las herramientas CASE (incluidas las gratuitas) existentes que permiten el modelado con UML. También se recomienda la lectura de los artículos que publica Microsoft en su Web (msdn.microsoft.com) sobre factorías de software. Por último, es muy recomendable la lectura del capítulo 9 de SWEBOK, que especifica los contenidos que esta guía propone en relación con el área de conocimiento *Software Engineering Process* (IEEE 2004).

12 Ejercicios y actividades propuestas

12.1 Actividades propuestas

- 1 Analizar los documentos en los que se describen el estándar ISO 12207 y Métrica Versión 3 ¿Con qué proceso definido por ISO 12207 se corresponde más fielmente cada uno de los procesos definidos por Métrica Versión 3?.
- 2 Realizar un estudio de compatibilidad entre los procesos del ciclo de vida del software propuestos por ISO 12207 y los de IEEE 1074.
- 3 Leer el artículo (Henderson-Sellers et al. 2001) en el que se propone una ampliación de la metodología OPEN para desarrollo Web y proponer una ampliación similar de Métrica para convertirla en metodología de Ingeniería Web
- 4 Localizar algún entorno de desarrollo que se acerque a la idea de factoría software, cumpliendo algunos e los requisitos exigidos.
- 5 Modelar el proceso de Análisis de la metodología Métrica utilizando un lenguaje de modelado de procesos.
- 6 Razonar si el Proceso Unificado puede considerarse una metodología ágil (se recomienda consultar el artículo (Fowler 2003)).

Bibliografía

- ATI (2004), 'Tecnología de proceso software (monografía)', *NOVATICA* (171), 3–47.
- Booch, G., Rumbaugh, J. & Jacobson, I. (1999), *El lenguaje unificado de modelado*, 1999 edn, Addison Wesley.
- Brown, A. & Wallnau, K. (1996), Engineering of component based systems, *in* 'Component-Based Software Engineering', pp. 7–15.
- Crosby, P. (1979), *Quality is Free*, 1979 edn, McGraw-Hill.
- Deming, W. (1986), *Out of the Crisis*, 1986 edn, MIT Center for Advanced Engineering Study.
- DRAE (2001), *Diccionario de la Lengua Española*, 2001 edn, Real Academia Española.
- Escalona, M., Mejías, M. & Torres, J. (2002), 'Metodologías de desarrollo de sistemas de información en la web y análisis comparativo', *NOVATICA* (159), 49–59.
- Fowler, M. (2003), 'The new methodology'.
- Garg, P. & Jazayeri, M. (1996), Process-centered software engineering environments: A grand tour, *in* 'Software Process (A. Fuggeta, A. Wolf eds.)', John Wiley and Sons', pp. 25–52.
- Greenfield, J. (2004), 'Software factories: Assembling applications with patterns, models, frameworks and tools. <http://msdn.microsoft.com>'.
- Hammer, M. (1990), 'Reengineering work: Don't automate, obliterate', *Harvard Business Review* (Jul./Aug.), 104–112.
- Henderson-Sellers, B., Haire, B. & Lowe, D. (2001), 'Adding web support to open', *Journal of Object Oriented Programming* **August/September**, 34–38.

- Humphrey, W. (1987), 'Characterizing the software process: a maturity framework. sei technical report cmu/sei-87-tr-11'.
- Humphrey, W. (1995), *A Discipline for Software Engineering*, Addison-Wesley.
- IEEE (1990), 'IEEE standard glossary of software engineering terminology'.
- IEEE (1997), 'IEEE 1074-1997, ieee standard for developing software life cycle processes'.
- IEEE (1998a), 'IEEE/EIA 12207.0-1996 industry implementation of international standard iso/iec 12207:1995 standard for information technology — software life cycle processes'.
- IEEE (1998b), 'IEEE/EIA 12207.1-1996 industry implementation of international standard iso/iec 12207:1995 standard for information technology — software life cycle processes - life cycle data'.
- IEEE (1998c), 'IEEE/EIA 12207.2-1996 industry implementation of international standard iso/iec 12207:1995 standard for information technology — software life cycle processes - implementation considerations'.
- IEEE (2000), 'IEEE recommended practice for architectural description of software-intensive systems'.
- IEEE (2004), *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 2004 edn, IEEE.
- ISO (1995a), 'ISO/IEC 12207, software life-cycle processes'.
- ISO (1995b), 'ISO/IEC 14102, guideline for the evaluation and selection of case tools'.
- ISO (2000), 'ISO 9001:2000, quality management systems - requirements'.
- ISO (2004a), 'ISO/IEC 15504, information technology-process assesment'.
- ISO (2004b), 'ISO/IEC 9000-3:2004, software engineering - guidelines for the application of iso 9001:2000 to computer software'.
- Jacobson, I., Booch, G. & Rumbaugh, J. (2000), *El proceso unificado de desarrollo de software*, 2000 edn, Addison Wesley.
- Maddison, R. (1983), *Information System Methodologies*, 1983 edn, Wiley-Heyden.

- Mair, P. (1996), 'Case report'.
- MAP (1995), *Especificaciones para el Tratamiento de Flujos Automatizados (ESTROFA)*, 1995 edn, Ministerio para las Administraciones Públicas.
- MAP (2001), *Metodología de Planificación y Desarrollo de Sistemas de Información. MÉTRICA Versión 3*, 2001 edn, Ministerio para las Administraciones Públicas.
- McClure, C. (1992), *CASE is Software Automation*, 1992 edn, Prentice Hall.
- OMG (2003a), 'Model driven architecture (mda)'.
- OMG (2003b), 'Xml metadata interchange (xmi)'.
- OMG (2005), 'Software process engineering metamodel (spem)'.
- Paulk, M., Curtis, B., Chrissis, M. & Weber, C. (1987), 'Capability maturity model for software, version 1.1. sei technical report cmu/sei-93-tr-24'.
- Ruiz, F., Calero, C. & Piattini, M. (2005), *Ontologies for software engineering and technology*, 2005 edn, Springer.
- SEI (2002a), 'Capability maturity model integration (cmmi)'.
- SEI (2002b), 'Software engineering capability maturity model integration (cmmi-sw)'.
- Tardieu, H., Rochfeld, A., Colletti, R., Panet, G. & Vahée, G. (1985), *La Méthode Merise*, 1985 edn, Les Editions d'Organisation, Paris.
- Thayer, R. (2002), 'Software system engineering: A tutorial', *IEEE Computer* **35**(4), 68–73.
- Welsh, J. & Han, J. (1994), 'Software documents: Concepts and tools', *Software: Concepts and Tools* **15**(1), 12–25.