PSEUDO PATRONES DE PROCESOS AGILES

MIGUEL ANGEL OVIEDO VELASCO

JULIO CESAR PALECHOR VALENCIA

Anexos de Monografía para optar por el título de

Ingeniero de Sistemas

Director

Wilson Libardo Pantoja Yépez

Ingeniero de Sistemas

Codirector

Julio Ariel Hurtado Alegría

Ingeniero Electrónico y de Telecomunicaciones

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
Grupo de Investigación IDIS – Investigación y Desarrollo en la Ingeniería de Software
POPAYAN 2010

INDICE DE CONTENIDO

Anexo A: CMMI

En este anexo se define con mayor detalle CMMI, modelo de calidad con reconocimiento internacional, pero difícil de abordar para las MiPymes en el caso de asumir un proyecto de mejora de procesos para este tipo de organizaciones.

Estructura de CMMI.

Área de proceso.

Un área de proceso es un conjunto de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos (CMMI Product Team, 2006).

Componente requerido.

Los componentes requeridos describen lo que una organización debe alcanzar para satisfacer un área de proceso (CMMI Product Team, 2006). Este alcance debe ser implementado visiblemente en los procesos de una organización. Los componentes requeridos son:

***Objetivo genérico*:** Los objetivos genéricos asociados a un nivel de capacidad establecen lo que una organización debe alcanzar en ese nivel de capacidad.

El logro de cada uno de esos objetivos en un área de proceso significa mejorar el control en la ejecución del área de proceso.

***Objetivo específico*:** Estos objetivos se aplican a un área de proceso y localizan particularidades que describen lo que se debe implementar para satisfacer el propósito del área de proceso.

Componente esperado.

Los componentes esperados describen lo que una organización puede implementar para alcanzar un componente requerido (CMMI Product Team, 2006), guía a quienes implementan mejoras o llevan a cabo evaluaciones e incluye prácticas específicas y genéricas:

Práctica genérica: Una práctica genérica se aplica a cualquier área de proceso porque puede mejorar el funcionamiento y el control de cualquier proceso.

Práctica específica*:* Una práctica específica es una actividad que se considera importante en la realización del objetivo específico al cual está asociado.

Las prácticas específicas describen las actividades esperadas para lograr la meta específica de un área de proceso

Componente informativo.

Los componentes informativos proveen detalles que ayudan a las organizaciones a pensar la forma de abordar los componentes necesarios y esperados (CMMI Product Team, 2006). Estos son:

- Propósito
- Notas introductorias
- Nombres
- Tablas de relaciones práctica - objetivo
- Prácticas
- Productos típicos
- Sub-prácticas: Una sub-práctica es una descripción detallada que sirve como guía para la interpretación de una práctica genérica o específica.
- Ampliaciones de disciplina: Las ampliaciones contienen información relevante de una disciplina particular y relacionada con una práctica específica.
- Elaboraciones de prácticas genéricas: Una elaboración de una práctica genérica es una guía de cómo la práctica genérica debe aplicarse al área de proceso.

La figura A-1 representa el esquema utilizando los tipos de componentes como estereotipos UML para describir gráficamente la relación puntual de los componentes del modelo (Hurtado & Barrica, 2005):

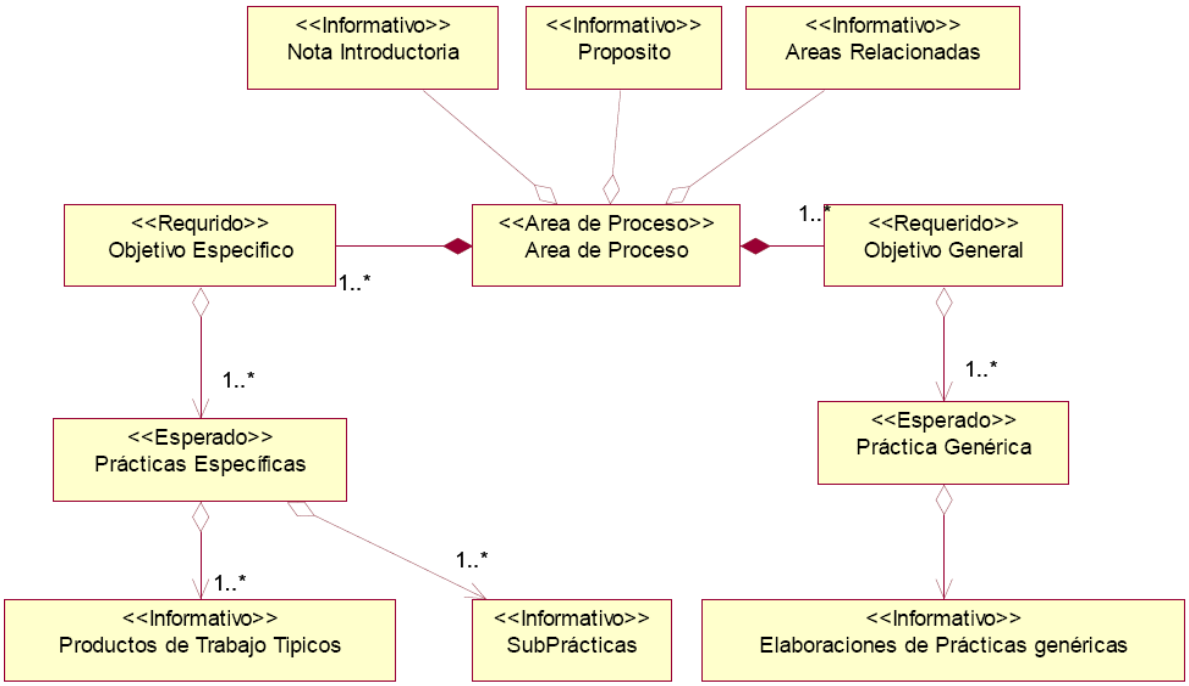

Figura A-1 . Componentes del modelo CMMI

Representación Continúa

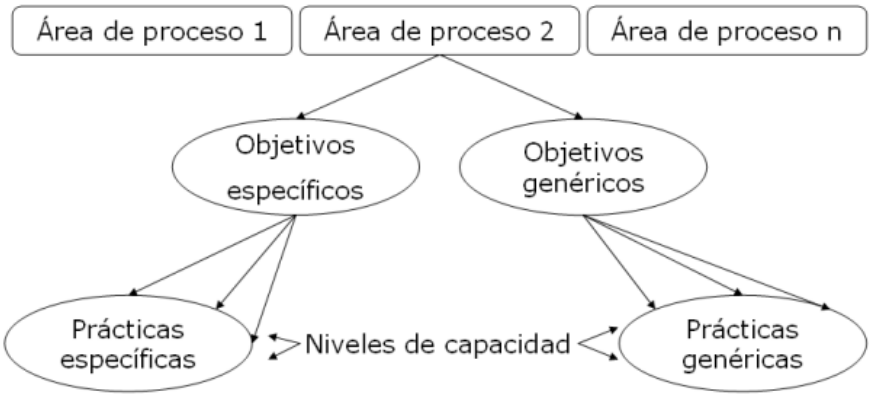

Figura A-2. Estructura de componentes del Modelo CMMI representación contínua.

La figura A-2 representa la forma como la capacidad hace referencia al logro de los objetivos genéricos y específicos que la organización ha alcanzado en una área de proceso específica, de acuerdo a la evaluación de las prácticas específicas y genéricas en el marco de los niveles de capacidad descritos anteriormente, en el escalón más bajo, nivel de capacidad incompleto, la ejecución del proceso no cumple con el propósito del mismo, arriesgando la calidad del producto generado en ese mismo proceso, mientras que en el escalón más alto, nivel de capacidad optimizado, se indica que en la ejecución del proceso se cumplen ampliamente los objetivos del negocio, asegurando la calidad del producto de software.

Si se conocen los procesos que se necesitan mejorar en la organización y se entienden las dependencias entre las áreas de proceso descritas en CMMI, la representación continua es una buena opción para la organización.

La guía de uso de CMMI (Chrissis, Konrad & Shrum, 2003) describe tres factores que pueden influenciar la decisión de seleccionar una representación: el negocio, la cultura y los modelos heredados.

- **Negocio:** En una organización con un conocimiento maduro de sus objetivos de negocio, es posible que tenga un mapeo entre estos y sus áreas de proceso. Una organización de este tipo puede resultarle útil usar la representación continua para evaluar sus procesos y determinar que tan bien estos soportan los objetivos de negocio. En una organización con enfoque de líneas de productos se ajusta más a la representación escalonada, pues esta le ayudará a seleccionar los procesos críticos para enfocar su mejoramiento. La misma organización puede optar por mejorar los procesos por cada línea de productos y obtener diferentes niveles de capacidad por cada línea de productos. El trabajo es conocer los objetivos de negocio y mirar su alineación con cada una de las representaciones y con esto decidir cuál es la mejor opción.

- **Cultura:** Los factores culturales tienen que ver con la forma en que será instalado el programa de mejora. Por ejemplo, se escogería la representación escalonada cuando una empresa tiene experiencia en la mejora de sus procesos o cuando tiene un proceso específico que necesita ser mejorado rápidamente. Una organización con poca experiencia debería escoger la representación escalonada, pues establece un camino de mejoramiento.

- **Modelos heredados:** Si una empresa tiene experiencia con alguna de las representaciones, lo mejor es que siga utilizando la misma representación.

Niveles de madurez de la representación escalonada

Nivel de madurez 1 – Inicial.

En este nivel los procesos son usualmente caóticos. La organización usualmente no provee un ambiente estable. El éxito en la organización depende de las competencias y actos heroicos de la gente y no del uso de procesos probados. En medio de este caos, las empresas producen productos y servicios que trabajan, sin embargo, la producción se excede en sus costos y no cumple con los cronogramas. En este nivel de madurez no existen áreas de proceso.

Nivel de madurez 2 – Gestionado.

La organización ha alcanzado los objetivos específicos y generales de las áreas de proceso de nivel 2. Los proyectos tienen garantizados que los requerimientos son gestionados y que los procesos son planeados, ejecutados, medidos y controlados (CMMI Product Team, 2006).

La disciplina del proceso reflejado en el nivel de madurez 2 ayuda a asegurar que las prácticas existentes se conservan durante momentos de estrés. Cuando estas prácticas están en su lugar, los proyectos se efectúan y se gestionan de acuerdo a sus planes documentados.

Las áreas de proceso del nivel de madurez 2 de CMMI son las siguientes:

- Administración de requerimientos (REQM - requirements management),
- Planificación de proyectos (PP - project planning),
- Monitoreo y control de proyectos (PMC - project monitoring and control),
- Medición y análisis (MA - measurement and analysis),
- Aseguramiento de calidad de procesos y productos (PPQA - process and product quality assurance).
- Gestión de la configuración (CM - configuration management)

Nivel de madurez 3 – Definido.

En el nivel de madurez 3, una organización ha alcanzado todos los objetivos generales y específicos de las áreas de proceso asignados a los niveles de madurez 2 y 3.

En este nivel, los procesos están bien caracterizados y entendidos, y son descritos en estándares, procedimientos, herramientas y métodos. El conjunto de procesos estándar de la organización, los cuales son la base para alcanzar el nivel de madurez 3, es establecido y mejorado en el tiempo. Estos procesos estándar son usados para proveer consistencia en la organización. Los proyectos establecen sus procesos definidos, instanciándolos (tailoring) a partir del conjunto de procesos estándar de la organización de acuerdo a unas guías de adaptación (tailoring guidelines).

Una distinción crítica entre los niveles de madurez 2 y 3 es el alcance de los estándares, descripciones de los procesos y procedimientos. En el nivel de madurez 2, estos pueden ser diferentes en cada instancia específica del proceso, en el nivel 3 estos, obligatoriamente, deberán ser instanciados a partir de los procesos estándares de la organización, excepto por las diferencias que permita la guía de instanciación. Otra diferencia significativa es que los procesos en el nivel 3 son definidos más rigurosamente. Un proceso definido claramente presenta el propósito, entradas, criterios de éxito, actividades, roles, medidas, pasos de verificación, salidas y criterios de éxito.

En este nivel los procesos son gestionados más proactivamente usando la compresión de las interrelaciones entre las diferentes actividades del proceso y sus mediciones y los productos de trabajo y los servicios. En conclusión, el proceso es institucionalizado.

Las áreas de proceso de nivel de madurez 3 de CMMI son:

- Desarrollo de requerimientos (RD -requirements development).
- Solución Técnica (TS - technical solution).
- Integración de producto (PI - product integration).
- Verificación (VER - verification).
- Validación (VAL - validation).
- Gestión del riesgo (RSKM - risk management).
- Enfoque del proceso del organizacional (OPF - organizational process focus).
- Definición del proceso organizacional (OPD - organizational process definition).
- Entrenamiento organizacional (OT - organizational training).
- Análisis y resolución de decisiones (DAR - decision analysis and resolution).
- Gestión integrada de proyectos (IPM - integrated project management).

Nivel de madurez 4 – Cuantitativamente Gestionado.

En el nivel de madurez 4, la organización ha alcanzado todos los objetivos específicos de las áreas de proceso asignadas a los niveles de madurez 2,3 y 4 y los objetivos genéricos asignados a los niveles de madurez 2 y 3.

Objetivos cuantitativos de calidad y desempeño de procesos son establecidos y usados como criterios en la gestión de procesos. Los objetivos cuantitativos son basados en las necesidades del cliente, usuarios finales. La calidad y el desempeño de los procesos se entienden en términos estadísticos y son gestionados mediante la vida de los procesos.

Las áreas de proceso de nivel de madurez 4 de CMMI son: desempeño del proceso organizacional (OPP - Organizational Process Performance) y gestión cuantitativa del proyecto (QPM - Quantitative Project Management)

Nivel de madurez 5 – Optimizado.

En el nivel de madurez 5, la organización ha alcanzado todos los objetivos específicos de las áreas de proceso asignados a los niveles de madurez 2, 3, 4 y 5 y los objetivos generales asignados a los niveles de madurez 2 y 3.

En este nivel la organización mejora continuamente sus procesos basados en el conocimiento cuantitativo de las causas comunes de variación inherente en los procesos (debido a las interacciones normales entre procesos). El foco del nivel de madurez es el mejoramiento continuo del desempeño del proceso, a través de un proceso incremental e innovador y de mejoras tecnológicas.

Las áreas de proceso de nivel de madurez 5 de CMMI son: innovación y despliegue organizacional (OID - Organizational Innovation and Deployment) y análisis y resolución causal (CAR - Causal Analysis and Resolution).

Anexo B: Metodologías Agiles

Este anexo complementa algunas de las características de las metodologías ágiles. El manifiesto ágil, sus principios y valores, XP y su ciclo de desarrollo y SCRUM, entre otros.

Metodologías ágiles.

En la actualidad continuamente se utiliza el concepto de agilidad para definir estas metodologías, pero ¿qué significa ser ágil desde una perspectiva software?.

Basados en el consenso de varias definiciones contemporáneas, Qumer y Henderson-Sellers ofrecieron la siguiente definición de agilidad (Qumer & Henderson-Sellers, 2007):

"La agilidad es un comportamiento persistente o habilidad, de entidad sensible, que presenta flexibilidad para adaptarse a cambios, esperados o inesperados, rápidamente; persigue la duración más corta en tiempo; usa instrumentos económicos, simples y de calidad en un ambiente dinámico; y utiliza los conocimientos y experiencia previos para aprender tanto del entorno interno como del externo."

Por tanto, el desarrollo ágil no especifica unos procesos o métodos que seguir, aunque bien es cierto que han aparecido algunas prácticas asociadas a este movimiento.

El desarrollo ágil es más bien una filosofía de desarrollo software. El punto de partida se establece en las ideas emanadas del Manifiesto Ágil tras la reunión de Utah (Beck et al., 2001), un documento que resume la filosofía *agile* estableciendo cuatro valores y doce principios

En febrero de 2001, tras una reunión Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Tras esta reunión se creó "The Agile Alliance" (Beck, 2002), una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil (Beck et al., 2001), un documento que resume la filosofía ágil (Abrahamsson et al., 2002).

Manifiesto ágil.

En la reunión en las montañas de Utah, 17 desarrolladores convencidos de que era necesario un cambio en las metodologías "clásicas" de desarrollo de software. Entre ellos se encontraban los creadores de XP, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development y Pragmatic Programming. Juntos proclamaron lo que se ha dado a conocer como el "Manifesto for Agile Software Development", estableciendo en él cuatro valores (Beck et al., 2001):

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.

La gente es el principal factor de éxito de un proceso software. Este primer valor expresa que es preferible utilizar un proceso indocumentado con buenas interacciones personales que un proceso documentado con interacciones hostiles. Se considera que no se debe pretender construir primero el entorno y esperar que el equipo se adapte automáticamente sino al revés, construir primero el equipo y que éste configure su propio entorno. El talento, la habilidad, la capacidad de comunicación y de tratar con personas son características fundamentales para los miembros de un equipo ágil.

***Desarrollar software que funcione por encima de una completa documentación*.**

Este valor es utilizado por muchos detractores de las metodologías ágiles que argumentan que éstas son la excusa perfecta para aquellos que pretenden evitar las tareas menos gratificantes del desarrollo software como las tareas de documentación. Sin embargo, el propósito de este valor es acentuar la supremacía del producto por encima de la documentación. El objetivo de todo desarrollador es obtener un producto que funcione y cumpla las necesidades del cliente y la documentación es un artefacto que utiliza para cumplir su objetivo. Por tanto, no se trata de no documentar sino de documentar aquello que sea necesario para tomar de forma inmediata una decisión importante. Los documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente por encima de la negociación contractual.

Se propone una interacción continua entre el cliente y el equipo de desarrollo de tal forma que el cliente forme un tándem con el equipo de desarrollo. Se pretende no diferenciar entre las figuras cliente y equipo de desarrollo sino que se apuesta por un solo equipo persiguiendo un objetivo común.

***Responder a los cambios más que seguir estrictamente un plan*.**

Planificar el trabajo a realizar es muy útil y las metodologías ágiles consideran actividades específicas de planificación a corto plazo. No obstante, adaptarse a los cambios es vital en la industria software actual y, por tanto, también consideran mecanismos para tratar los cambios de prioridades. La regla es "planificar es útil. Seguir un plan es útil hasta que el plan se distancia de la situación actual. Pender de un plan desactualizado es caminar por un callejón sin salida".

Principios del manifiesto ágil.

Este manifiesto se basa en los siguientes principios:

- Satisfacer al cliente a través de entregas continuas y tempranas es la mayor prioridad.
- Los cambios a los requerimientos son bienvenidos, aún en fases tardías del desarrollo.
- Entregar frecuentemente software que funciona, desde un par de semanas a un par de meses, prefiriendo los periodos más cortos.
- Desarrolladores, gerentes y clientes deben trabajar juntos diariamente, a lo largo del proyecto.
- Construir proyectos alrededor de personas motivadas, dándoles el entorno y soporte que necesitan, y confiando en que realizarán el trabajo.
- El método más eficiente y efectivo de transmitir información entre un equipo de desarrolladores es la conversación frontal (cara a cara).
- Tener software que funciona es la medida primaria del progreso.
- El proceso ágil promueve el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo de trabajo constante en forma permanente a lo largo del proyecto.
- La atención continua a la excelencia técnica y el buen diseño mejoran la agilidad.
- Simplicidad – el arte de maximizar el trabajo que no se debe hacer – es esencial.
- Las mejores arquitecturas, requerimientos y diseños surgen de los equipos auto-organizados.
- A intervalos regulares, el equipo debe reflexionar sobre cómo ser más efectivos, y ajustar su comportamiento de acuerdo a ello.

Los desarrollos de software ágil, no son anti-metodológicos, por el contrario, siguen su metodología, diferente a la de los métodos clásicos de desarrollo. Se trata de lograr un equilibrio, por ejemplo, la documentación es concreta y útil, y no burocrática y los planes existen, pero reconociendo sus limitaciones en el actual mundo en permanente cambio.

Programación extrema XP - Extreme Programming.

La metodología XP define cuatro variables para cualquier proyecto de software: **costo, tiempo, calidad** y **alcance**. Además, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres. Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. Este modelo es analizado por Kent Beck, en (Beck & Cleal, 1999), donde propone las ventajas de un contrato con alcances opcionales.

El ciclo de vida de un proyecto XP incluye, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.

Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP (y que serán detalladas más adelante).

Principios de la programación extrema.

a) Trabajo en equipo: El trabajo en equipo es fundamental para sacar un proyecto adelante. XP sugiere grupos no muy grandes (de 2 a 12 desarrolladores) para manejar mejor los problemas del cambio de los requisitos. Es necesario que todos los miembros del grupo estén comprometidos con todo el software producido. XP acepta que uno de los principales y más importantes factores en un equipo de desarrollo son los integrantes del equipo en sí.

b) Participación del cliente: Para que el cliente pueda quedar satisfecho al final de un proyecto de software debe ser parte activa del equipo. Un cliente no puede entregar los requisitos, irse y luego volver a mirar los resultados. Involucrar al cliente no significa que el esté al tanto de los detalles mínimos de implementación, simplemente la idea es que el cliente (o un representante) esté disponible para responder a dudas o preguntas que surjan dentro de las iteraciones del desarrollo.

c) Simplicidad: mantener el diseño y el código lo más simple posible. Con esto se logra una reducción muy importante en los costos de mantenimiento. Por lo tanto se pide al desarrollador, ante todo, simplicidad a la hora de diseñar o implementar. Es tal la importancia de la simplicidad que si se encuentra una forma más sencilla de implementar algo después de hecho, se debería considerar seriamente volver a implementarlo.

d) Buenas pruebas: Con XP se busca eliminar malas prácticas como probar todo el software en los últimos días antes de salir el proyecto. Para esto se propone lograr un buen diseño de pruebas y una ejecución completa y eficiente. XP introduce el concepto de pruebas automatizadas las cuales ayudan a desarrollar mejores aplicaciones. Las pruebas automatizadas proveen al equipo de desarrollo una garantía de que los cambios hechos durante el desarrollo no afectan a otros componentes de la aplicación, o si los afectan, poder ubicar las partes comprometidas fácilmente.

e) La productividad es secundaria: La satisfacción del cliente debe ser el primer objetivo. Idealmente XP mejora la productividad de los desarrolladores pero si en la realidad ésta baja, no debe importar mientras que se logre la satisfacción del cliente.

Ciclo de XP.

Típicamente un proyecto con XP lleva 10 a 15 ciclos o iteraciones.

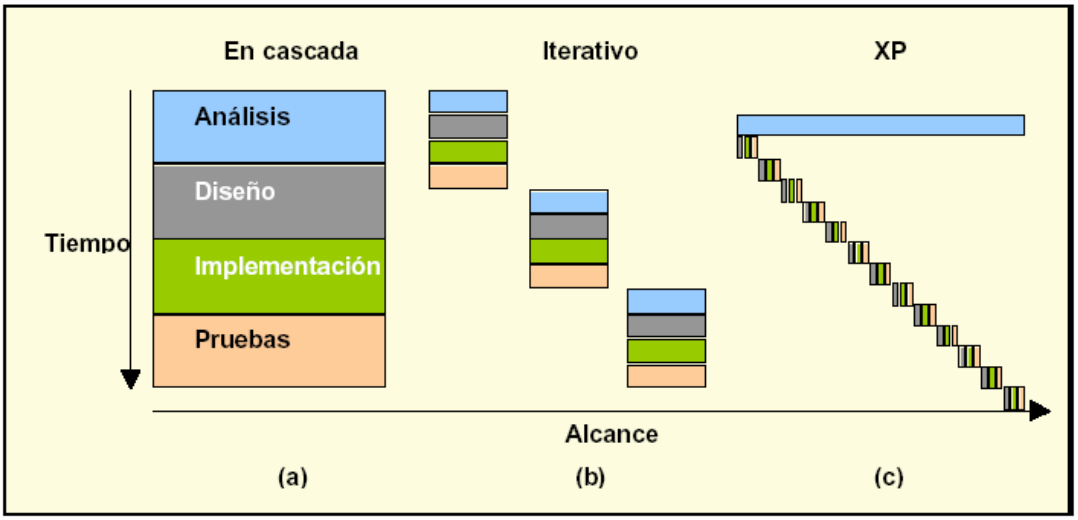

Figura B-1. Ciclos de desarrollo de algunas metodologías.

La figura B-1 esquematiza ciclos de desarrollo en cascada e iterativos tradicionales (por ejemplo, incremental o espiral), comparados con el de XP. En una comparación de tiempo contra el alcance a obtener dentro del proyecto de desarrollo de software se puede observar que la metodología en cascada desarrolla todas sus fases en un solo ciclo, las metodologías iterativas dividen ese mismo tiempo en algunos ciclos, pero la programación extrema es la que permite ciclos muy cortos donde se obtienen pequeñas entregas después de aplicar todas las fases de desarrollo.

Fase de exploración.

Fase en la que se define el alcance general del proyecto. El cliente establece lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Fase de planificación.

La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas, o *“Release Plan”*, como se detallará en la sección “Reglas y Prácticas”.

Fase de iteraciones.

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir

su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios.

El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo.

Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

Fase de puesta en producción.

Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa.

En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste ("fine tuning").

Reglas y prácticas para la planificación.

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando "Historias de usuarios", las que sustituyen a los tradicionales "casos de uso". Una vez obtenidas las "historias de usuarios", los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas tiene "riesgos" que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (*"spikes"*), para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas (*"Release Plan"*) en los que todos estén de acuerdo.

Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas "historias de usuarios".

Según Martín Fowler (uno de los firmantes del "Manifesto Agil"), los planes en XP se diferencian de las metodologías tradicionales en tres aspectos (Beck et al., 2001):

- Simplicidad del plan. No se espera que un plan requiera de un "gurú" con complicados sistemas de gerenciamiento de proyectos.
- Los planes son realizados por las mismas personas que realizarán el trabajo.
- Los planes no son predicciones del futuro, sino simplemente la mejor estimación de cómo saldrán las cosas. Los planes son útiles, pero necesitan ser cambiados cuando las circunstancias lo requieren. De otra manera, se termina en situaciones en las que el plan y la realidad no coinciden, y en estos casos, el plan es totalmente inútil.

Los conceptos básicos de esta planificación son los siguientes:

Historias de Usuarios

Las "Historias de usuarios" (*"User stories"*) sustituyen a los documentos de especificación funcional, y a los "casos de uso". Estas "historias" son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el

momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

Plan de Entregas ("Release Plan")

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión "Juego de planeamiento" (*"Planning game"*), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, *"Planning meeting"* o *"Planning workshop"*)

Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

Plan de Iteraciones ("Iteration Plan")

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Así mismo, por cada historia de usuario se establecen pruebas de aceptación.

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

Reuniones Diarias de Seguimiento ("Stand-up meeting")

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar.

Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

Reglas y prácticas del diseño.

La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

Simplicidad

Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

Soluciones "spike"

Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "spike"1), para explorar diferentes soluciones. Estos programas son únicamente para probar o evaluar una solución, y suelen ser desechados luego de su evaluación.

Recodificación

La recodificación (*"refactoring"*) consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, mas clara y eficientemente. Sin embargo, como ya está pronto y "funciona", rara vez es reescrito. Las metodologías de XP sugieren recodificar cada vez que sea necesario. Si bien, puede parecer una pérdida de tiempo innecesaria en el plazo inmediato, los resultados de ésta práctica tienen sus frutos en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad. La filosofía que se persigue es, como ya se mencionó, tratar de mantener el código más simple posible que implemente la funcionalidad deseada.

Metáforas

Una "metáfora" es algo que todos entienden, sin necesidad de mayores explicaciones. La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo.

Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta "metáfora", para que puedan dialogar en un "mismo idioma". Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Sin embargo, esta práctica resulta, muchas veces, difícil de realizar.

Reglas y prácticas del desarrollo del código.

Disponibilidad del Cliente

Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP.

Al comienzo del proyecto, el cliente debe proporcionar las historias de usuarios.

Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el

cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo. No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, “cara a cara” a los desarrolladores.

Si bien esto parece demandar del cliente recursos por un tiempo prolongado, debe tenerse en cuenta que en otras metodologías este tiempo es insumido por el cliente en realizar los documentos detallados de especificación.

Adicionalmente, al estar el cliente en todo el proceso, puede prevenir a tiempo de situaciones no deseables, o de funcionamientos que no eran los que en realidad se deseaban. En otras metodologías, estas situaciones son detectadas en forma muy tardía del ciclo de desarrollo, y su corrección puede llegar a ser muy complicada.

Uso de estándares

Si bien esto no es una idea nueva, XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.

Programación dirigida por las pruebas (“Test-driven programming”)

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de las mismas, es usualmente realizada sobre el final del proyecto, o sobre el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, en el que, lo primero que se escribe son las pruebas que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas.

Las pruebas a los que se refieren esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estas pruebas al comienzo, condiciona o “dirige” el desarrollo.

Programación en pares

XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que esta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales. En un estudio realizado en (Cockburn & Williams, 2000), se concluye que la programación en pares tiene un sobre costo aproximado de 15%, y no de un 100% como se puede pensar a priori. Este sobre costo es rápidamente pagado por la mejor calidad obtenida en el producto final.

Integraciones permanentes

Todos los desarrolladores necesitan trabajar siempre con la “última versión”. Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

Propiedad Colectiva del Código

En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, cualquier pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar.

En otras metodologías, este concepto puede parecer extraño. Muchas veces se asume que, si hay algo de propiedad colectiva, la responsabilidad también es colectiva. Y que "todos sean responsables", muchas veces significa que "nadie es responsable". Ward Cunningham explica en una entrevista con Bill Veners (Venners, 2003). En este caso, quienes encuentran un problema, o necesitan desarrollar una nueva función, pueden resolverlo directamente, sin necesidad de "negociar" con el "dueño" o autor del módulo (ya que, de hecho, este concepto no existe en XP).

Muchas veces, explica Cunningham, una solución pasa por la recodificación de varios módulos, que atraviesan de forma horizontal una determinada jerarquía vertical. Si es necesario dialogar y convencer al encargado de cada módulo, posiblemente la solución no se pueda implementar, por lo menos en tiempos razonables. En XP, se promueve la recodificación, en aras de generar códigos más simples y adaptados a las realidades cambiantes. Cualquier pareja de programadores puede tomar la responsabilidad de este cambio. Las pruebas permanentes deberían de aseguran que los cambios realizados cumplen con lo requerido, y además, no afectan al resto de las funcionalidades.

Ritmo Sostenido

XP indica que debe llevarse un ritmo sostenido de trabajo. Anteriormente, esta práctica se denominaba "Semana de 40 horas". Sin embargo, lo importante no es si se trabajan, 35, 40 o 42 horas por semana. El concepto que se desea establecer con esta práctica es el de planificar el trabajo de manera de mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Cuando un proyecto se retrasa, trabajar tiempo extra puede ser más perjudicial que beneficioso. El trabajo extra desmotiva inmediatamente al grupo e impacta en la calidad del producto. En la medida de lo posible, se debería renegociar el plan de entregas (*"Release Plan"*), realizando una nueva reunión de planificación con el cliente, los desarrolladores y los gerentes. Adicionalmente, agregar más desarrolladores en proyectos ya avanzados no siempre resuelve el problema.

Reglas y Prácticas en las Pruebas.

Pruebas Unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (*"Test-driven programming"*). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código.

En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.

Detección y Corrección de Errores asimismo

Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Así mismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

Pruebas de Aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra” (*“Black box system tests”*). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución.

Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

Roles.

De acuerdo a la propuesta original los roles dentro de la programación extrema son: programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador, consultor, gestor.

Programador

El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Cliente

El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

Encargado de Pruebas (*Tester*)

El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (*Tracker*)

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Entrenador (*Coach*)

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor

Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

Gestor (*Big boss*)

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

SCRUM.

SCRUM es un proceso para la gestión y control del producto que trata de eliminar la complejidad en estas áreas para centrarse en la construcción de software que satisfaga las necesidades del negocio. Es simple y escalable, ya que no establece prácticas de ingeniería del software sino que se aplica o combina, fácilmente, con otras prácticas ingenieriles, metodologías de desarrollo o estándares ya existentes en la organización.

SCRUM se concentra, principalmente, a nivel de las personas y equipo de desarrollo que construye el producto. Su objetivo es que los miembros del equipo trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. Podríamos entender SCRUM como un tipo de ingeniería social que pretende conseguir la satisfacción de todos los que participan en el desarrollo, fomentando la cooperación a través de la auto-organización. De esta forma se favorece la franqueza entre el equipo y la visibilidad del producto. Pretende que no haya problemas ocultos, asuntos u obstáculos que puedan poner en peligro el proyecto.

Los equipos se guían por su conocimiento y experiencia más que por planes de proyecto formalmente definidos. La planificación detallada se realiza sobre cortos espacios de tiempo lo que permite una constante retroalimentación que proporciona inspecciones simples y un ciclo de vida adaptable. Así, el desarrollo de productos se produce de forma incremental y con un control empírico del proceso que permite la mejora continua (Schwaber & Beedle, 2006).

Figura B-2. Desarrollo convencional contra desarrollo SCRUM

La figura B-2 resume la forma como se asumen las propiedades (descritas en la columna central) en el desarrollo de software desde el modelo en cascada y desde SCRUM.

Proceso en SCRUM.

Independientemente del tipo de metodología que se utilice, cualquier desarrollo software parte siempre de un mismo problema: conocer las necesidades de los clientes. SCRUM, al igual que el resto de metodologías ágiles, pretende no centrar las tareas de desarrollo en un conjunto de requisitos formalmente definidos sino que aboga por la incorporación del cliente como un miembro más del equipo de desarrollo. De este modo, no se considera el proceso de definición de requisitos como un

fin dentro del desarrollo del proyecto, sino que los requisitos aparecen implícitamente dentro del contenido de las denominadas historias de usuario.

Las historias de usuario son el elemento base que utiliza SCRUM para describir las características que el usuario espera que tenga el software que se va a desarrollar (Cohen, 2004). Por lo tanto, pueden incorporar tanto cuestiones relacionadas con las funciones del sistema como con cualquier otro aspecto del mismo (restricciones, rendimiento, etc.).

Las historias de usuario se presentan desde la perspectiva del usuario. Así, no se describen utilizando una terminología técnica sino que se escriben utilizando un lenguaje cercano al dominio de la aplicación que se está desarrollando de forma que sea comprensible por los clientes y por los desarrolladores. Las historias de usuario se construyen bajo un mismo esqueleto que centra el foco de las características del producto.

Primero se determina *quién* propone la historia de usuario, luego se describe la *característica* que se cubre con la historia de usuario y finalmente se especifica la *razón* por la que dicha característica es necesaria.

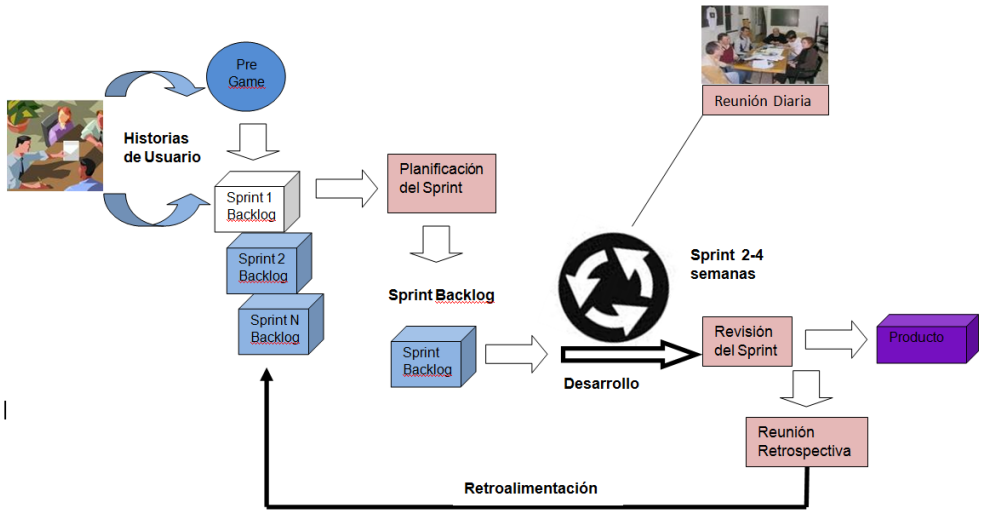

Figura B-3. Ciclo de vida de desarrollo SCRUM

De acuerdo a la figura B-3, el proceso comienza con la fase de Pre-game, en la que se realiza de forma conjunta con el cliente una definición sencilla y clara de las características que debe tener el sistema que vaya a ser desarrollado, definiendo las historias de usuario que van a guiar el proceso de desarrollo. Posteriormente, cada historia de usuario se refina de manera que se identifican las tareas que son necesarias para llevar a cabo el desarrollo de la historia de usuario. En principio no es necesario detallar de forma completa todas las historias de usuario sino sólo las que tienen un mayor nivel de prioridad por parte del usuario. Esto permite que el proceso de desarrollo pueda adaptarse a posteriores modificaciones de las necesidades de usuario, de forma que el proceso de desarrollo se vuelve más flexible. El resultado del Pre-game es lo que se denomina en SCRUM "*Product Backlog*", que contiene una lista de todas las historias de usuario priorizadas así como de las tareas que se deben llevar a cabo para la realización del proyecto.

Este enfoque basado en quién, qué y por qué facilita la tarea de priorización de las historias de usuario, lo que permite realizar la planificación inicial del proyecto. Como puede deducirse de lo anteriormente expuesto, la obtención del Product Backlog se realiza con la ayuda del cliente/usuario del sistema que se va a desarrollar, por lo que puede considerarse que se realiza en directo y, por lo tanto, las posibles dudas en el establecimiento de las historias de usuario que pueden resolverse.

Una vez identificadas y priorizadas las historias de usuario del Product Backlog, se realiza la separación de historias de usuario en etapas de corta duración (no más de 30 días) denominadas sprints. Para cada sprint se realiza una reunión de planificación de lo que se va a llevar a cabo en ese sprint y se establece la fecha de finalización del mismo. El objetivo es mover aquellas historias de usuario con mayor prioridad para el cliente al denominado “Sprint Backlog”, que contiene el conjunto de tareas a desarrollar en ese sprint, incluyendo diseño, desarrollo, pruebas, integración, etc.

Las historias de usuario se congelan en el Sprint Backlog de forma que durante dicho periodo no puedan producirse cambios sobre los aspectos que se encuentran en fase de desarrollo.

De forma iterativa, todos los días que dure el sprint, se realiza una reunión operativa, informal y ágil con el equipo de desarrollo, de un máximo de quince minutos, en la que a cada integrante del equipo se le hacen tres preguntas:

¿Qué tareas ha hecho desde la última reunión? Es decir, tareas realizadas en un día.
¿Qué tareas va ha hacer hoy?
¿Qué ayuda se necesita para poder realizar este trabajo? Es decir, identificación de obstáculos o riesgos que impiden o pueden impedir el normal avance.

Una vez finalizado el sprint se debería obtener parte del producto, un entregable o algo que se pueda mostrar y que enseñe los avances acometidos en el Sprint. En este punto se procede a una fase de revisión del proyecto para mostrar dichos avances e incorporar, si fuera necesario, nuevas historias de usuario al Product Backlog. Para mantener la calidad del producto se establece que una historia de usuario está 100% completa si supera los test unitarios, pasa las pruebas de aceptación, supera los test adicionales para mantener la calidad del producto, el código está construido e integrado satisfactoriamente, está basado en un diseño factorizado sin duplicaciones, el código es claro, estructurado y autodocumentado y, por supuesto, es aceptado por el cliente.

Además, tras la realización de cada sprint se lleva a cabo una reunión retrospectiva que permite aprender de los conocimientos y experiencias adquiridas hasta el momento y mejorar de forma continua el proceso de desarrollo. Se revisará con el equipo los objetivos marcados inicialmente en el Sprint Backlog concluido, se aplicarán los cambios y ajustes si son necesarios, y se marcarán los aspectos positivos (para repetirlos) y los aspectos negativos (para evitar que se repitan) del Sprint, que servirán de retroalimentación para el siguiente sprint. Los elementos que servirán de entrada y guiarán la planificación de un nuevo sprint serán el Product Backlog, las capacidades o habilidades del equipo, las condiciones que en ese momento se den del negocio, el avance tecnológico que se haya podido producir y el incremento que se pretenda hacer al producto que se está desarrollando. De esta forma, en contraposición con las metodologías convencionales, el proceso de desarrollo adquiere flexibilidad, agilidad y capacidad para adaptarse a su entorno.

Roles.

Para orquestar este proceso SCRUM distingue distintos actores con diferentes papeles dentro del proceso. De forma general, podemos distinguir propietario del producto ó *Product Owner*, maestro de scrum ó *Scrum Master*, equipo de desarrollo ó *Scrum Team* y *cliente* o usuario.

Product Owner .

Es la única persona encargada de la dirección y control del Product Backlog, es decir, de las historias de usuario que debe cumplir el sistema. Se trata de una persona física (solamente una persona para eliminar las posibles confusiones o interferencias), no una organización o comité. Bien puede ser el propio cliente *in situ* en el lugar de desarrollo u otra persona que tenga el conocimiento suficiente sobre el producto o pueda estar en continuo contacto con el cliente para marcar las prioridades del proyecto. Es, por tanto, la persona oficialmente responsable del proyecto que de forma visible, vocal y

objetiva debe tomar todas las decisiones de negocio para el producto. Para que el Product Owner tenga éxito, el resto del equipo de la organización tiene que respetar sus decisiones. En cuanto a su implicación debe estar en constante interacción con el equipo de desarrollo. Debe asistir, al menos, a las reuniones de planificación y de revisión de cada sprint y estar en continuo contacto con el equipo para proporcionar detalles sobre las historias de usuario y constante retroalimentación que dirija el desarrollo del sprint.

Scrum Master.

Es la persona responsable del éxito al aplicar la metodología SCRUM en el desarrollo del proyecto o producto, asegurando que los valores, prácticas y reglas son seguidos por el resto del equipo. En concreto, es la persona que asegura el seguimiento de la metodología guiando las reuniones, ayudando al equipo ante cualquier problema que pueda aparecer y controlando que el trabajo sigua el ritmo adecuado. Por tanto debe tomar decisiones inmediatas y eliminar los impedimentos que vayan surgiendo en el momento, aunque en ocasiones no cuente con toda la información necesaria. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas y motivar al resto del equipo.

Por tanto, la labor de Scrum Master requiere una fuerte personalidad ya que debe facilitar el trabajo del equipo sin imponer autoridad. Las personas que ejercen este role deben ser capaces de hacer cualquier cosa por ayudar al equipo de desarrollo, incluso aunque estas acciones estén enfrentadas con sus propios intereses. Deben ser personas poco conformistas y con mucha iniciativa. Eliminar impedimentos requiere determinación y tenacidad. La metáfora utilizada por Kent Beck refleja perfectamente este papel: *"El Scrum Master es el perro pastor que hace cualquier cosa para proteger al rebaño, y nunca se distrae de su deber".*

Scrum Team.

Lo conforman las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner. Debe ser un conjunto de personas motivadas con habilidades y capacidades complementarias que estén comprometidos por un propósito común, cumplir el objetivo del sprint. El equipo tiene plena autoridad para tomar todas las decisiones que consideren adecuadas en el desarrollo del proyecto, auto-organizándose y auto-disciplinándose. Así, por ejemplo, en las reuniones de planificación el Product Owner y el Scrum Team deben llegar a un acuerdo realista sobre las historias de usuario que se van a completar en el siguiente sprint y si en algún momento el Scrum Team considera que algunas de las prioridades del Product Owner no es razonable dispone de libertad absoluta para reseñar esta circunstancia y obligar al propietario del producto a variar sus prioridades. Teóricamente, se estima que debería estar formado por un número de entre 7 u 8 miembros como máximo y 2 como mínimo.

Cliente(s).

Son los beneficiarios finales del producto, y quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades. Su participación es importantísima e imprescindible en esta metodología.

Anexo C: SPEM y EPFC

Este anexo contiene más detalle acerca del lenguaje y de la herramienta libre utilizada para la sistematización de los pseudo patrones, como la forma de hacerlos aplicables dentro de los procesos de desarrollo de software de las MiPymes desarrolladoras de software.

ECLIPSE PROCESS FRAMEWORK COMPOSER

Contenido de método.

A continuación se presentan los conceptos del Contenido de Método (Method Content) de SPEM 2. No es una revisión exhaustiva, ya que no se incluyen aquellos conceptos que quedan "ocultos" al usuario cuando modela procesos con algún editor de SPEM, por ejemplo, con EPF Composer.

El contenido de método puede ser organizado a voluntad del usuario mediante una jerarquía de paquetes de contenido (Content Package), cada uno de los cuales puede incluir roles, tareas, productos de trabajo y guías.

Elementos de contenido.

Los elementos de contenido (Content Element) son los constructores básicos y se derivan del patrón primitivo de trabajo: alguien (rol) hace algo (tarea) para obtener algo (producto de trabajo) basándose o ayudándose en algo (guía). En consecuencia, los cuatro tipos de elementos de contenido son: Tarea, Rol, Producto de Trabajo y Guía. Adicionalmente, existe un quinto tipo de elemento de contenido incluido con fines de clasificación y agrupación, llamado Categorías. En los siguientes subapartados se presenta cada uno de ellos.

Tarea.

Una Tarea (Task Definition) describe una unidad de trabajo asignable y gestionable, es decir, es la unidad atómica de trabajo para definir procesos. Su granularidad es de unas pocas horas a unos pocos días, afectando a unos pocos productos de trabajo y vinculando a unos pocos roles. Es un Elemento de Método que define el trabajo realizado por roles, pero también es una definición de trabajo (en procesos).

Roles.

Un Rol (Role Definition) define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo. No se deben confundir roles con personas, ya que la vinculación entre personas y roles se realiza durante la planificación del proyecto y puede ocurrir que un individuo desempeñe varios roles o que un rol sea desempeñado por varios individuos. Un rol es un Elemento de Método usado en las Definiciones de Tareas para señalar quienes las realizan.

Productos de Trabajo.

Un Producto de Trabajo (Work Product Definition) es consumido, producido o modificado por Tareas. Un Producto de Trabajo puede estar asociado con otros 0..* Productos de Trabajo.

Existen tres tipos predefinidos (especializaciones) de Productos de Trabajo:

- Artefacto (Artifact): de naturaleza tangible (modelo, documento, código, archivos).Un artefacto puede estar formados por otros artefactos más simples.
- Entregable 5 (Deliverable): provee una descripción y definición para empaquetar otros productos de trabajo con fines de entrega a un cliente interno o externo. Representa una salida de un proceso que tiene valor para un usuario, cliente u otro participante (*stakeholder*).

Está asociado con 0.* componentes de entregable (Deliverable Component), que son los productos de trabajo, habitualmente artefactos, que lo forman.
- Resultado (Outcome): un producto de trabajo de naturaleza intangible (resultado o estado), o que no está formalmente definido.

***Guías*.**

Una guía o instrucción (Guidance) es un elemento de método (o de proceso) que provee información adicional relacionada con otros elementos. Por ejemplo: ayuda o información sobre cómo trabaja un rol, cómo crear un producto de trabajo, cómo usar una herramienta o cómo realizar una tarea.

***Categorías*.**

Una Categoría (Category) es un elemento de contenido, o de proceso, usado para categorizar, es decir, clasificar o agrupar dichos elementos en base a los criterios que desee el ingeniero de procesos. Una categoría puede tener 0..* subcategorías. Esto permite establecer cualquier tipo de jerarquía de agrupamiento de elementos. SPEM 2 distingue dos clases de categorías:

- Estándar (Standard Category): Vienen predefinidas en SPEM.
- Personalizada (Custom Category): Sirven para que el ingeniero de procesos pueda definir otras categorías nuevas.

Elementos de Desglose de Trabajo.

Los Elementos de Desglose de Trabajo (Work Breakdown Element) son el principal tipo de elemento de desglose, ya que representan la descomposición del trabajo. Existen dos tipos: Actividad e Hito. El flujo entre los Elementos de Desglose de Trabajo se representa por medio de Secuencias de Trabajo (Work Sequence). Cada Secuencia de Trabajo conecta dos Elementos de Desglose de Trabajo, nominados como predecesor y sucesor.

En SPEM 2 la descomposición del trabajo en distintos niveles de detalle se realiza mediante el concepto de Actividad (Activity), de forma que cualquier elemento de desglose de trabajo con estructura interna (es decir, que incluye elementos de desglose) recibe el nombre de Actividad, independiente del nivel de desglose, es decir, del tamaño del fragmento de trabajo representado. En consecuencia, una estructura de desglose de trabajo (WBS) se representa mediante una recursividad de agregaciones: una actividad está formada por agregación de diversos elementos de desglose, entre los que se encuentran actividades más pequeñas que a su vez pueden estar formadas de igual manera.

Actividades.

Una Actividad (Activity) representa una unidad de trabajo general en un proceso. Una actividad puede tener estructura interna formada por agregación de elementos de desglose, que pueden ser de varios tipos, no solo de trabajo: actividades más simples (anidamiento), elementos de método en uso (roles en uso, productos de trabajo en uso), e hitos. La complejidad de la estructura de desglose de una actividad puede variar entre 0 tareas (pueden estar formadas sólo por productos de trabajo en uso) o todo un proceso completo.

Contenido de método en uso.

Los elementos del contenido de método (tareas, roles y productos de trabajo) reutilizados en los procesos reciben el nombre de elementos en uso. Un elemento en uso es una instancia de un elemento de método particularizada para un contexto de proceso determinado.

Por ello, los elementos en uso no son reutilizables (para reuso ya están las descripciones de los elementos en el contenido de método).

Así, una Tarea en Uso (Task Use) representa la ocurrencia de una Definición de Tarea (Contenido de Método) en el contexto de una Actividad. En ella se pueden precisar y modificar, respecto de la tarea original, su documentación, pasos, roles y productos de entrada y de salida. Además, se pueden definir dos asociaciones que no aparecen en las tareas como elementos de contenido de método: los roles que asisten en la tarea y las entradas externas. Existen dos maneras de utilizar tareas en actividades:

a) Reutilizar una descripción de tarea del contenido de método, que pasa a ser una tarea en uso, o
b) Crear directamente en la actividad una instancia de tarea en uso. En este caso no se tendrá descripción asociada, sino sólo el nombre.

Un Producto de Trabajo en Uso (Work Product Use) representa la ocurrencia de un Producto de Trabajo real en el contexto de una Actividad, pudiendo modificar su documentación. Además, se pueden definir dos atributos que no aparecen en la definición de producto de trabajo en el contenido de método: los estados de entrada y de salida del producto en la actividad. Por ejemplo, un producto "Requisitos" puede entrar en una actividad en el estado "identificados" y salir en el estado "refinados".

También existen las dos maneras de utilizar productos de trabajo en actividades equivalentes a las dos indicadas para las tareas:

a) Reutilizar una descripción de producto de trabajo del contenido de método, que pasa a ser un producto de trabajo en uso.
b) Crear directamente en la actividad una instancia de producto de trabajo en uso, que sólo tendrá nombre pero no descripción asociada.

De forma similar a las dos anteriores, un Rol en Uso (Role Use) representa la ocurrencia de un Rol real en el contexto de una Actividad, pudiendo particularizar la documentación, productos de los que es responsable o que modifica, y equipos (roles compuestos) a los que pertenece. Igualmente, existen dos maneras de utilizar roles en actividades:

a) Reutilizar una descripción de rol del contenido de método, que pasa a ser un rol en uso. SPEM le llama Realizador (Performer).
b) Crear directamente en la actividad una instancia de rol en uso. En este caso no se tendrá descripción asociada, sino sólo el nombre. SPEM le llama Participante (Participant).

Un Rol Compuesto (Composite Role) es un rol en uso especial, que se corresponde con más de una Definición de Rol del Contenido de Método

Tipos de proceso.

En SPEM 2, un Proceso (Process) es un tipo de Actividad que describe una estructura para tipos particulares de proyectos o partes de ellos. Mediante un proceso de SPEM 2 se pueden representar distintos tipos de métodos de ingeniería del software: un proceso, un modelo de procesos completo, un ciclo de vida con sus diversos procesos, o una metodología completa (Ruiz & Verdugo, 2008).

En SPEM 2 existen dos clases principales de procesos:

- Patrón de Proceso (Capability Pattern): Es un “fragmento de proceso” que describe un grupo de actividades reutilizable como solución a algún tipo de problema o situación habitual. Se definen para poder ser empleados más de una vez en uno o varios procesos o con fines de organización. Se pueden almacenar en una jerarquía de Paquetes de Proceso (Process Package). Algunos escenarios de uso de patrones de proceso son:

    - Servir como bloques para construir Procesos para Despliegue o Patrones de Proceso más complejos.
    - Ayudar a la ejecución de proyectos que no siguen un proceso bien definido, sino que trabajan en base a fragmentos de proceso (buenas prácticas) de una manera flexible (métodos ágiles).
    - En formación, para describir el conocimiento de una cierta área clave, buena práctica, disciplina, etc.

- Proceso para Despliegue (Delivery Process): Describe una aproximación completa e integrada para realizar un tipo específico de proyecto, abarcando un ciclo de vida completo de desarrollo o mantenimiento. Sirven como plantillas para planificar y ejecutar los proyectos. En un proceso de despliegue se ensamblan patrones de proceso y elementos en uso (tareas, roles y productos de trabajo en uso). Ejemplos: “proceso unificado de Rational (RUP)”, “programación extrema (XP)”, “mantema, metrica 3”, etc.

Todos los procesos, independiente del tipo que sean, tienen en común las siguientes propiedades: name (nombre), presentation name (nombre de presentación), brief description (descripción breve), external ID (ID externo), purpose (objetivo), main description (descripción principal), scope (ámbito), usage notes (notas de utilización), alternatives (alternativas), how to staff (cómo proveer de personal), key considerations (factores clave). Adicionalmente, un proceso para despliegue tiene éstas otras propiedades: scale (escala), Project characteristics (características del proyecto), risk level (nivel de riesgo), estimating techniques (técnicas de estimación 13), project member expertise (especialidad de miembros del proyecto), type of contract (tipo de contrato).

Los pseudo patrones de proceso pueden ser incorporados como elementos de desglose a una WBS. Esto implica la reutilización automática de todos sus elementos y estructura (actividades, tareas, roles, productos de trabajo, WBS, etc.).

Anexo D: Plantillas de evaluación de CMMI

Este anexo contiene la plantilla que se siguió para realizar la evaluación en las nueve áreas de proceso de CMMI después del uso de los pseudo patrones en un proceso real de desarrollo de software en una MiPyme. Además, se relacionan los resultados registrados en una plantilla por cada práctica específica (y subpráctica) de cada área de proceso.

CMMI NIVEL DE MADUREZ 2 -						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1-	SP 1.1 -					
	SP 1.2 -					
	SP 1.3 -					
	SP 1.4 -					
	SP 1.5 -					

Tabla D-1. Plantilla de evaluación de CMMI

CMMI NIVEL DE MADUREZ 2 - Gestión de Requisitos (REQM)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Gestionar requerimientos	SP 1.1 - Obtener una comprensión de los requerimientos	Establecer los criterios para distinguir a los proveedores apropiados de los requerimientos.	7,67	7	8	8
		Establecer criterios objetivos para la evaluación y aceptación de los requerimientos.	7,67	8	6	9
		Analizar requerimientos para asegurar que se cumplen los criterios establecidos.	8,33	8	8	9
		Alcanzar una comprensión de los requerimientos con el proveedor de requerimientos para que los participantes del proyecto puedan comprometerse con ellos.	8,33	9	8	8
	SP 1.2 - Obtener el compromiso sobre los requerimientos	Evaluar el impacto de los requerimientos sobre los compromisos existentes	8	8	8	8
		Negociar y registrar los compromisos	7,67	8	7	8
	SP 1.3 - Gestionar los cambios de los requerimientos	Documentar todos los requerimientos y los cambios a los requerimientos que son dados o generados por el proyecto	8,67	9	8	9
		Mantener la historia de cambios de requerimientos con la razón del cambio.	9	9	9	9
		Evaluar el impacto de los cambios de requerimientos desde el punto de vista de las partes interesadas relevantes.	8,67	9	8	9
		Poner los requerimientos y los datos de los cambios disponibles para el proyecto	8	8	7	9
	SP 1.4 - Mantener la trazabilidad bidireccional sobre los requerimientos	Mantener la trazabilidad de los requerimientos para asegurar que la fuente de requerimientos de nivel más bajo (derivados) está documentada.	8,33	8	8	9
		Mantener la trazabilidad de los requerimientos desde un requerimiento a sus requerimientos derivados y la asignación a las funciones, a las interfaces, a los objetos, a la gente, a los procesos y a los productos de trabajo.	8	8	8	8
		Generar la matriz de trazabilidad de requerimientos	8,33	8	9	8

	SP 1.5 - Identificar inconsistencias entre el trabajo del proyecto y los requerimientos	Revisar los planes, las actividades y los productos de trabajo del proyecto en cuanto a la consistencia con los requerimientos y los cambios realizados a ellos.	8,33	9	8	8
		Identificar la fuente de la inconsistencia y la razón.	7,67	8	8	7
		Identificar los cambios que necesitan realizarse a los planes y productos de trabajo resultantes de los cambios a la línea base de los requerimientos.	7,67	8	6	9
		Iniciar las acciones correctivas	8,33	9	7	9

Tabla D-2. Plantilla de evaluación de CMMI - Gestión de Requisitos – REQM

CMMI NIVEL DE MADUREZ 2 - Planificación de Proyecto (PP)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Establecer estimaciones.	SP 1.1 - Estimar el alcance del proyecto	Desarrollar una WBS basada en la arquitectura del producto	7,3	7	8	7
		Identificar los paquetes de trabajo en detalle suficiente para especificar estimaciones de las tareas, las responsabilidades y el calendario del proyecto	6,0	6	7	5
		Identificar el producto o los componentes del producto que serán adquiridos externamente	7,7	8	8	7
		Identificar productos que serán reutilizados	7,7	8	8	7
	SP 1.2 - Establecer las estimaciones de los atributos del producto de trabajo y de las tareas	Determinar la aproximación técnica del proyecto (arquitecturas, tecnología).	8,3	8	9	8
		Utilizar métodos apropiados para determinar los atributos de las tareas y productos de trabajo que se utilizarán para estimar los requerimientos de recursos.	7,3	7	8	7
		Estimación de atributos de las tareas y productos de trabajo.	8,7	8	9	9
	SP 1.3 - Definir el ciclo de vida del proyecto	Definir las fases del ciclo de vida del proyecto	9,0	9	9	9
	SP 1.4 - Determinar estimaciones de esfuerzo y de costo	Recopilar los modelos o datos históricos que se utilizarán para transformar los atributos de tareas y productos de trabajo en estimaciones de horas de de trabajo y costos.	8,0	7	9	8
		Incluir necesidades de infraestructura de soporte en la estimación de esfuerzo y costos	8,7	9	9	8

		Estimar el esfuerzo y el costo utilizando modelos y/o datos históricos.	7,3	7	7	8
SG2 - Desarrollar un plan de proyecto	SP 2.1- Establecer el presupuesto y el calendario	Identificar los hitos principales.	7,7	8	7	8
		Identificar los supuestos de calendario.	7,7	8	8	7
		Identificar las restricciones	7,3	8	7	7
		Identificar dependencias de las tareas	6,7	8	5	7
		Definir el presupuesto y el calendario.	7,0	7	7	7
		Establecer los criterios de acción correctiva.	7,3	7	8	7
	SP 2.2 - Identificar los riesgos del proyecto	Identificar los riesgos.	6,3	7	6	6
		Documentar los riesgos.	5,3	5	6	5
		Revisar y obtener el acuerdo con las partes interesadas relevantes sobre la completitud y eficacia de los riesgos documentados.	6,3	5	7	7
		Corregir los riesgos.	5,0	5	7	3
	SP 2.3 - Planificar la gestión de los datos	Establecer los requerimientos y los procedimientos para asegurar la privacidad y la seguridad de los datos	7,3	7	7	8
		Establecer un mecanismo para almacenar los datos y acceder a los datos almacenados	7,7	8	8	7
		Determinar los datos de proyectos que serán identificados, recopilados y distribuidos	7,7	8	8	7
	SP 2.4 - Planificar los recursos del proyecto	Determinar los requerimientos del proceso.	7,7	8	8	7
		Determinar los requerimientos de personal.	7,0	8	7	6
		Determinar los requerimientos de medios, equipamiento y componentes.	7,7	8	8	7
	SP 2.5 - Planificar el conocimiento y las habilidades necesarias	Identificar el conocimiento y habilidades necesarias para realizar el proyecto.	7,7	8	7	8
		Evaluar el conocimiento y habilidades disponibles.	8,0	8	8	8
		Seleccionar mecanismos para proporcionar el conocimiento y habilidades necesarios.	6,3	5	6	8
		Incorporar los mecanismos seleccionados en el plan de proyecto.	7,0	6	7	8
	SP 2.6 - Planificar la involucración de las partes interesadas	Plan de participación de los stakeholders.	7,0	8	5	8
	SP 2.7 - Establecer el plan del proyecto	Establecer el plan de proyecto.	7,7	8	8	7
	SP 3.1 - Revisar los planes que afectan al proyecto	Revisión de planes que afectan al proyecto.	7,3	7	7	8

SG3 - Obtener el compromiso con el plan	SP 3.2 - Reconciliar los niveles de trabajo y de recursos	Documentar todos los compromisos de la organización, tanto completos como provisionales, asegurando el nivel apropiado de signatarios.	7,7	9	6	8
	SP 3.3 - Obtener el compromiso con el plan	Identificar el soporte necesario y negociar los compromisos con las partes interesadas relevantes.	7,0	6	7	8
		Documentar todos los compromisos de la organización, tanto completos como provisionales, asegurando el nivel apropiado de signatarios	7,7	8	8	7
		Revisar los compromisos internos con la dirección según se requiera	6,3	6	5	8
		Revisar los compromisos externos con la dirección según se requiera.	6,0	7	6	5
		Identificar los compromisos sobre las interfaces entre los elementos en el proyecto, y con otros proyectos y unidades de la organización de tal forma que puedan monitorizarse.	7,3	9	8	5

Tabla D-3. Plantilla de evaluación de CMMI – Planificación de Proyecto – PP

CMMI NIVEL DE MADUREZ 2 - Monitorización y Control de proyectos (PMC)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Monitorizar el proyecto frente el plan	SP 1.1 - Monitorizar los parámetros de planificación del proyecto	Seguir el progreso frente al calendario.	7,7	8	7	8
		Monitorizar el costo y esfuerzo consumido del proyecto.	7,7	6	8	9
		Monitorizar los atributos de tareas y productos de trabajo.	7,3	7	7	8
		Monitorizar los recursos proporcionados y los usados.	7,7	8	8	7
		Monitorizar el conocimiento y habilidades del personal del proyecto.	7,3	8	8	6
		Documentar las desviaciones significativas de los parámetros de planificación de proyecto.	8,0	9	9	6
	SP 1.2 - Monitorizar los compromisos	Revisar regularmente los compromisos externos e internos.	7,3	8	7	7
		Identificar los compromisos que no se han cumplido o que tienen un riesgo significativo de no cumplirse	8,0	8	8	8
		Documentar los resultados de las reuniones de los compromisos	7,3	8	6	8
	SP 1.3 - Monitorizar los riesgos del proyecto	Revisar periódicamente la documentación de los riesgos en el contexto del estado actual del proyecto y sus circunstancias.	7,0	7	7	7
		Revisar la documentación de los riesgos, así como cualquier información que sea disponible para incorporar cambios.	6,7	7	6	7

		Comunicar el estado de los riesgos a los afectados relevantes.	7,3	9	7	6
	SP 1.4 - Monitorizar la gestión de los datos	Revisar periódicamente las actividades de gestión de los datos frente su descripción en el plan de proyecto.	8,0	7	8	9
		Identificar y documentar cuestiones significativas y sus impactos.	7,0	6	6	9
		Documentar resultados de revisiones de la actividad de gestión de datos.	6,3	7	7	5
	SP 1.5 - Monitorizar la involucración de las partes interesadas	Revisar periódicamente el estado de la involucración de las partes interesadas.	7,0	7	8	6
		Identificar y documentar cuestiones significativas y sus impactos	6,7	8	6	6
		Documentar los resultados de las revisiones del estado de la involucración de las partes interesadas	7,0	7	7	7
SG1- Monitorizar el proyecto frente el plan	SP 1.6 - Llevar a cabo revisiones de progreso	Comunicar con regularidad el estado de las actividades asignadas y de los productos de trabajo a las partes interesadas relevantes.	8,0	8	9	7
		Revisar el resultado de la recogida y medidas de análisis para controlar el proyecto	8,0	8	8	8
		Identificar y documentar las cuestiones y desviaciones significativas del plan.	7,7	7	8	8
		Documentar las peticiones de cambio y problemas identificados en cualquiera de los procesos o productos de trabajo.	8,7	9	8	9
		Documentar los resultados de las revisiones	7,3	7	7	8
		Hacer un seguimiento de los problemas y peticiones de cambio hasta su cierre.	7,3	8	7	7
	SP 1.7 - Llevar a cabo revisiones de hitos	Realizar revisiones en puntos significativos del calendario del proyecto (tales como la finalización de etapas seleccionadas) con las partes interesadas.	6,7	7	6	7
		Revisar los compromisos, el plan, el estado, y los riesgos del proyecto.	6,3	6	6	7
		Identificar y documentar cuestiones significativas y sus impactos.	7,3	7	7	8
		Documentar resultados de revisiones, sus elementos de acción y decisiones	7,7	8	7	8
		Hacer un seguimiento de los elementos de acción hasta su cierre.	7,7	8	7	8
SG2 - Gestionar acciones correctivas hasta su cierre	SP 2.1 - Analizar los problemas	Recoger los problemas para su análisis.	7,7	8	8	7
		Analizar los problemas para determinar la necesidad de realizar acciones correctivas.	7,7	7	8	8
	SP 2.2 - Llevar a cabo las acciones correctivas	Determinar y documentar las acciones apropiadas necesarias para tratar los problemas identificados.	8,0	8	9	7
		Revisar y obtener el acuerdo con las partes interesadas relevantes sobre las acciones a llevar a cabo.	7,3	7	9	6

		Negociar cambios sobre los compromisos externos e internos.	6,7	6	8	6
	SP 2.3 - Gestionar acciones correctivas	Monitorizar las acciones correctivas hasta su terminación.	8,3	9	9	7
		Analizar resultados de las acciones correctivas para determinar su efectividad.	7,7	7	8	8
		Determinar y documentar las acciones apropiadas para corregir desviaciones sobre los resultados planificados de las acciones correctivas	7,3	8	7	7

Tabla D-4. Plantilla de evaluación de CMMI – Monitorización y Control de Proyectos – PCM

CMMI NIVEL DE MADUREZ 2 - Medición y Análisis (MA)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Alinear actividades de medición y análisis con los objetivos y las necesidades de información	SP 1.1 - Definir cuáles van a ser los objetos de la medición	Documentos de información necesitada y objetivos	7,0	7	8	6
		Priorizar la información necesaria y los objetivos	7,0	8	7	6
		Documento, revisión y actualizar objetos de medición	5,7	5	6	6
		Proveer retroalimentación para redefinir y clarificar la información necesaria y los objetivos como sea necesario	6,7	6	7	7
		Mantener la trazabilidad de los objetivos de medida para identificar la información y los objetivos necesarios	7,3	7	8	7
	SP 1.2 - Especificar medidas	Identificar medidas candidatas basado en objetos de medida documentados	7,0	7	7	7
		Identificar las medidas existentes que ya se ocupan de los objetos de medida.	6,3	7	6	6
		Especificar definiciones operacionales para las medidas	8,0	8	9	7
		Priorizar, revisar y actualizar medidas	7,7	9	8	6
	SP 1.3 - Establecer procedimientos de recolección y almacenamiento de datos.	Identificar fuentes existentes de datos que son generados desde productos de trabajo, procesos o transacciones.	5,7	4	7	6
		Identificar medidas por las cuales los datos son necesitados pero no son actualmente habilitados.	6,0	5	6	7
		Especificar como recolectar y almacenar los datos por cada medida requerida	6,7	6	7	7
		Crear mecanismos de recolección de datos y guías de procesos	6,7	5	7	8
		Soporte de recogida automático de los datos donde sea apropiado y factible.	6,3	5	7	7
		Priorizar, revisar y actualizar los procedimientos de recolección y almacenamiento de datos	7,3	7	7	8
		Actualizar medidas y objetos de medidas como sea necesario	7,7	7	8	8

	SP 1.4 - Establecer el procedimiento de análisis.	Especificar y priorizar los análisis.	7,7	8	8	7
		Seleccionar herramientas y métodos de análisis de datos apropiados	7,3	8	8	6
		Especificar procedimientos administrativos para analizar los datos y comunicar resultados	6,7	8	6	6
		Revisar y actualizar los contenidos propuestos y formato de los reportes y análisis especificados	6,3	7	6	6
		Actualizar medidas y objetos de medida como sea necesario	6,7	6	7	7
		Especificar criterios para evaluar la utilidad de los resultados de análisis	7,3	7	7	8
SG2 - Proporcionar resultados de las mediciones	SP 2.1 - Guardar las mediciones	Generar los datos de medidas derivadas	7,7	8	7	8
		Ejecutar revisión de integridad de datos y cerrar la fuente de los datos como sea posible	7,7	7	8	8
		Obtener los datos de medidas base	8,0	8	8	8
	SP 2.2 - Analizar las mediciones.	Llevar a cabo los análisis iniciales, interpretar los resultados y sacar las conclusiones preliminares	8,7	9	9	8
		Llevar a cabo mediciones y análisis adicionales según sea necesario y preparar los resultados para su presentación	7,7	8	6	9
		Revisar los resultados iniciales con las partes interesadas relevantes	8,3	9	7	9
		Refinar los criterios para análisis futuros	7,7	8	7	8
	SP 2.3 - Almacenar los datos y resultados obtenidos	Revisar los datos para asegurar su integridad, exactitud y ocurrencia	8,7	9	8	9
		Hacer los contenidos almacenados habilitados para uso solamente por grupos y personal apropiado	7,3	8	7	7
		Prevenir que la información almacenada sea usada inapropiadamente	7,0	7	8	6
	SP 2.4 - Comunicar los resultados del proceso a los involucrados	Mantener a los participantes relevantes informados de los resultados de las medidas oportunamente	7,0	7	7	7
		Asistir a los participantes relevantes en el entendimiento de los resultados	7,7	7	8	8

Tabla D-5. Plantilla de evaluación de CMMI – Medición y Análisis – MA

CMMI NIVEL DE MADUREZ 2 - Aseguramiento de calidad de Procesos y Productos (PPQA)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Evaluar objetivamente procesos y productos de trabajo.	SP 1.1 - Evaluar objetivamente los procesos.	Promover un entorno (creado como parte de la gestión del proyecto) que incentive la participación del empleado en la identificación y comunicación de los problemas de calidad	7,0	7	7	7

		Establecer y mantener criterios claramente establecidos para las evaluaciones.	8,0	8	8	8
		Usar los criterios establecidos para evaluar los procesos ejecutados para la adhesión a la descripción de procesos, estándares y procedimientos	7,3	7	7	8
		Identificar cada incumplimiento encontrado durante la evaluación	7,0	7	6	8
		Identificar lecciones aprendidas que podrían mejorar los procesos para productos y servicios futuros.	7,0	7	7	7
	SP 1.2 - Evaluar objetivamente los productos de trabajo y servicios.	Seleccionar los productos de trabajo a evaluar, en base a criterios de muestreo documentados en caso de usar muestreo	7,3	8	6	8
		Establecer y mantener criterios claros para la evaluación de los productos de trabajo	7,3	8	7	7
		Usar los criterios durante las evaluaciones de los productos de trabajo	7,3	8	8	6
		Evaluar los productos de trabajo antes de que ellos sean entregados a los clientes	8,7	9	8	9
		Evaluar los productos de trabajo en los hitos seleccionados en su desarrollo	5,3	5	7	4
		Ejecutar las evaluaciones incrementales de los productos de trabajo y servicios contra la descripción de procesos, estándares y procedimientos.	5,7	6	6	5
		Identificar cada caso de incumplimiento encontrado durante las evaluaciones	6,3	6	5	8
		Identificar lecciones aprendidas que podrían mejorar los procesos para futuros productos y servicios.	7,3	5	8	9
SG2 - Proporcionar comunicación interna objetiva.	SP 2.1 - Comunicar las no conformidades y asegurar su resolución.	Resolver casa incumplimiento con los miembros apropiados del personal como sea posible	7,7	7	8	8
		Documentar los casos de incumplimiento cuando no puedan ser resueltos en el proyecto	7,7	8	9	6
		Escalar casos de incumplimiento que no pueden ser resueltos dentro del proyecto	7,7	8	8	7
		Analizar los casos de incumplimiento para ver si hay tendencias de calidad que puedan ser identificadas y tratadas	7,0	6	7	8
		Asegurar que los participantes relevantes son conscientes de los resultados de las evaluaciones y de las tendencias de calidad en forma oportuna	7,3	7	8	7
		Periódicamente revisar los casos de incumplimiento y tendencias con el	7,7	8	7	8

		administrador designado a recibir y actuar sobre los casos de incumplimientos.				
		Seguir la resolución de los casos de incumplimiento.	8,0	7	8	9
	SP 2.2 - Establecer registro de actividades	Actividades de procesos de registro y aseguramiento de la calidad en detalle suficiente tal que el estado y los resultados son conocidos	7,3	8	7	7
		Revisar el estado y la historia de las actividades de aseguramiento de la calidad como sea necesario.	7,0	7	8	6

Tabla D-6. Plantilla de evaluación de CMMI – Aseguramiento de Calidad de Procesos y Producto – PPQA

CMMI NIVEL DE MADUREZ 2 - Gestión de la configuración (CM)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Establecer líneas base	SP 1.1 - Identificar elementos de configuración	Seleccionar los puntos de la configuración y los productos de trabajo que lo componen basado en criterios documentados	7,0	6	8	7
		Asignar identificadores únicos para la configuración	7,0	6	8	7
		Especificar las características importantes de cada punto de configuración	7,3	7	7	8
		Especificar cuando cada configuración es situado bajo la gestión de la configuración	6,3	6	7	6
		Identificar los responsables propios por cada punto de configuración.	6,7	7	6	7
	SP 1.2 - Establecer un sistema de gestión de configuración	Establecer un mecanismo para gestionar múltiples niveles de control de gestión de la configuración	6,7	6	6	8
		Almacenar y recuperar puntos de la configuración en el sistema de gestión de la configuración	6,3	5	6	8
		Compartir y recuperar versiones de archivos de la configuración	7,3	6	7	9
		Almacenar, actualizar y recuperar registros de gestión de la configuración	6,3	7	7	5
		Crear reportes de gestión de la configuración desde el sistema de gestión de la configuración	6,0	5	7	6
		Preservar los contenidos del sistema de gestión de la configuración.	7,0	6	8	7
		Revisar la estructura de la gestión de la configuración como sea necesario	6,7	7	8	5
	SP 1.3 - Crear o liberar líneas base	Obtener autorización desde la junta de control de configuración (CCB) antes de crear o liberar líneas de base de la configuración	6,7	7	7	6

		Crear o liberar líneas de base sólo desde la configuración en el sistema de gestión de la configuración	7,0	7	7	7
		Documentar el conjunto de puntos de la configuración que son contenidos en una línea de base	5,7	6	6	5
		Hacer el actual conjunto líneas de base disponible	6,3	7	6	6
SG2 - Seguir y controlar los cambios	SP 2.1 - Seguir las peticiones de cambio	Iniciar y registrar las solicitudes de cambio en la base de datos de cambio de solicitudes	8,0	8	9	7
		Analizar el impacto de los cambios y el arreglo propuesto en las solicitudes de cambio	7,3	8	8	6
		Revisar solicitudes de cambio que serán direccionados en la próxima base de línea con aquellos quienes serán afectados por los cambios y obtener su acuerdo	7,3	7	8	7
		Seguir el estado de las solicitudes de cambio a cerrar	6,0	6	7	5
	SP 2.2 - Controlar los elementos de configuración	Control de cambios a elementos de la configuración total a la vida del producto.	6,7	5	8	7
		Obtener la autorización apropiada antes que los elementos de configuración cambiados sean introducidos en el sistema de gestión de configuración.	7,7	6	9	8
		Ejecutar revisiones para asegurar que los cambios no han causado efectos no deseados en las líneas base	7,0	7	8	6
		Registrar cambios en los elementos de configuración y las razones para los cambios como sea apropiado	6,3	7	7	5
SG3 - Establecer la integridad	SP 3.1 - Establecer registros de gestión de configuración	Grabar acciones de gestión de la configuración en detalle por lo que el contenido y el estado de cada elemento de la configuración se conoce y versiones previas pueden ser recuperadas	6,0	6	6	6
		Asegurar que los participantes relevantes han accedido y tienen conocimiento del estado de la configuración de los elementos de la configuración	7,3	7	8	7
		Especificar la última versión de las líneas de base.	7,7	6	9	8
		Identificar la versión de los elementos de la configuración que constituye una línea de base particular	6,7	6	8	6
		Describir las diferencias entre líneas de bases sucesivas	6,0	6	7	5
		Revisar el estado y la historia de cada elemento de la configuración	6,7	6	8	6

	SP 3.2 - Realizar auditorías de configuración	Evaluar la integridad las líneas de base	6,3	6	6	7
		Confirmar que los registros de configuración correctamente identifican la configuración de los elementos de la configuración Revisar la estructura e integridad de los elementos en el sistema de gestión de la configuración	7,0	7	8	6
		Confirmar la integridad y correcciones de los elementos el sistema de gestión de la configuración	6,7	8	7	5
		Confirmar cumplimiento estándares y procedimientos de gestión de la configuración aplicables	7,0	8	7	6

Tabla D-7. Plantilla de evaluación de CMMI – Gestión de la configuración – MC

CMMI NIVEL DE MADUREZ 3 - Solución Técnica (TS)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Seleccionar las soluciones de componentes de producto	SP 1.1 - Desarrollar las soluciones alternativas y los criterios de selección	Identificar los criterios de filtrado para seleccionar un conjunto de soluciones alternativas a considerar	7,7	8	8	7
		Identificar las tecnologías actualmente en uso y las nuevas tecnologías de producto para ventajas competitivas	7,7	8	7	8
		Identificar los componentes de producto comerciales (COTS, commercial off-the-shelf) candidatos que satisfagan los requerimientos	8,0	8	7	9
		Generar soluciones alternativas	8,0	9	7	8
		Obtener una asignación completa de los requerimientos para cada alternativa.	8,0	9	8	7
		Desarrollar los criterios para seleccionar la mejor solución alternativa	7,7	9	7	7
	SP 1.2 - Seleccionar las soluciones de componentes de producto	Evaluar cada solución/conjunto de soluciones alternativas frente a los criterios de selección establecidos en el contexto de los conceptos y de los escenarios operacionales	7,7	7	8	8
		En base a la evaluación de alternativas, evaluar la adecuación de los criterios de selección y actualizar estos criterios según sea necesario	7,3	8	7	7
		Identificar y resolver problemas con las soluciones alternativas y con los requerimientos	7,7	8	7	8

		Seleccionar el mejor conjunto de soluciones alternativas que satisfagan los criterios de selección establecidos	7,0	7	8	6
		Establecer los requerimientos asociados con el conjunto seleccionado de alternativas, así como con el conjunto de requerimientos asignados a esos componentes de producto	7,0	7	7	7
		Identificar las soluciones de los componentes de producto que serán reutilizadas o adquiridas	8,0	8	7	9
		Establecer y mantener la documentación de las soluciones, de las evaluaciones y de los fundamentos	7,3	6	8	8
SG2 - Desarrollar el diseño.	SP 2.1 - Diseñar el producto o el componente de producto.	Establecer y mantener los criterios frente a los cuales puede evaluarse el diseño	7,0	7	7	7
		Identificar, desarrollar o adquirir los métodos de diseño apropiados para el producto.	8,0	8	8	8
		Asegurar que el diseño se adhiere a los estándares y a los criterios de diseño aplicables.	7,7	9	7	7
		Asegurar que el diseño se adhiere a los requerimientos asignados	8,0	9	8	7
		Documentar el diseño.	7,3	8	7	7
	SP 2.2 - Establecer un paquete de datos técnicos	Determinar el número de niveles de diseño y el nivel apropiado de documentación para cada nivel de diseño	8,0	9	8	7
		Basar las descripciones de diseño detallado en los requerimientos asignados de los componentes de producto, en la arquitectura y en los diseños de alto nivel	7,3	7	7	8
		Documentar el diseño en el paquete de datos técnicos.	7,0	6	8	7
		Documentar los fundamentos de las decisiones claves	7,7	8	7	8
		Corregir el paquete de datos técnicos según sea necesario	6,3	4	8	7
	SP 2.3 -Diseñar las interfaces usando criterios	Definir los criterios de la interfaz.	7,0	7	7	7
		Identificar las interfaces asociadas con otros componentes de producto.	8,0	8	7	9
		Identificar las interfaces asociadas con los elementos externos.	7,7	9	8	6
		Identificar las interfaces entre los componentes de producto y los procesos de ciclo de vida asociados al producto	7,0	7	7	7

		Aplicar los criterios para las alternativas de diseño de la interfaz	7,7	7	8	8
		Documentar los diseños de la interfaz seleccionados y los fundamentos de la selección	7,3	8	7	7
	SP 2.4 - Realizar los análisis sobre si hacer, comprar o reutilizar.	Desarrollar los criterios para la reutilización de los diseños de los componentes de producto	7,0	7	8	6
		Analizar los diseños para determinar si deberían desarrollarse, reutilizarse o comprarse los componentes de producto	7,3	8	7	7
		Analizar las implicaciones para el mantenimiento cuando se consideran los elementos comprados o no desarrollados	8,3	9	8	8
SG 3 - Implementar el diseño de producto.	SP 3.1 - Implementar el diseño.	Usar métodos eficaces para implementar los componentes de producto	7,3	7	8	7
		Adherirse a los estándares y a los criterios aplicables	8,0	9	9	6
		Llevar a cabo revisiones entre pares de los componentes seleccionados de producto	7,7	7	8	8
		Realizar pruebas unitarias del componente de producto apropiado	7,3	8	7	7
		Corregir el componente de producto según sea necesario.	8,7	9	8	9
	SP 3.2 - Desarrollar la documentación de soporte de producto.	Revisar los requerimientos, el diseño, el producto y los resultados de pruebas para asegurar que se identifican y resuelven los problemas que afectan a la documentación de instalación, de operación y de mantenimiento	6,7	7	7	6
		Utilizar métodos eficaces para desarrollar la documentación de instalación, de operación y de mantenimiento	7,0	8	6	7
		Adherirse a los estándares aplicables de documentación	7,3	7	7	8
		Desarrollar las versiones preliminares de la documentación de instalación, de operación y de mantenimiento en fases tempranas del ciclo de vida del proyecto para la revisión por las partes interesadas relevantes	8,0	7	8	9
		Llevar a cabo revisiones entre pares de la documentación de instalación, de operación y de mantenimiento	7,7	7	9	7
		Corregir la documentación de instalación, de operación y de mantenimiento según sea necesario	7,3	8	8	6

Tabla D-8. Plantilla de evaluación de CMMI – Solución Técnica - TS

CMMI NIVEL DE MADUREZ 3 - Desarrollo de Requerimientos (RD)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Desarrollar los requerimientos de cliente.	SP 1.1 - Obtener las necesidades.	Comprometer a las partes interesadas relevantes usando métodos para obtener las necesidades, las expectativas, las restricciones y las interfaces externas	7,3	7	7	8
	SP 1.2 - Desarrollar los requerimientos de cliente	Traducir las necesidades, las expectativas, las restricciones y las interfaces de las partes interesadas en requerimientos de cliente documentados	7,7	8	8	7
		Definir las restricciones para la verificación y la validación	7,3	7	7	8
SG2 - Desarrollar los requerimientos de producto.	SP 2.1 - Establecer los requerimientos de producto y de componentes del producto	Desarrollar los requerimientos en los términos técnicos necesarios para el diseño del producto y de componentes del producto	7,7	8	7	8
		Derivar los requerimientos resultantes de las decisiones de diseño	7,0	7	7	7
		Establecer y mantener las relaciones entre los requerimientos para su consideración durante la gestión del cambio y la asignación de los requerimientos.	6,3	6	7	6
	SP 2.2 - Asignar los requerimientos de componentes del producto.	Asignar los requerimientos a las funciones.	7,3	7	8	7
		Asignar los requerimientos a los componentes del producto	7,3	8	8	6
		Asignar las restricciones de diseño a los componentes del producto	7,0	7	7	7
		Documentar las relaciones entre requerimientos asignados	7,7	6	9	8
	SP 2.3 - Identificar los requerimientos de interfaz	Identificar las interfaces tanto externas como internas al producto (es decir, entre las particiones funcionales u objetos)	7,3	6	8	8
		Desarrollar los requerimientos para las interfaces identificadas.	7,0	6	7	8
SG3 - Analizar y validar los requerimientos	SP 3.1 - Establecer los conceptos operativos y los escenarios	Desarrollar los conceptos operativos y los escenarios que incluyan funcionalidad, rendimiento, mantenimiento, soporte y retirada según sea apropiado	7,7	6	8	9
		Definir el entorno en el cual funcionará el producto o los componentes del producto, incluyendo los límites y las restricciones	7,7	7	7	9

		Revisar los conceptos operativos y los escenarios para refinar y descubrir los requerimientos.	7,0	6	8	7
		Desarrollar un concepto operativo detallado, a medida que se seleccionan los productos y los componentes del producto, que defina la interacción del producto, del usuario final y del entorno, y que satisfaga las necesidades operativas, de mantenimiento, de soporte y de retirada.	7,7	7	8	8
SG3 - Analizar y validar los requerimientos	SP 3.2 - Establecer una definición de la funcionalidad requerida	Analizar y cuantificar la funcionalidad requerida por los usuarios finales.	7,7	7	7	9
		Analizar los requerimientos para identificar las particiones lógicas o funcionales	7,0	7	7	7
		Dividir los requerimientos en grupos, en base a los criterios establecidos, para facilitar y para enfocar el análisis de requerimientos.	6,3	6	6	7
		Considerar la secuenciación de las funciones críticas en el tiempo tanto inicialmente como posteriormente durante el desarrollo de componentes del producto	6,7	6	5	9
		Asignar los requerimientos de cliente a las particiones funcionales, objetos, personal o elementos de soporte para dar soporte a la síntesis de las soluciones	6,7	6	7	7
		Asignar los requerimientos funcionales y de rendimiento a las funciones y a las subfunciones.	7,0	7	8	6
	SP 3.3 - Analizar los requerimientos	Analizar las necesidades, las expectativas, las restricciones y las interfaces externas de las partes interesadas para eliminar conflictos y para organizarlos en temas relacionados.	7,3	6	9	7
		Analizar los requerimientos para determinar si satisfacen los objetivos de los requerimientos de nivel más alto	7,3	6	8	8
		Analizar los requerimientos para asegurarse de que son completos, factibles, realizables y verificables	6,7	7	6	7
		Identificar los requerimientos claves que tienen una fuerte influencia en el costo, calendario, funcionalidad, riesgos o rendimiento	7,0	8	7	6
		Identificar las medidas de rendimiento técnico que serán seguidas durante el esfuerzo de desarrollo	7,3	8	8	6

		Analizar los conceptos operativos y los escenarios para refinar las necesidades, las restricciones y las interfaces del cliente, y para descubrir nuevos requerimientos	7,3	8	9	5
SG3 - Analizar y validar los requerimientos	SP 3.4 - Analizar los requerimientos para alcanzar el equilibrio	Usar modelos, simulaciones y prototipos probados para analizar el equilibrio entre las necesidades y las restricciones de las partes interesadas	7,0	7	7	7
		Ejecutar una evaluación de riesgos sobre los requerimientos y la arquitectura funcional	8,0	8	8	8
		Examinar los conceptos del ciclo de vida del producto en cuanto a los impactos de los requerimientos en los riesgos	8,3	7	9	9
	SP 3.5 - Validar los requerimientos	Analizar los requerimientos para determinar el riesgo de que el producto resultante no se ejecute apropiadamente en su entorno de uso previsto	7,7	9	7	7
		Explorar la adecuación y la completitud de los requerimientos desarrollando las representaciones del producto y obteniendo realimentación sobre ellos de las partes interesadas relevantes	7,0	8	7	6
		Evaluar el diseño a medida que madura en el contexto del entorno de validación de los requerimientos para identificar los problemas de validación y para exponer necesidades y requerimientos de cliente sin especificar	7,3	7	9	6

Tabla D-9. Plantilla de evaluación de CMMI – Desarrollo de Requerimientos - RD

CMMI NIVEL DE MADUREZ 3 - Definición de Procesos de la Organización (OPD)						
Objetivo Específico	Practica Específica	Subpráctica	Prom.	P1	P2	P3
SG1- Establecer los activos de proceso de la organización	SP 1.1 - Establecer los procesos estándar	Descomponer cada proceso estándar en elementos que constituyen elementos de proceso hasta el detalle necesario para comprender el proceso.	6,7	8	7	5
		Especificar los atributos críticos de cada elemento del proceso	6,3	8	7	4
		Especificar las relaciones de los elementos del proceso	6,3	8	6	5
		Asegurar que el conjunto de procesos estándar de la organización se adhiere a políticas, estándares y modelos aplicables	6,7	7	7	6
		Asegurar que el conjunto de procesos estándar de la organización satisface las necesidades del proceso y los objetivos de la organización	6,3	6	8	5
		Asegurar que existe una integración apropiada entre los procesos que se incluyen en el conjunto de procesos estándar de la organización.	6,3	6	9	4
		Documentar el conjunto de procesos estándar de la organización	5,7	6	7	4
		Llevar a cabo revisiones entre pares sobre el conjunto de procesos estándar de la organización	6,7	8	6	6
		Corregir conjunto de procesos estándar de la organización según sea necesario	7,0	7	7	7
	SP 1.2 - Establecer las descripciones de los modelos de ciclo de vida	Seleccionar los modelos de ciclo de vida basándose en las necesidades de los proyectos y de la organización.	7,0	6	9	6
		Documentar las descripciones de los modelos de ciclo de vida	6,3	7	7	5
		Llevar a cabo revisiones entre pares de los modelos de ciclo de vida	6,0	8	6	4
		Corregir descripciones de los modelos de ciclo de vida según sea necesario.	6,3	7	7	5
	SP 1.3 - Establecer los criterios y las guías de adaptación	Especificar criterios de selección y los procedimientos para adaptar el conjunto de procesos estándar de la organización	6,3	6	9	4
		Especificar los estándares para documentar los procesos definidos	7,3	8	7	7
		Especificar los procedimientos para proponer y obtener la aprobación de excepciones a partir de los requerimientos del conjunto de procesos estándar de la organización	6,7	6	6	8
		Documentar las guías de adaptación para el conjunto de procesos estándar de la organización.	7,0	9	7	5
		Llevar a cabo revisiones entre pares sobre las guías de adaptación	6,7	8	6	6
		Corregir las guías de adaptación	7,3	6	9	7

SG1- Establecer los activos de proceso de la organización	SP 1.4 - Establecer el repositorio de medición de la organización	Determinar las necesidades de la organización para almacenar, recuperar y analizar las mediciones	6,7	5	7	8
		Definir un conjunto común de medidas de proceso y de producto para el conjunto de procesos estándar de la organización	6,7	7	6	7
		Diseñar e implementar el repositorio de medición	6,7	6	7	7
		Especificar los procedimientos para almacenar, actualizar y recuperar las medidas	7,3	8	6	8
		Llevar a cabo revisiones entre pares sobre las definiciones del conjunto común de medidas y los procedimientos para almacenarlas y recuperarlas	8,0	9	8	7
		Introducir las medidas especificadas en el repositorio	8,0	7	8	9
		Poner los contenidos del repositorio de medición a disposición de la organización y de los proyectos para su uso, según sea apropiado	6,7	6	7	7
		Corregir el repositorio de medidas, el conjunto común de mediciones y los procedimientos, a medida que cambien las necesidades de la organización.	7,0	7	6	8
	SP 1.5 - Establecer la biblioteca de activos de proceso de la organización	Diseñar e implementar la biblioteca de activos de proceso de la organización, incluyendo la estructura de la biblioteca y el entorno de soporte.	7,3	8	7	7
		Especificar los criterios para incluir elementos en la biblioteca	7,7	9	6	8
		Especificar los procedimientos para almacenar y recuperar elementos	7,3	7	7	8
		Introducir los elementos seleccionados en la biblioteca y clasificarlos para facilitar su referencia y recuperación	6,7	6	7	7
		Poner los elementos a disposición de los proyectos para su uso	6,3	5	6	8
		Revisar periódicamente el uso de cada elemento y usar los resultados para mantener los contenidos de la biblioteca.	7,3	7	7	8
		Corregir la biblioteca de activos de proceso de la organización según sea necesario	7,3	8	6	8
	SP 1.6 - Establecer los estándares del entorno de trabajo	Evaluar los estándares del entorno de trabajo, comercialmente disponibles, apropiados para la organización	8,0	9	8	7
		Adoptar los estándares existentes del entorno de trabajo y desarrollar nuevos para cubrir las carencias, basándose en las necesidades del proceso y los objetivos de la organización	7,0	6	7	8

Tabla D-10. Plantilla de evaluación de CMMI – Definición de Procesos de la organización – OPD

REFERENCIAS BIBLIOGRAFICAS

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2002). Agile software development methods Review and analysis. Publicación VTT.

Beck, K. (2002). Una explicación de la Programación extrema: aceptar el cambio. Publicación Addison-Wesley.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., y otros. (2001). Manifesto for Agile Software Development.

Beck, K., Cleal, Dave. (1999). Optional Scope Contracts

Chrissis, M., Konrad, M., Shrum, S. (2003). CMMI Guidelines for Process Integration and Product Improvement. SEI Series in Software Engineering. Addison-Wesley.

Cockburn, Alistair., Williams, Laurie. (2000). The Costs and Benefits of Pair Programming

Cohen, M. (2004). User Stories Applied for Agile Software Development. Publicación Addison-Wesley.

CMMI Product Team. (2006). CMMI for Development", version 1.2. Technical Report, CMU/SEI-2006-TR008, ESC-TR-2006-008, Software Engineering Institute.

Cunningham, W. (2001). Página Oficial Manifesto for Agile Software Development, URL: http://agilemanifesto.org/. Consultado en febrero 2009.

DeMarco, T., Boehm, B. (2002). The Agile Methods Fray. 90–92.

Díaz, Y. (2009). Estudio sobre la correspondencia entre prácticas agiles y practicas CMMI y su aplicación en Pymes.

Dybå, T., Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Inf. Softw. Technol.

Galin, D., Avrahami, M. (2006). Are CMM Program Investments Beneficial? Analyzing Past Studies. 81- 87

Gamma, E. Helm, R. Johnson, Ralph. (1994). Design Patterns. Elements of Reusable Object-Oriented Software.

Gibson, D., Goldenson, D., Kost, K. (2006). TECHNICAL REPORT CMU/SEI-2006-TR-004 ESC-TR-2006-004 - Performance Results of CMMI®-Based Process Improvement.

Goldenson, D., Gibson, L. (2003). Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results. *CMU/SEI- 2003-SR-009*

Goldenson D., Herbsleb, J. (1995). After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success.

Hurtado J., Bastarrica C. (2005). Trabajo de Investigación: Hacia una Línea de Procesos Ágiles Agile SPsL. Versión 1.0.PROYECTO SIMEP-SW (Sistema Integral de mejoramiento de los procesos de desarrollo de software en Colombia).

Paulk, M. (1998). Using the software cmm in small organizations. 350 – 360.

Qumer, A., Henderson-Sellers, B. (2007). An evaluation of the degree of agility in six agile methods and its applicability for method engineering.

Ruiz, F., Verdugo J. (2008). Guía de Uso de SPEM 2 con EPF Composer versión 3.0. Universidad de Castilla-La Mancha. Escuela Superior de Informática Departamento de Tecnologías y Sistemas de InformaciónGrupo Alarcos. URL: http://alarcos.inf-cr.uclm.es/doc/psgc/doc/lec/parte2b/guia-spem2&epf_v30.pdf. Consultado en febrero 2009.

Schwaber, K., Beedle, M. (2006) Agile Software Development with SCRUM.

Schwaber, Ken. (2004). Agile project management with Scrum. Microsoft Press.

Venners, Bill. (2003). Collective Ownership of Code and Text, A Conversation with Ward Cunningham. URL: http://www.artima.com/intv/ownership.html . Consultado en febrero 2009.