TEMA 12

Calidad y mejora del proceso *software*

Introducción

→Definición proceso software.
→Gestión efectiva del proceso: definir, medir, controlar y mejorar.
→Plan de calidad.

Medición del *software*

→Ontología para la medición del software.
→Atributos.
→Entidades.
→Métricas
→Tipo de escala.
→Estándares y *frameworks* de medición.

Pruebas de *software*

→Definición prueba del software.
→Modelo en V.
→Pruebas de regresión.
→Pruebas de humo.
→Cualidades de un ingeniero de pruebas.

Modelo de calidad de procesos software: CMMI

→Modelo de procesos.
→Niveles de madurez.
→Áreas de proceso.
→Medición en CMMI.

12.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave,** además de **los apartados 3.1 y 3.2 del capítulo 3 (páginas 47-52), apartado 3.4 del capítulo 3 (páginas 57-59) y apartado 4.2 del capítulo 4 (páginas 69-71)** disponible en el aula virtual bajo licencia CEDRO, del libro:
Piattini, M.G., García, F.O., Garzás, J. y Genero, M.G. (2008). *Medición y estimación del software. Técnicas y métodos para mejorar la calidad y la productividad*. España: RaMa.

En este tema se estudian aspectos relevantes asociados a la calidad y mejora del proceso *software* desde un punto de vista global.

12.2. Introducción

Cuando se empezó a desarrollar *software*, en los primeros años de la informática, los sistemas que se construían eran bastante sencillos, por lo que se asumía que las aplicaciones tenían errores y no se le daba mayor importancia. El foco de atención en aquella época estaba en el *hardware*. Ahí, sí que era indispensable que funcionara bien y no tuviera fallos.

Sin embargo, como ya se ha estudiado al inicio de la asignatura, hubo un momento en lo que se denominó «la crisis del *software*» en el que la pésima calidad de las aplicaciones y el elevado número de errores sí que empezaron a preocupar. Así fue cómo surgió la **ingeniería de *software***. Desde entonces, no ha cesado el estudio y la investigación en el campo de la ingeniería de *software* con objetivo primordial de mejorar y tratar de garantizar la calidad de las aplicaciones desarrolladas.

En este sentido, el **proceso de *software*** ligado al **desarrollo y evolución de los productos *software***, así como a la **gestión de los proyectos *software*** resulta fundamental para la obtención de aplicaciones de calidad. Tal y como se recoge en Piattini y Garzás (2010, 169), **el proceso *software*** se puede considerar como «un

campo de estudio amplio y complejo en el mundo de la ingeniería de *software* en el que hay una gran cantidad y diversidad de elementos a abordar».

Con más detalle, Fuggeta (2000), define el proceso *software* como «conjunto coherente de políticas, estructuras organizativas, tecnologías, procedimientos y artefactos que son necesarios para idear, desarrollar, desplegar y mantener un producto *software*». Los requisitos fundamentales en **el estudio de la calidad del proceso *softwar*e** son (Piattini y Garzás, 2010, 169-170):

- Los resultados obtenidos cumplen los requisitos.
- El proyecto *software* se ha definido correctamente.
- Los resultados obtenidos se pueden mejorar en función de los objetivos de negocio de la organización, que seguramente cambiarán con demasiada frecuencia debido a la competitividad actual que se da en la industria del *software*.

Piattini y Garzás (2010) identifican cuatro actividades que se han de llevar a cabo para una **gestión efectiva del proceso *software*** (ver Figura 1):

- Definir el proceso.
- Medir el proceso.
- Controlar el proceso.
- Mejorar el proceso.

Figura 1. Actividades a realizar en la gestión del proceso software (Adaptado de Piattini y Garzás, 2010).

La adopción de estas actividades para la gestión del proceso *software* implica tener en cuenta los siguientes aspectos (Piattini y Garzás, 2010):

- **Definición del proceso.** La definición del proceso sería la primera responsabilidad que hay que asumir en una organización para hacer una gestión efectiva del mismo. Para su definición, se hará un modelo del proceso que mostrará todos los elementos que intervienen en dicho proceso.

- **Ejecución y control del proceso.** Una vez modelado el proceso *software*, el modelo generado se tomará de referencia en todo proyecto que realice la organización. En consecuencia, habrá que controlar la ejecución de todos los proyectos de la organización, que están siguiendo dicho modelo, para asegurarse de que se están cumpliendo los objetivos definidos para cada proyecto.

- **Medición y mejora.** Medir y mejorar son dos aspectos que están también claramente interrelacionados. Para poder mejorar el proceso *software* de una organización será necesario medir previamente el proceso y determinar así las partes que son susceptibles de mejora. Los resultados obtenidos de las mediciones proporcionan una información objetiva y valiosa para la organización que favorecerá la planificación, identificación y realización de las acciones necesarias para la mejora del proceso *software*.

Además, todo proceso *software* debería contar con la definición de un **plan de calidad** que estará basado en los siguientes pasos (Humphrey, 1990, 344):

1. Se debe establecer un conjunto de objetivos de calidad de forma cuantitativa, es decir, métricas. Sin indicadores numéricos que midan el estado del proyecto, el esfuerzo que se pueda poner para mejorar los procesos no podrá tener un impacto duradero.

2. Las métricas de calidad utilizadas para el proyecto han de proporcionar información lo más objetiva posible, para que no dependa de la interpretación de quien las analice.

3. Las métricas utilizadas se deben definir y documentar de una manera precisa de tal forma se pueda disponer de herramientas informáticas que las puedan calcular y procesar.

4. El plan de calidad se debe definir al inicio del proyecto y se podrá actualizar ante cambios importantes o hitos determinados.

5. El plan de calidad se debe revisar para comprobar que sigue siendo conforme a los objetivos de calidad que se intentan alcanzar y se modificará en caso de no conformidad.

6. Se hará un seguimiento del rendimiento de la calidad. En caso de incumplir el plan de calidad, se deberán realizar las acciones correctivas oportunas.

7. Mientras no existan métricas concretas para representar proyectos complejos, todas las métricas de calidad que se utilicen se tratarán como indicadores del rendimiento global del proyecto.

Por su parte, David Garvin propone adoptar un punto de vista multidimensional para evaluar la calidad, que podría ser aplicable a los objetivos del proceso *software* de la siguiente manera (Pressman, 2010, 341):

» **Calidad del desempeño.** ¿El *software* entrega todo el contenido, funcionalidades y características especificadas en el modelo de requisitos, tal y como espera el cliente? »
Calidad de las características. ¿El *software* presenta características que son del interés y el agrado del cliente?
» **Fiabilidad.** ¿Cuántos defectos presenta el *software*?
» **Conformidad.** ¿El *software* es conforme a todas las normas y estándares establecidos para su implementación?
» **Durabilidad.** ¿El *software* es fácil de mantener y admite cambios de una manera flexible y sin efectos colaterales?
» **Servicio.** ¿Los cambios en el *software* se realizan en un periodo razonable de tiempo?
» **Estética.** ¿El cliente está satisfecho con la interfaz de usuario que presenta el sistema?
» **Percepción.** ¿Cuál es la percepción inicial que tiene el cliente hacia el *software* implementado?

12.3. Medición del *software*

Un aspecto muy relevante a tener en cuenta a la hora de analizar la calidad y mejora del proceso *software* es la medición. La medición en la **ingeniería de *software*** es una disciplina bastante reciente, ya que al contrario que en otras ingenierías, equivocadamente, en los comienzos de la ingeniería de *software* las mediciones no se consideraban necesarias, hasta que se vio su relación directa con la calidad y la mejora continua del proceso *software*: «Si no se puede medir, no se puede controlar. Si no se puede controlar, no se puede mejorar».

Como esquema general y válido, la Figura 2 muestra la Ontología propuesta por García et al. (2006) para la medición del *software*.

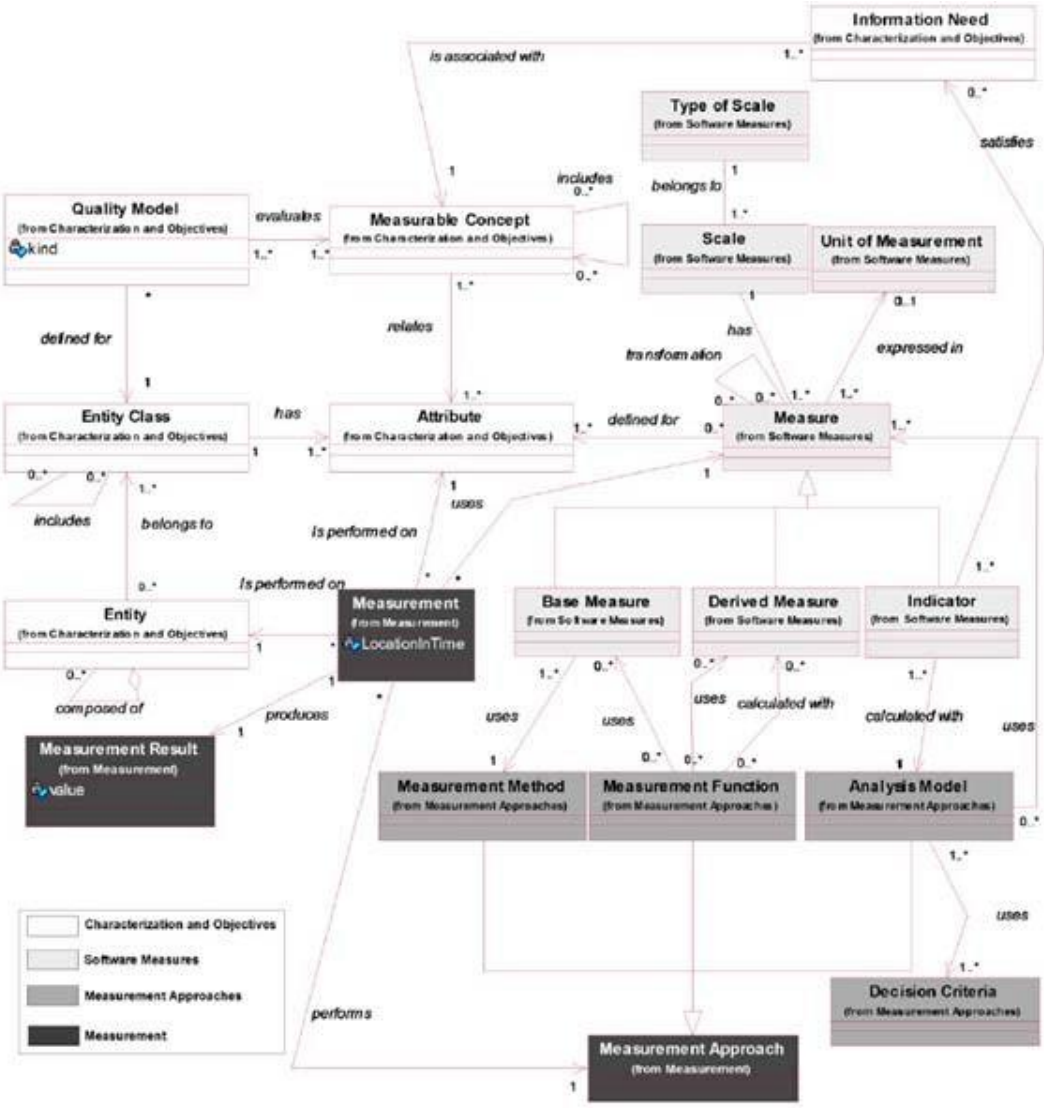


Figura 2. Diagrama de clases UML que ilustra la Ontología para la medición del *software* (García et al., 2005).

De acuerdo a la Figura 2, en toda medición del *software*, los **atributos** se corresponderán con las propiedades que comparten todas las entidades de un tipo (clase) de entidad; las **entidades** representarán los objetos que van a ser objeto de medición por medio de sus atributos; y las **métricas** constituirán la **forma de medir** (método de medición, función de medición o modelo de análisis), así como la **escala** (conjunto de valores con propiedades definidas, como por ejemplo, los valores 1, 2, 3, 4 y 5 para medir el nivel de madurez CMMI, o un número natural [ratio] para medir el *software* en líneas de código.) que se utilice para dicha medición.

De esta manera, por ejemplo, se puede definir la métrica «**líneas de código**» (unidad de medición) para realizar mediciones en base al **«tamaño»** (escala) de una clase en Java (entidad). El **tipo de escala** es el que va a identificar el tipo de relación que se da entre los valores válidos en la escala (Piattini, García, Garzás y Genero, 2008). Piattini, García, Garzás y Genero (2008, 5-6) clasifican las escalas en cinco tipos diferentes:

- **Escala nominal.** Esta escala clasifica las entidades en base a la asignación de un nombre para cada atributo. Así, por ejemplo, se puede distinguir de forma nominal por su dorsal a los jugadores de un equipo de baloncesto, donde el jugador con el dorsal «30» no quiere decir que sea el doble del jugador con el dorsal «15». Simplemente es un valor identificativo

- **Escala ordinal.** Esta escala ordena los atributos en rangos, pero la distancia entre los rangos no es significativa. Así, si se valora un curso de 0 a 4, la distancia entre los valores 0 y 1 no tiene porqué ser la misma que la distancia entre 2 y 3.

- **Escala de intervalo**. Esta escala ordena también los atributos en rango, pero en este caso, la distancia sí que es significativa, como por ejemplo, la temperatura en ºC.

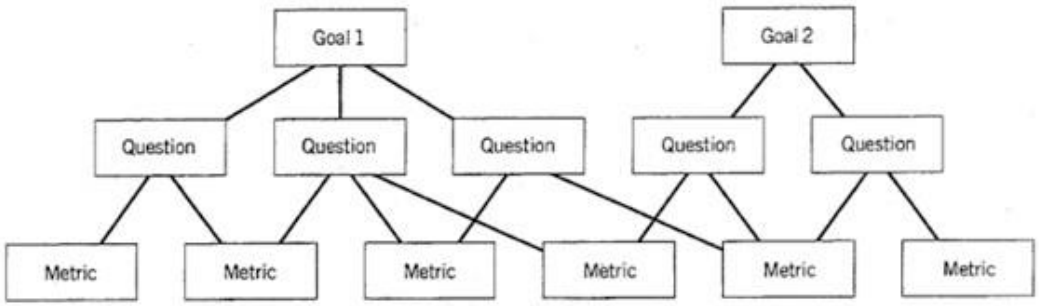
- **Escala de ratio.** Esta escala resulta muy útil ya que respeta el orden, tamaño de los intervalos y los ratios entre las entidades. Además, contiene el punto fijo de referencia 0, es decir, la escala siempre ha de partir del valor 0 y ha de incrementarse en pasos con la misma distancia. Sobre este tipo de escala se podrán realizar operaciones matemáticas como por ejemplo, suma, resta, multiplicación y división. Un ejemplo de tipo de escala de ratio sería el peso de una entidad.

- **Escala absoluta.** Este tipo de escala es la más restrictiva y se utilizará cuando la opción de contar (del inglés *count*), es decir, el número de elementos, sea la única

forma posible de medir el atributo de una entidad. Ejemplos de tipos de escala absoluta serían el número de defectos encontrados en un componente *software* y el número de personas que participan en un proyecto.

La **unidad de medición**, de acuerdo con la Ontología mostrada en la Figura 2, se corresponderá con una cantidad determinada, definida y acordada, con la que se podrán comparar otras cantidades que tengan asociada la misma unidad de medición. Ejemplos de unidades de medición serían «líneas de código», «páginas», «personas-mes» y «número de clases».

Por tanto, de acuerdo a esta propuesta, la medición se puede ver como «el conjunto de operaciones que permite obtener el valor del resultado de la medición para un atributo de una entidad, usando una forma de medir» (Piattini, García, Garzás y Genero, 2008, 7). Entre los estándares y *frameworks* que ofrecen soporte al proceso de medición se encuentran los estándares ISO 15939 e IEEE Std 1061-1998 , y los *frameworks Goal Question Metric* (GQM) (ver Figura 3) y *Practical Software Measurement* (PSM) (ver Figura 4).



Goal	Purpose	Improve
	Issue	the timeliness of
	Object (process)	change request processing
	Viewpoint	from the project manager's viewpoint
Question	What is the current change request processing speed?	
Metrics	Average cycle time	
	Standard deviation	
	% cases outside of the upper limit	
Question	Is the performance of the process improving?	
Metrics	$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$	
	Subjective rating of manager's satisfaction	

Figura 3. Estructura jerárquica de la fase de definición del programa de medición perteneciente al modelo GQM (Basili, Caldiera y Rombach, 1994).

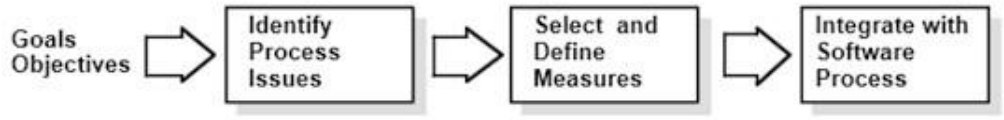


Figura 4. Planificación del programa de medición de acuerdo a PSM (Florac, Park y Carleton, 1997).

12.4. Pruebas del *software*

Las **pruebas del *software*** (del inglés *test cases*) constituyen uno de los procesos más complejos y costosos encaminados a la mejora de la calidad en el desarrollo y evolución del *software*. Sin embargo, si se mejora el proceso de prueba se podrá aumentar la calidad del producto *software* de manera significativa, así como reducir costes, por lo que realmente merece la pena el esfuerzo.

La definición clásica y más estandarizada es la proporcionada por Myers (2004) que describe una **prueba del *software*** como «el proceso de ejecución de un sistema *software* con la intención de **encontrar errores**». De manera un poco más especializada, IEEE (2008) define **prueba del *software*** como el proceso que determina si **la salida que genera una actividad concreta es conforme a los requisitos** establecidos para dicha actividad, **y si el producto *software*, por tanto, proporciona valor al cliente**. Las pruebas del *software* determinan si una actividad es conforme a los requisitos asignados a través de, por ejemplo, su análisis, demostración e inspección.

De manera más amplia, las pruebas del *software* se definen como un proceso compuesto de todas las actividades, estáticas (por ejemplo, revisión o análisis estático de los requisitos, diseño o código, sin ejecución de los artefactos) y dinámicas (las que implican ejecución del *software*), que forman parte del ciclo de vida de los productos *software*, como serían la planificación, preparación y evaluación, implementadas con un determinado objetivo y que permitirán determinar si se satisfacen los requisitos y está libre de defectos (ISTQB, 2015).

Esta definición lo que también viene a decir es que la realización de pruebas no es exclusividad del programador del sistema, sino que «debe ser un proceso completamente integrado en la organización y en la planificación», que debe llevarse a cabo de una

manera sistemática, con ayuda de los recursos, técnicas y herramientas más adecuadas en cada caso (Piattini y Garzás, 2010).

El **modelo en V** es uno de los modelos más utilizados en el proceso *software* como la forma de representar la vinculación de las pruebas del *software* a las distintas fases del ciclo de vida del proyecto. La Figura 5 ilustra el modelo en V con los dos niveles de proceso: **desarrollo y pruebas**.

Figura 5. Modelo en V (Adaptado de Piattini y Garzás, 2010).

En la Figura 5, las flechas sólidas indican secuencialidad temporal, mientras que las flechas huecas representan actividades de **planificación y preparación de casos de prueba para el sistema**. Así, una vez que estén implementados los componentes, se podrán ejecutar los casos de prueba que permitirán su validación. Por otro lado, las **pruebas de integración y sistema** permitirán validar el diseño del sistema y su ejecución en el entorno de operación.

Por último, las pruebas de **aceptación** permitirán validar los requisitos y las necesidades del cliente (Piattini y Garzás, 2010). A pesar de su aparente secuencialidad, la planificación y preparación de las pruebas se irá elaborando en paralelo con las distintas fases del ciclo de vida del proyecto, y para cada uno de los niveles de prueba (**componentes, integración y sistema, y aceptación**), con el objetivo de llevarlas a cabo lo más pronto posible dentro del proceso de desarrollo de *software*. Con la aplicación del modelo en V lo que se busca es:

- Minimizar los riesgos del proyecto.
- Favorecer la comunicación entre los *stakeholders* del proyecto.
- Reducir los costes del proyecto.
- Mejorar y asegurar una calidad aceptable de los productos *software* que se construyen.

Otro tipo de pruebas también muy comunes en el desarrollo de productos *software*, que se suelen automatizar y se realizan con relativa frecuencia, son las **pruebas de regresión** y las **pruebas de humo** (del inglés *smoke test*). Las **pruebas de regresión** se basan en la ejecución de pruebas que ya se habían realizado anteriormente con el objetivo de detectar posibles efectos colaterales que hayan podido surgir por la resolución de problemas que habían sido identificados, modificaciones realizadas sobre el sistema o incorporación de nuevas funcionalidades. Las **pruebas de humo** son las pruebas que representan un subconjunto de las pruebas totales diseñadas para el proyecto y que cubren de una manera superficial los componentes más importantes y críticos del sistema (Piattini y Garzás, 2010).

Por último, para que las pruebas del *software* se realicen de la manera más óptima posible, los ingenieros de pruebas (del inglés *testers*), responsables de llevarlas a cabo, han de presentar unas cualidades especiales (Piattini y Garzás, 2010):

- Atención en los detalles y asumir el desafío de detectar defectos en el *software*.
- Creatividad, curiosidad e intuición para dar con situaciones que pueden dar problemas.
- Capacidad para trabajar bajo presión ya que normalmente sus intereses entran en conflicto con los intereses de los programadores que han implementado la parte afectada.
- Capacidad comunicativa para poder expresar y explicar a los programadores de manera clara, precisa y lo más sencilla posible los fallos encontrados en el *software*.
- Cuando las circunstancias así lo requieran, conocimiento más profundo de las funcionalidades del producto *software* para que las pruebas que realice no se limiten a comprobar simplemente que el sistema no falla.

12.5. Modelo de calidad de procesos *software*: CMMI

Otro aspecto muy relevante en lo que respecta a la calidad del proceso *software* es el constituido por **los modelos de procesos y modelos de evaluación** que **juntos constituyen** lo que se denomina **modelos de mejora de procesos**, ya que favorecen

la identificación, integración, medición y optimización de las buenas prácticas existentes para el desarrollo y evolución del *software*.

Los modelos de evaluación evalúan y valoran la calidad de los procesos que están sujetos a su estudio o análisis, por lo que a partir de los resultados obtenidos de estos modelos se podrá establecer una estrategia en la organización para mejorar los procesos afectados (Piattini y Garzás, 2010).

Capability Madurity Model Integration (CMMI) es un modelo de procesos que se creó en el *Software Engineering Institute* (SEI) de la Universidad Carnegie Mellon para establecer un sistema de evaluación de las organizaciones que suministraban *software* al gobierno de los Estados Unidos.

CMMI constituye un modelo de calidad con un enfoque orientado a la mejora de procesos que proporciona los elementos necesarios para llevarla a cabo de una manera efectiva, ayuda a establecer prioridades y objetivos en dicha mejora, proporciona guías para los procesos de calidad y sirve como punto de referencia para la evaluación de los procesos actuales.

Sin embargo, resulta importante destacar que CMMI no es ni un proceso ni una metodología, ya que no indica la secuencia de ejecución que ha de seguirse. CMMI contiene un conjunto de prácticas que van a describir las características que han de presentar los procesos para que sean efectivos y que por tanto se deberán tener en cuenta a la hora de gestionar, implementar y mantener productos *software* (Garzás y Piattini, 2010).

CMMI está basado en un modelo de madurez, donde el objetivo final de CMMI es el de mejorar la usabilidad de dicho modelo de madurez en el dominio de ingeniería de *software* y de otras disciplinas mediante la integración de varios modelos en un único *framework*. Además, normalmente, cuando alguien habla de CMMI se suele referir realmente a los modelos de procesos para desarrollo de *software*, CMMI-DEV, que es el que resulta de interés para esta asignatura.
El modelo de madurez en el que está basado CMMI-DEV considera 5 niveles que clasifican el nivel de implementación que tienen los procesos pertenecientes a una organización. Cada nivel de madurez define, por un lado, la escala de medida de la capacidad de los procesos de la organización, y por otro lado, los objetivos que se han de marcar en la organización de cara a focalizar sus esfuerzos en la mejora de sus procesos:

- **Nivel 1. Inicial.** En este nivel los procesos de la organización son generalmente caóticos, no repetibles, donde gran parte del trabajo se realiza sin procedimientos preestablecidos y *ad hoc* para cada proyecto al que se enfrentan.
- **Nivel 2. Gestionado.** En este nivel los procesos de la organización tienen establecidas las actividades de gestión de proyectos, por lo que los procesos en este caso sí que podrán ser repetibles (aunque por falta de rigurosidad puede que a lo mejor no para todos los proyectos de la organización) con resultados consistentes.
- **Nivel 3. Definido.** En este nivel los procesos de desarrollo y mantenimiento de *software* están documentados y estandarizados y se encuentran dentro del ciclo de mejora continua de la organización. A diferencia del nivel anterior, en este caso, todos los proyectos de la organización se llevan a cabo de acuerdo a un conjunto de procedimientos establecidos.
- **Nivel 4. Gestionado cuantitativamente.** En este nivel los procesos de la organización tienen asociado un programa detallado y organizado de medición de procesos de desarrollo de *software*. Además, todos los procedimientos de gestión establecidos en la organización que se utilizan para ajustar y adaptar los procesos según el proyecto están documentados y tendrán un carácter público.
- **Nivel 5. Optimizado.** En este nivel los procesos de la organización forman parte a su vez de un proceso de mejora continua. En dicho proceso, se recopilan todos los datos vinculados a cada proyecto de la organización, que se estudiarán y analizarán de cara a poder mejorar e innovar los procesos de la organización. En el nivel 5, el más alto, las organizaciones, por tanto, usan procesos definidos y repetibles, aplican métricas para la mejora continua de sus procesos y, en definitiva, se encuentran en una búsqueda continua de innovación y de poder hacer las cosas mejor.

Para que una organización se certifique en un nivel de madurez concreto tiene que tener implementadas correctamente las áreas de proceso que CMMI-DEV define para ese nivel. CMMI-DEV v. 1.3 establece 22 áreas de proceso. Cada una de estas áreas de proceso representa un conjunto de buenas prácticas relacionadas que, cuando se implementan conjuntamente, satisfacen objetivos importantes que permiten obtener mejoras significativas en el área correspondiente.

Las áreas de proceso se agrupan por nivel de madurez, indicando qué áreas de proceso hay que implementar para alcanzar ese nivel de madurez. Así, por ejemplo, si una organización quiere alcanzar el nivel de madurez 4, deberá fijarse en las áreas de proceso marcadas como nivel 4 y alcanzar los objetivos establecidos para cada una de dichas áreas.

Una vez que una organización ha certificado en el nivel 4, entonces se certifica que dicha organización no utiliza solamente las áreas de proceso de nivel 4 a los niveles adecuados de madurez, sino también todas las que se encuentran en los niveles inferiores que sería en este caso todas las de los niveles 2 y 3 (no se incluye el nivel 1 ya que este nivel no tiene áreas clave de proceso establecidas). La Tabla 1 resume las áreas de proceso que recoge cada nivel de madurez.

Tabla 1: áreas de proceso por nivel de madurez CMMI-DEV v. 1.3 (Garzás, Irrazábal e Santa, 2011).

Nivel de madurez	Área de proceso
2	Gestión de requisitos (REQM) Planificación del proyecto (PP) Monitorización y control del proyecto (PMC) Gestión de acuerdos con proveedores (SAM) Gestión de configuración (CM) Aseguramiento de la calidad del proceso y producto (PPQA) Medición y Análisis (MA)
3	Desarrollo de requisitos (RD) Solución técnica (TS) Integración del producto (PI) Verificación (VER) Validación (VAL) Definición de los procesos de la organización (OPD) Enfoque de los procesos de la organización (OPF) Formación organizativa (OT) Gestión integrada del proyecto (IPM) Gestión de riesgos (RSKM) Análisis de decisiones y resolución (DAR)
4	Gestión cuantitativa del proyecto (QPM) Rendimientos de los procesos de la organización (OPP)
5	Análisis causal y resolución (CAR) Gestión del rendimiento de la organización (OPM)

Además de certificar el nivel en el que se encuentra una organización o área de procesos, lo cual es muy interesante para una organización por lo que implica, lo que CMMI también aporta a las organizaciones es el prestigio de ser un entidad certificada CMMI, lo que le permitirá optar a contratos en cuyos pliegos de condiciones se solicite como requisito de concurso la demostración del nivel en el que la organización se ha certificado.

Por último, mencionar que en CMMI también se da especial importancia a la medición en la madurez de los procesos. Es por ello que incorpora, como se puede observar en la

Tabla 1, un área específica denominada «*Medición y Análisis*». Esta área proporciona el enfoque y la visibilidad que las organizaciones necesitan para poder realizar mediciones en sus procesos e implementar acciones de mejora.

En la Figura 6 se ilustran los principales procesos vinculados al área de «*Medición y Análisis*» de CMMI (Piattini, García, Garzás y Genero, 2008). El primer paso sería entonces identificar los objetivos de la medición para poder implementar posteriormente el proceso de medición y análisis, que requerirá a su vez la integración del programa de medición en los procesos de trabajo que forman parte de la organización.

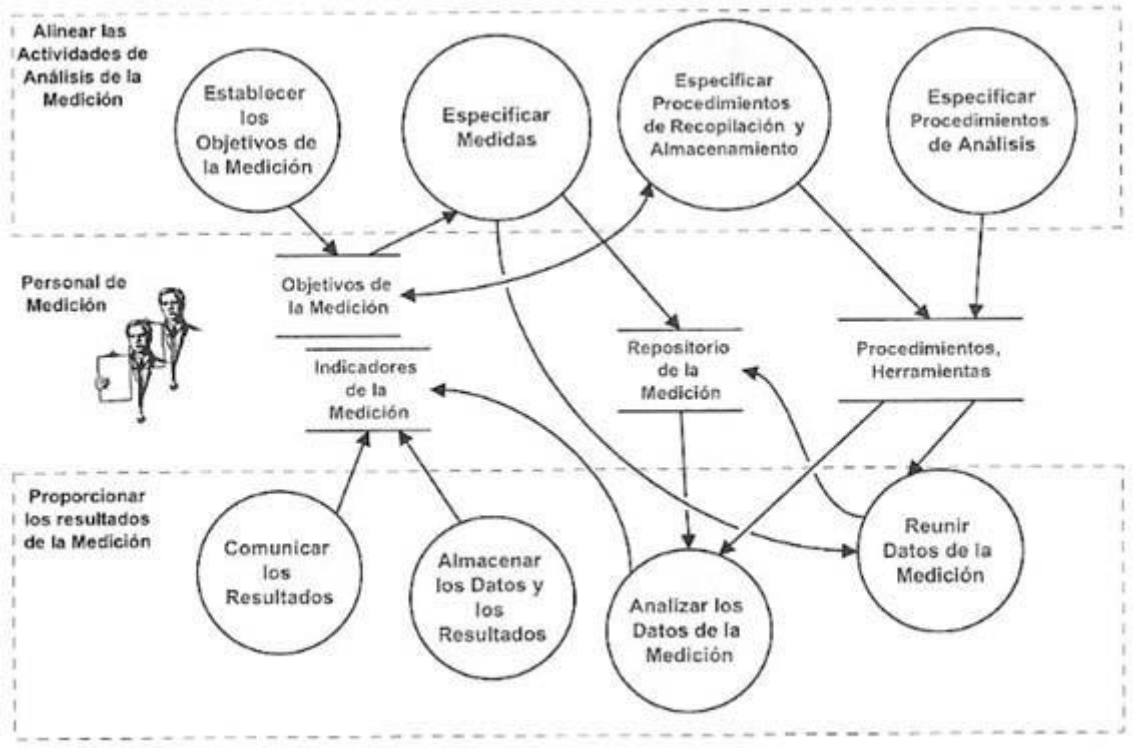


Figura 6. Área «Medición y Análisis» de CMMI (Piattini, García, Garzás y Genero, 2008).

La importancia de la medición en el modelo CMMI también puede verse reflejada por su incorporación en distintas prácticas genéricas del modelo de madurez, tal y como se muestra en la Tabla 2.

Tabla 2. Prácticas genéricas del modelo CMMI que incorporan aspectos de medición (Piattini, García, Garzás y Genero, 2008).

Tarea	Descripción
2.8. Monitorizar y controlar el proceso	Monitorizar y controlar el proceso con respecto al plan para la realización del proceso y llevar a cabo las acciones correctivas más adecuadas.
3.2. Recopilar información de mejora	Seleccionar los productos de trabajo, medidas, resultados de la medición e información de la mejora obtenida a partir de la planificación y realización del proceso para dar soporte a su uso futuro y a la mejora de procesos de la organización.
4.1. Establecer objetivos cuantitativos para el proceso	Establecer y mantener objetivos cuantitativos sobre la calidad y rendimiento del proceso, basados tanto en las necesidades de los clientes como en los objetivos de negocio.
4.2. Estabilizar el rendimiento de los subprocesos	Para cada proceso, equilibrar el rendimiento del conjunto de subprocesos que sea necesario, de cara a poder determinar su capacidad para la obtención de la calidad establecida de una forma cuantitativa y para alcanzar los objetivos de rendimiento del proceso.
5.1. Asegurar la mejora continua del proceso	Asegurar la mejora continua del proceso en la consecución de los objetivos de negocio más relevantes para la organización.