Gestión y procesos en empresas de software

Juan Palacio

Introducción

Es tanta la información de estándares, modelos, marcos y prácticas para desarrollo de software, que apostar por uno u otro puede acabar siendo más una decisión guiada por conformidad con la mayoría, que por conocimiento y evaluación de las opciones existentes. Para ayudar en esta decisión este artículo dibuja el marco general de los modelos de procesos y prácticas de la industria del software: CMMI, ISO 15504, Scrum, Extreme Programming, DSDM, MSF, RUP, PMI, etc. en una síntesis ejecutiva que muestra los principios de los principales modelos, sus fortalezas y debilidades.

La metodología empleada en el desarrollo de software es un factor decisivo de su calidad, por lo que conviene conocer adecuadamente el mapa de procesos y marcos de trabajo de la ingeniería del software para adoptar el más adecuado al tipo de proyecto y cultura corporativa de nuestra empresa.

El objetivo de este artículo no es tanto la descripción de las diferentes metodologías, como mostrar cuáles son sus "principios activos", porque al ser éstos pocos y conceptualmente simples, conocerlos simplifica la aparente complejidad del abanico de metodologías, y facilita su comprensión.

Este texto tiene como premisa la realidad sistémica de la empresa, como entidad de múltiples áreas y departamentos que deben comunicarse e interactuar armónicamente con la dirección estratégica y la cultura de la organización.

Amenaza u oportunidad

Todas las empresas quieren producir más rápido, mejor y con menores costes, y sin duda es posible porque la naturaleza del software encierra importantes oportunidades, pero hay que tener cuidado porque también ofrece riesgos considerables, y es la forma de gestionar su desarrollo la que hace de él una materia prima ingestionable y de resultados impredecibles, o una fuente de oportunidades.

La evolución hacia entornos de ingeniería del software adecuados a la empresa requiere cambios severos en la organización, además del convencimiento, implicación y empuje de la gerencia, y los mejores resultados se alcanzan con un modelo propio capaz de aprovechar las características propias de la organización, y de responder en la mejor forma posible a las particularidades de su negocio.

Gestión basada en procesos

Trabajar sin un método fue la forma empleada por los primeros programadores cuando aparecieron los primeros ordenadores en la segunda mitad del siglo pasado. Los pioneros, atraídos por el encanto de diseñar y construir artefactos útiles y verlos funcionar, fueron los primeros en sentarse frente al teclado y decir con una sonrisa:"en un par de meses esto estará funcionando".

Fueron también los primeros en pasar noches en vela programando, engañándose una y otra vez con que "tan sólo es cuestión de un par de días más", demostrando finalmente que al programar sin un método la probabilidad de obtener resultados de calidad en tiempos predecibles es escasa.

SEI, (Software Engineering Institute) denomina "poco maduras" a las organizaciones que desarrollan software sin un método institucionalizado; y emplea una escala de 1 a 5 para determinar el grado de madurez, y en consecuencia el nivel de garantía que se ofrece en cuanto a calidad, predictibilidad y eficiencia en el desarrollo de software.

Las organizaciones que trabajan sin método tienen un nivel de madurez 1, que no quiere decir necesariamente que vayan a producir mal software, de forma ineficiente y con retraso, sino que la probabilidad de éxito en estos aspectos

es baja. Hay equipos que lo consiguen, pero son pocos, y la razón es sencilla: los resultados en estos casos son tan buenos como las personas que los producen, y los buenos programadores escasean.

En este modo de trabajo la calidad del resultado se debe al "saber hacer" de los programadores. El éxito o fracaso de las organizaciones que trabajan sin una metodología depende del conocimiento tácito de su personal. Es el modelo "start-up": un equipo de emprendedores con talento, capaces de construir sistemas de software. El resultado será tan bueno como ellos lo sepan hacer. El cumplimiento de agendas dependerá de su capacidad de previsión y organización. Pero no hay que engañarse, en este caso no se trata de empresas que saben hacer software, sino de personas que saben hacer software.

Este es el guión habitual de las pequeñas empresas de desarrollo que surgen impulsadas por el empuje de sus emprendedores: pueden llegar tan lejos como la combinación de talento, capacidad relacional y de marketing de sus creadores les permitan. O sea, acabarán cerrando o alcanzarán el nivel de mediocridad o éxito que puedan lograr combinando esos elementos, y el crecimiento más allá de ese punto supondrá un reto importante: pasar de personas que saben hacer software a empresa que sabe hacer software, porque para ir más allá ya no tienen que ser ellos, sino la organización la que deberá saber producir con eficiencia y calidad repetible en todos los proyectos.

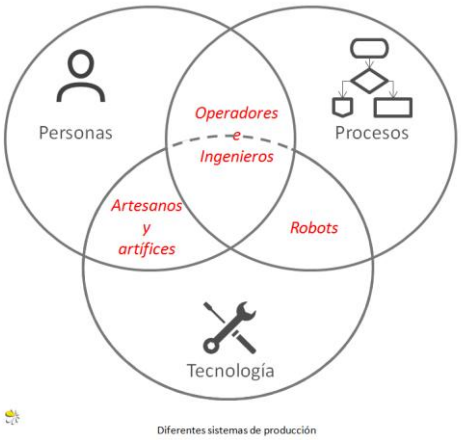
Los objetivos de la gestión basada en procesos son:

- **Repetibilidad**. La calidad del resultado depende de la calidad del proceso, de forma que al aplicar el mismo proceso se obtiene el mismo resultado.
- **Escalabilidad**. Es una consecuencia de la repetibilidad. Los resultados de calidad los obtienen todos los equipos que aplican los procesos.
- **Mejora continua**. Introduciendo ingeniería de procesos se institucionaliza la mejora continua del sistema, y por tanto de los resultados.
- **Know-how propio**. Los procesos de la empresa son los que contienen el "sabe hacer". El modelo de procesos es un activo valioso de la organización: la clave para hacer las cosas con eficiencia y calidad homogénea.

Procesos

Los procesos marcan pautas de trabajo, pero se necesitan además agentes activos: tecnología y/o personas para producir los resultados.

Un ejemplo de las diferencias entre distintos sistemas de producción es el montaje de un mueble en *kit*. Es un trabajo que requiere personas y herramientas o tecnología: destornillador (manual o eléctrico), tornillos, pasadores, cola de montaje, etc. Si realizamos el trabajo sólo con estos dos componentes, sin un proceso determinado (el manual con las instrucciones de cómo proceder) lo haremos en un sistema propio de artesanos o artífices: personas + tecnología, que no puede garantizar productividad y calidad homogéneas, porque dependen de las personas, y hay quien es capaz de ensamblar el mueble en cuestión de minutos, y quien necesita toda una tarde. Habrá quien montará muebles robustos, y otros terminarán con peor fortuna, afirmando que el montaje está bien hecho pero que las piezas venían mal cortadas de fábrica, que quien lo diseñó era un perfecto inútil, o que los tornillos son de mala calidad, o incluso sin percatarse de los fallos que ha cometido.



Diferentes sistemas de producción

Personas + tecnología" es una combinación que produce resultados, y puede resultar más o menos apropiada para un taller de carpintería, pero una empresa de carpintería industrial necesita optar por una de estas estrategias para obtener resultados de calidad:

- Introducir un tercer elemento en el sistema de producción: los procesos.
- Emplear sólo a buenos carpinteros, proporcionándoles las mejores herramientas.

Parece lógico pensar en la primera: implementar un sistema de producción basado en procesos, normalmente la estrategia empleada en la manufactura industrial, aunque hay que tener en cuenta no responde con la misma solvencia en todos los trabajos, porque la gestión basada en procesos tiene su kryptonita: las tareas que requieren un conocimiento aprehensible por las personas pero que resulta difícil explicitar en procesos o tecnología.

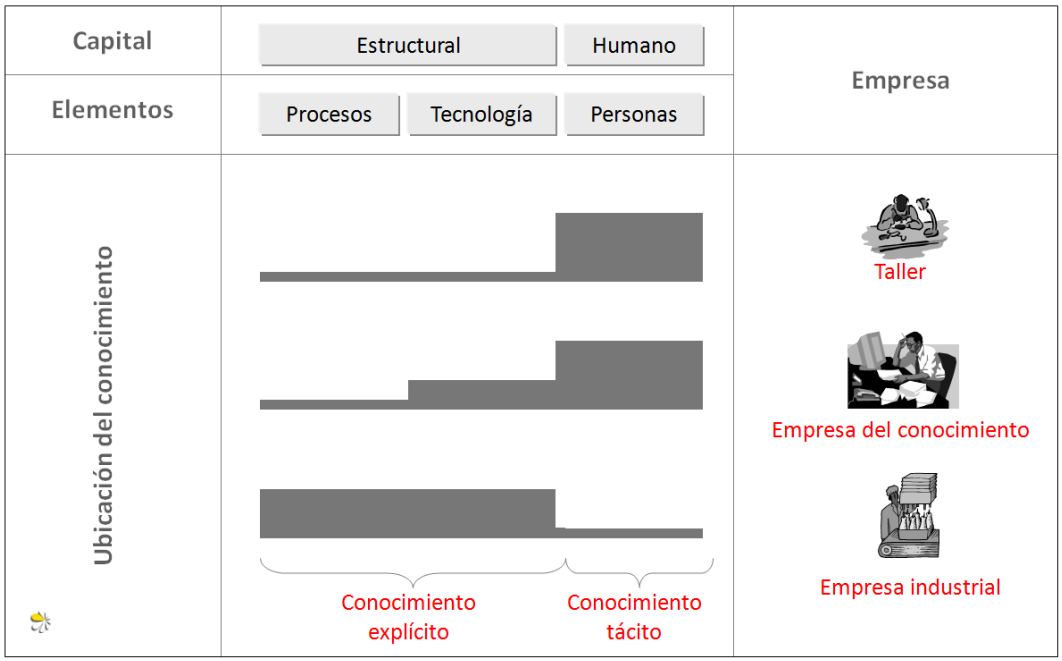
Capital estructural y capital humano

El capital estructural lo forman los bienes que quedan en la empresa cuando las personas están en sus casas: patentes, licencias, cartera de clientes, equipos, maquinaria, vehículos, etc.

El capital humano es el valor de la empresa, que resulta inseparable de las personas. Todas las organizaciones emplean ambos, y la proporción en la que cada uno condiciona la calidad del resultado es distinta en cada caso.

Normalmente el capital estructural es el factor más relevante en las empresas industriales, pero cuál tiene mayor peso en la calidad de la producción no depende sólo del sector o tipo de industria.

En un mismo sector, distintas empresas pueden tener modelos de producción diferentes. Por ejemplo, dos restaurantes. Uno de ellos de alta cocina, y por el otro un franquiciado de "PhonoPizza". Sin entrar en consideraciones gastronómicas, ambos negocios tienen perfectamente definida su identidad, personalidad y segmento; ambos tienen también su propio patrón de calidad. La pizzería emplea una serie de procedimientos para garantizar de forma continua la calidad de su producto, de forma que depende muy poco del conocimiento tácito de los cocineros, y en su mayor parte se debe a los procesos y la tecnología empleada. Los hornos regulan automáticamente el tiempo y la temperatura, los ingredientes se han producido y embolsado en las cantidades adecuadas para cada pizza y se distribuyen de forma masiva a todos los establecimientos en estuches, ambientes frigoríficos y medios de transporte en base a los procesos que son los principales responsables de los resultados. Por esta razón es menos crítico el personal de cocina de un establecimiento dePhonoPizzaque el de un restaurante de alta cocina.



Optimizar el sistema de procesos, tecnología y personas no es fácil. El objetivo es conseguir que cada factor aporte valor hasta el límite de la mejor relación eficiencia / calidad. La revisión permite mejorar de forma continua el sistema e ir consiguiento nuevos equilibrios con mejores parámetros de eficiencia y calidad en el sistema. En esta línea es

aconsejable comenzar analizando si por las características de los proyectos de nuestra empresa resulta más aconsejable un modelo basado en procesos, o en personas (ágil), y comenzar con un marco estándar de una u otra línea, por ejemplo CMMI como modelo de procesos, o Scrum como marco ágil; pero es aconsejable ir más allá de la mera implantación de los procesos o prácticas estándar.

Los modelos de procesos incluyen procedimientos de revisión y mejora continua, de forma que una vez lograda su institucionalización, los resultados mejoran de forma constante.

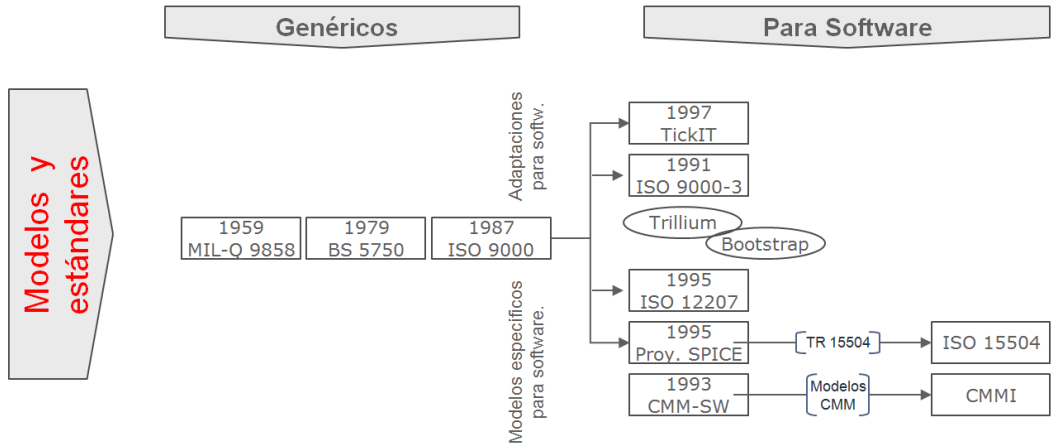
Si se trata de la incorporación de un marco ágil, no debe hacerse de forma metodista, de forma que las prácticas estándar deben ir cuestionándose y revisando para mejorar también de forma constante la fluidez de la organización.

Industria del software

Los modelos de procesos o calidad, y las metodologías de trabajo se resisten a funcionar en la ingeniería del software con la misma solvencia que en las ingenierías industriales, por un lado por la relativa juventud del software que anda aún dando las primeras vueltas de su espiral de conocimiento, y por otro porque el conocimiento para garantizar la calidad del resultado en muchas dimensiones resulta difícil de explicitar en los procesos y la tecnología del sistema de desarrollo.

El siguiente texto extraído del artículo “Criticism of software engineering” de wikipedia expone la situación actual:“*En la ingeniería tradicional hay un claro consenso de cómo deben construirse las cosas, cuáles son los estándares que deben seguirse y qué riesgos deben tratarse: si un ingeniero no sigue estas prácticas y algo falla, puede ser demandado. Sin embargo no hay un consenso similar en la ingeniería del software. Cada uno promueve sus propios métodos, proclamando grandes beneficios en la productividad, que generalmente no respalda con evidencias científicas imparciales.*”

El problema en este momento no es la falta de estándares, modelos o técnicas, sino la abundancia de ellos, por lo que resulta aconsejable tomar un apunte del plano general para saber por dónde nos movemos.



La figura anterior representa la evolución y estado actual de los principales modelos de procesos de referencia para las organizaciones de software.

La norma considerada como punto de arranque de los posteriores estándares y modelos que se han extendido a todas las industrias es la desarrollada por EE.UU. en 1959: “Quality Program Requirements” MIL-Q-9858, diseñada para el ámbito militar que estableció un esquema auditable (a través de la norma de inspección MIL-I.45208) de los requisitos que los proveedores debían cumplir.

El uso de esta norma se fue generalizando mientras en paralelo, diferentes organizaciones comenzaron a desarrollar las suyas propias. Así por ejemplo la OTAN en 1968 adoptó las especificaciones AQAP “Allied Quality Assurance Procedures”.

El estándar británico BS5750, adoptado en el Reino Unido en 1979 fue el siguiente hito relevante en el camino de las normalizaciones, al lograr gran reconocimiento en varios países. Esta normativa fue la precursora de ISO 9000. Los estándares que fueron surgiendo de los 60 a los 90 dieron respuesta a las garantías de homogeneidad, calidad y predictibilidad de los entornos de fabricación y especialmente en el desarrollo de sistemas complejos que comprendían tecnologías e ingenierías diferentes: mecánica, aeroespacial, naval, atómica etc.

Estos sistemas comenzaban a integrar una tecnología recién nacida: el software que aún no disponía de una base de conocimiento asentada y consensuada. Algunos de los fiascos que se produjeron en determinados sistemas por introducción del software sin la madurez suficiente pusieron de manifiesto la necesidad de desarrollar una ingeniería del software que pudiera ofrecer la nueva tecnología con las garantías necesarias.

En las últimas décadas la industria del software ha desarrollado: modelos de procesos para garantizar calidad y eficiencia, y una base de conocimientos técnicos.

En este punto es interesante diferenciar entre unos y otros. Los primeros, los procesos, tienen como objetivo fijar "**qué**" hay que hacer para ofrecer garantías en los proyectos, y los segundos **cómo** debe realizarse. Un marco de procesos determinará, por ejemplo, la necesidad de gestionar adecuadamente las modificaciones de requisitos o de realizar un diseño detallado antes de comenzar la codificación. El valor de los modelos de procesos radica en el conocimiento que aportan al señalar las actividades cuya omisión incrementa las posibilidades de fracaso en los proyectos.

Sin embargo un marco de procesos no establece si el diseño, por ejemplo, debe realizarse con diagramas IDEF o UML. Este es el campo de la técnica de la ingeniería del software. En la línea de los modelos de procesos (**qué** cosas hay que hacer) a partir de finales de los 80 se comenzaron a desarrollar marcos específicos para el software, para cubrir a las particularidades de este producto, en el que las normas generales se quedaban cortas. En esta línea ISO 9000 desarrolló una norma específica para el software: ISO 9000-3, y la BSI (British Standards Institution) hizo lo propio desarrollando TickIT.

En esta primera aparición de estándares surgieron también, aunque con menor repercusión: Trillium y Bootstrap. El primero desarrollado por la empresa Bell Canadá, que liberó sus derechos haciéndolo de dominio público. Bootstrap es una metodología para la evaluación, medición y mejora de los procesos de software. Su desarrollo lo llevó a cabo una comisión del ESPRIT (European Strategic Program for Research). Sin embargo las dos líneas que surgieron a principios de los 90 y que continúan como referentes en la actualidad son. CMMI, y las normalizaciones de ISO 12207 y 15504.

CMM / CMMI

SEI (Software Engineering Institute), un peso pesado en la normalización de los procesos del software, desarrolló una línea de trabajo sobre el concepto de la “madurez” de las organizaciones para producir software.

Por madurez se entiende el nivel de garantía de la organización para asegurar la calidad de sus proyectos (fecha, coste y funcionalidad), la homogeneidad (siempre y en todos los desarrollos) y la capacidad de aprendizaje de la propia experiencia y su aplicación en la mejora continua.

Como resultado de estas líneas de trabajo en 1990 publicó el modelo de madurez de la capacidad para el desarrollo de software (CMM-SW) que tras más de una década de existencia demostró eficacia en muchas organizaciones.

Este modelo emplea una escala de cinco niveles para determinar la madurez de una organización.

Nivel 1.- INICIAL: Pertenecen a él las empresas que no siguen procesos definidos ni aplican técnicas de gestión de proyectos. Son talleres de producción de resultados no predecibles que dependen exclusivamente de la valía de las personas.

Nivel 2.- REPETIBLE: Define a las organizaciones que aplican técnicas de gestión de proyectos, aunque no dispongan de procesos definidos.

Nivel 3.- DEFINIDO: Pertenecen a él las organizaciones que disponen de procesos definidos con precisión y ejecutados de forma regular. Las empresas con este nivel de madurez examinan la experiencia de los proyectos que realizan y emplean las lecciones aprendidas para mejorar sus procesos.

Nivel 4.- GESTIONADO: En el cuarto nivel de madurez se sitúan las organizaciones que han depurado el análisis de los proyectos realizados, hasta institucionalizarlo como procesos que miden cuantitativamente, de forma que pueden predecir cuantificablemente los resultados y evaluar las mejoras con mediciones objetivas.

Nivel 5.- OPTIMIZADO: Las organizaciones con un nivel 5 de madurez tienen definidos, y practican de forma institucionalizada, procesos de mejora continua que se nutren con la información cuantificada de los procesos del nivel 4.

Tras el desarrollo del modelo SW-CMM, SEI desarrolló otros para la medición y mejora de la capacidad de los procesos, todos relacionados con el foco original del SW-CMM: el desarrollo de software:

- Systems Engineering Capability Maturity Model (SE-CMM)
- Integrated Product Development Capability Maturity Model (IPD-CMM)
- People Capability Maturity Model (P-CMM)
- Software Acquisition Capability Maturity Model (SA-CMM)

En 2000 se desarrolló una versión de SW-CMM que integraba algunos de estos modelos (SE-CMM o IPD-CMM). El resultado fue un nuevo modelo que relevó al original SW-CMM: CMMI (Capability Maturity Model Integration).

En la actualidad hay varios modelos CMMI, en función de las áreas que integran

- CMMI-SE/SW/IPPD/SS (Systems Engineering, Software Engineering, Integrated Product and Process Development, Supplier Sourcing).
- CMMI-SE/SW/IPPD (Systems Engineering, Software Engineering, Integrated Product and Process Development).
- CMMI-SE/SE (Systems Engineering, Software Engineering)

Además de la integración de varias disciplinas, los modelos CMMI introdujeron otra novedad que afectaba a su implantación: dos formas posibles de llevara a cabo: escalonada o continua. La versión escalonada prescribe a la organización en qué orden debe abordar las diferentes áreas de procesos, las prácticas que debe implantar y los objetivos que debe alcanzar para ir consiguiendo los sucesivos niveles de madurez. La versión continua permite cierta libertad a la organización sobre las áreas de proceso que desea mejorar, y le orienta para ir mejorando el nivel de capacidad en esas áreas.

CMMI al igual que su predecesor CMM, tiene dos utilidades: puede servir como guía para la mejora en una organización, o como criterio para evaluación, pero mientras CMM centraba estas dos finalidades en la dimensión de la **madurez** de la organización, CMMI introduce una segunda dimensión, también válida para guiar las actividades de mejora y para evaluar a las organizaciones: la**capacidad** de los procesos.

CMMI establece 6 niveles para determinar la capacidad de un proceso:

- Nivel 0.- INCOMPLETO: El proceso no se realiza.
- Nivel 1.- REALIZADO: Se lleva a cabo el proceso, consiguiendo transformar elementos de entrada identificados, en productos de salida.
- Nivel 2.- GESTIONADO: El proceso se ejecuta siempre de la misma manera, de una forma gestionada.
- Nivel 3.- DEFINIDO: El proceso está definido en la organización y se ejecuta siempre.
- Nivel 4.- CUANTIFICADAMENTE GESTIONADO: La ejecución del proceso tiene institucionalizado en la organización un sistema de medición objetivo y cuantificable de su capacidad.
- Nivel 5.- OPTIMIZADO: El proceso, que se ejecuta siempre, está definido en la organización, se mide y está integrado en un plan también institucionalizado de mejora continua, basada en mediciones de parámetros del mismo objetivamente cuantificables.

De esta forma los modelos CMMI presentan 2 versiones:

- Versión escalonada. Guía a la organización acerca de las áreas de procesos que debe ir abordando, las prácticas que debe implantar y los objetivos que debe alcanzar para ir consiguiendo los sucesivos niveles de madurez.
- Versión continua. Permite cierta libertad a la organización sobre las áreas de proceso que desea mejorar, y le orienta para ir elevando el nivel de capacidad de las mismas.

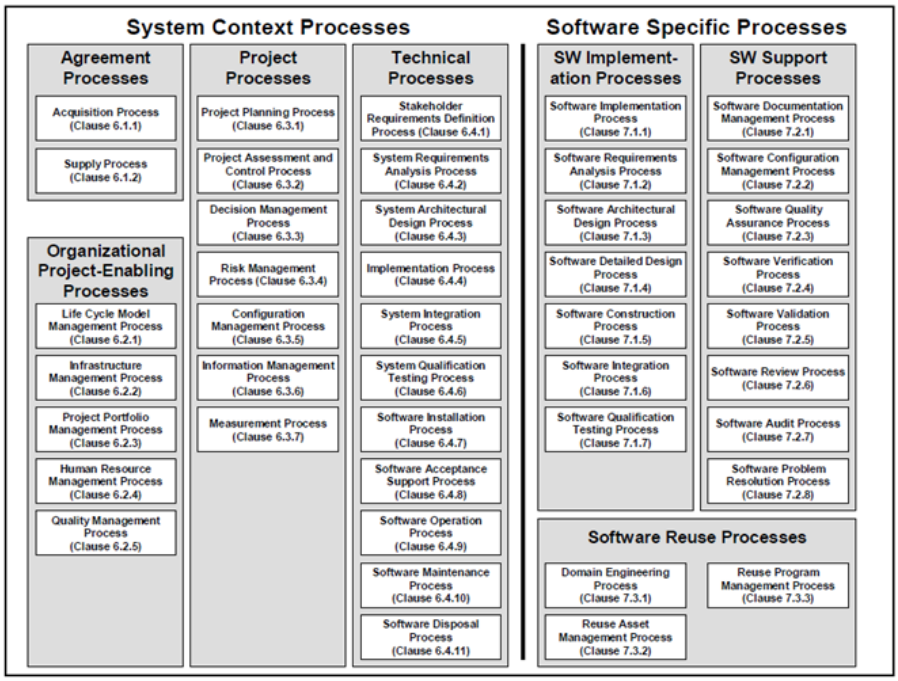
Consecuentemente al evaluar a una organización se puede hacer sobre la dimensión de su madurez, estableciendo cuál es el nivel que ha alcanzado, o sobre la dimensión de la capacidad, reflejando cuál es el nivel de ésta en las diferentes áreas de procesos.

Mientras SEI publicaba y comenzaba a definir su modelo CMM, ISO emprendía los otros dos proyectos que hoy forman los principales puntos de referencia en el ámbito de la calidad para nuestra industria: ISO 12207 e ISO 15504.

ISO/IEC 12207

Es un estándar internacional referente para la normalización de los procesos que constituyen

el ciclo de vida de un proyecto de software, y las tareas que intervienen en cada una de sus fases.



ISO IEC 12207 - Procesos del ciclo de vida de un sistema de software.

El estándar 12207 considera que el ciclo de vida de un sistema de software comienza en el momento que se concibe su idea o necesidad, momento desde el que es necesario comenzar a actuar de manera normalizada para describir el ámbito del problema, las soluciones posibles, etc. El ciclo se extiende desde ese momenoto, cubriendo el desarrollo, mantenimiento y operación; y no concluye hasta que el sistema deja de utilizarse y es definitivamente retirado.

SPICE – ISO/IEC STD. 15504.

En enero de 1993 la comisión ISO/IEC JTC1 aprobó un programa de trabajo para el desarrollo de un modelo que fuera la base de un futuro estándar internacional para la evaluación de los procesos del ciclo de vida del software. Este trabajo recibió el nombre de proyecto SPICE (Software Process Improvement and Capability dEtermination), y en junio de 1995, con la publicación de su primer borrador, desde ISO fueron invitadas diferentes organizaciones para aplicarlo y valorar sus resultados.

En 1998, pasada la fase de proyecto, y tras las primeras evaluaciones, el trabajo pasó a la fase de informe técnico con la denominación ISO/IEC TR 15504.

La instrucción técnica consta de 9 apartados, recogidos en volúmenes independientes, que se han ido publicando como redacción definitiva del estándar internacional ISO/IEC 15504 durante el periodo 2003-2005.

Aunque ISO comenzó con el proyecto SPICE algo más tarde que SEI con el modelo CMM, durante su evolución han ido convergiendo y ambas instituciones vienen trabajando de forma conjunta desde 1998 para lograr la compatibilidad que finalmente han garantizado entre el modelo CMMI y el estándar 15504, de forma que la conformidad con uno de ellos implica la también conformidad con el otro.

El estándar está recogido en 9 documentos:

1. (informativo) Conceptos y guía de introducción.
2. (normativo) Modelo de referencia de los procesos y de su capacidad.
3. (normativo) Ejecución de las auditorías.
4. (informativo) Guía para la realización de auditorías.
5. (informativo) Modelo de asesoría e indicadores.
6. (informativo) Guía de formación de consultores.
7. (informativo) Guía para usar en los procesos de mejora.
8. (informativo) Guía para determinar la capacidad de los procesos del proveedor.
9. (normativo) Vocabulario.

ISO/IEC 15504, al igual que CMMI es un modelo para evaluar los procesos de la organización y determinar si resultan efectivos para conseguir los objetivos. También es un modelo para guiar las acciones de mejora de los procesos. 15504 tiene gran similitud con la representación continua de CMMI. Al igual que este último centra el foco en la capacidad de los procesos, y permite trabajar sólo con una parte de ellos en lugar de hacerlo con todos los de la organización.

ISO 15504 agrupa los procesos de las organizaciones de software en cinco categorías:

- Cliente – proveedor
- Ingeniería
- Soporte
- Gestión
- Organización

Y para la medición de la capacidad de cada proceso define una escala de 6 niveles

- 0 Proceso incompleto
- 1 Proceso realizado
- 2 Proceso gestionado
- 3 Proceso establecido
- 4 Proceso predecible
- 5 Proceso optimizado.

ISO 15504 y CMMI son los referentes de las metodologías formales. Para ambos el centro de su desarrollo son los procesos, no sólo para el desarrollo, mantenimiento y operación de los sistemas de software, sino también para mejorar la capacidad de los propios procesos y la madurez de las organizaciones.

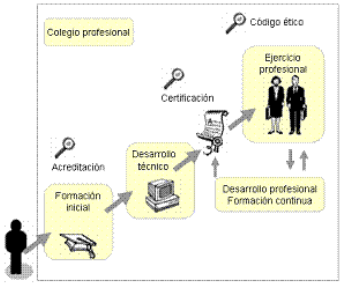
SWEBOK

Aplicar gestión basada en procesos para asegurar un nivel de calidad homogéneo en los resultados es una estrategia adecuada para sistemas de producción en cadena o de manufactura industrial en industrias con una base de conocimiento profesional maduro.

La industria del software sin embargo emplea un conocimiento profesional relativamente joven, y quizá no lo suficientemente asentado, por lo que la aparición de marcos y modelos de calidad se encuentra con un movimiento paralelo de propuestas sobre cuál debe ser el currículo profesional de un ingeniero de software.
La profesionalización exige:

- Disponer de una base de conocimiento, desarrollada con metodología científica, y contrastada con la experiencia.
- Consenso y aceptación por la comunidad que ejerce la profesión.
- Desarrollo de de currículos profesionales, que gracias al consenso resulten homogéneos sobre centros de formación, universidades y países diferentes.
- Desarrollo de las organizaciones académicas y profesionales de autorregulación.

Este es el esquema que han seguido muchas disciplinas para alcanzar un nivel de solvencia profesional socialmente reconocido.



(Gary Ford y Norman Gibbs 1996)

En esta línea han estado trabajando organizaciones profesionales agrupadas en el proyecto SWEBOK (Software Engineering Body Of Knowledge http://www.swebok.org)

Su objetivo ha sido asentar de forma consensuada la base de conocimiento necesaria para el currículo de un ingeniero de software. Primer paso para lograr un entorno profesional, socialmente reconocido.

El trabajo de este proyecto comenzó en 1998, y la publicación de la primera versión, consensuada por todos los participantes, se produjo en Febrero de 2004.

Esto da una idea de que se trata de una profesión con una base de conocimiento y trayectoria curricular que quizá aún no está suficientemente asentada.

La heterodoxia: métodos ágiles

A finales de los 90, reputados profesionales de renombre y autoridad foros técnicos, cuestionaron las metodologías formales, que representadas por CMM e ISO 15504, y respaldadas por la autoridad y los medios de sus respectivas organizaciones estaban definiendo una ingeniería del software basada en procesos y gestión de proyectos predictiva.

En marzo de 2001, 17 críticos de estos modelos, convocados por Kent Beck, que acababa de definir una nueva metodología denominada Extreme Programming, se reunieron en Salt Lake City para discutir sobre los modelos de desarrollo de software.

En la reunión se acuñó el término "**Métodos Ágiles**" para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales, a las que consideraban excesivamente "pesadas" y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas, previas al desarrollo.

Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como "Manifiesto Ágil", que compendia el espíritu en el que se basan estos métodos:

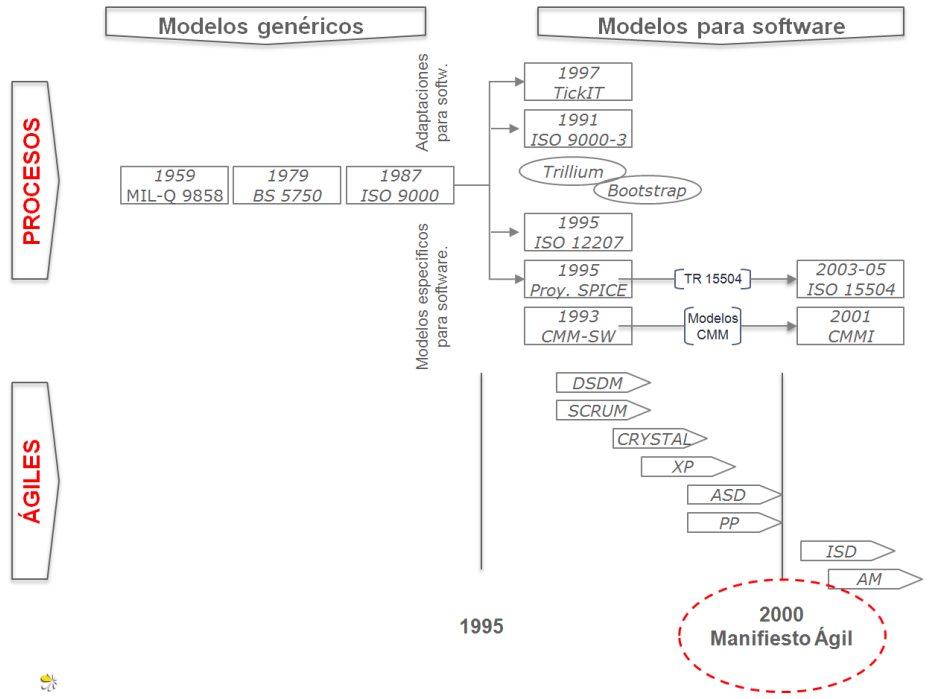
Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- *A los individuos y su interacción por encima de los procesos y las herramientas.*
- *El software que funciona, por encima de la documentación exhaustiva.*
- *La colaboración con el clinte, por encima de la negociación contractual.*
- *La respuesta al cambio, por encima del seguimiento de un plan.*

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.

Los firmantes afirman que los puntos de su manifiesto se sustentan en 12 principios:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.



El manifiesto ágil surgió con espíritu de respuesta desafiante y beligerante contra los métodos basados en procesos. Los propios integrantes del manifiesto se autocalifican como “anarquistas organizacionales”, y en esos años, desde uno y otro lado se lanzaron argumentos punzantes buscando más la descalificación ajena que la justificación propia.

Los principales métodos ágiles son:

DSDM (Dynamic Systems Development Method)

Es la metodología más veterana de las autodenominadas ágiles. Surgió en 1994 de los trabajos de Jennifer Stapleton, la actual directora del DSDM Consortium.

DSDM es la metodología ágil más próxima a los métodos formales, de hecho la implantación de un modelo DSDM en una organización la lleva a alcanzar lo que CMM consideraría un nivel 2 de madurez.

Sin embargo los aspectos que DSDM reprocha a CMM son:

- Aunque es cierto que se ha desarrollado con éxito en algunas organizaciones, lo que funciona bien en unos entornos no tiene por qué servir para todos.
- CMM no le da al diseño la importancia que debería tener.
- CMM plantea un foco excesivo en los procesos, olvidando la importancia que en nuestra industria tiene el talento de las personas.
- El tener procesos claramente definidos no es sinónimo de tener buenos procesos.

En común con los métodos ágiles, DSDM considera imprescindible una implicación y una relación estrecha con el cliente durante el desarrollo, así como la necesidad de trabajar con métodos de desarrollo incremental y entregas evolutivas.

DSDM cubre los aspectos de gestión de proyectos, desarrollo de los sistemas, soporte y mantenimiento y se autodefine como un marco de trabajo para desarrollo rápido más que como un método específico para el desarrollo de sistemas.

Extreme Programming

Este es el método que más popularidad ha alcanzado entre las metodologías ágiles, y posiblemente sea también el más transgresor de la ortodoxia basada en procesos.

Su creador, Kent Beck fue el alma mater del Manifiesto Ágil.

Extreme Programming (XP) se irgue sobre la suposición de que es posible desarrollar software de gran calidad a pesar, o incluso como consecuencia del cambio continuo. Su principal asunción es que con un poco de planificación, un poco de codificación y unas pocas pruebas se puede decidir si se está siguiendo un camino acertado o equivocado, evitando así tener que echar marcha atrás demasiado tarde.

Cuatro son los valores que lo inspiran: simplicidad, feedback, coraje y comunicación.

XP no es un modelo de procesos ni un marco de trabajo, sino un conjunto de 12 prácticas que se complementan unas a otras y deben implementarse en un entorno de desarrollo cuya cultura se base en los cuatro valores citados:

- **Comunicación** XP pone en comunicación directa y continua a clientes y desarrolladores. El cliente se integra en el equipo para establecer prioridades y resolver dudas. De esta forma ve el avance día a día, y es posible ajustar la agenda y las funcionalidades de forma consecuente.
- **Feedback rápido y continuo** Una metodología basada en el desarrollo incremental de pequeñas partes con entregas y pruebas frecuentes y continuas, proporciona un flujo de retro-información valioso para detectar los problemas o desviaciones. De esta forma los fallos se localizan muy pronto. La planificación no puede evitar algunos errores que sólo se evidencian al desarrollar el sistema. La retro-información es la herramienta que permite reajustar la agenda y los planes.
- **Simplicidad** La simplicidad consiste en desarrollar sólo el sistema que realmente se necesita. Implica resolver en cada momento sólo las necesidades actuales. "*Los costes y la complejidad de predecir el futuro son muy elevados, y la mejor forma de acertar es esperar al futuro.*" Con este principio de simplicidad, junto con la comunicación y el feedback resulta más fácil conocer las necesidades reales.
- **Coraje** El coraje implica saber tomar decisiones difíciles. Reparar un error cuando se detecta. Mejorar el código siempre que tras el feedback y las sucesivas iteraciones se manifieste susceptible de mejora. Tratar rápidamente con el cliente los desajustes de agendas para decidir qué partes y cuándo se van a entregar.

Las 12 prácticas que aplicadas sobre esta cultura conforman Extreme Programming son:

PRÁCTICAS DE CODIFICACIÓN

- 1.- Simplicidad de código y de diseño para producir software fácil de modificar.
- 2.- Reingeniería continua para lograr que el código tenga un diseño óptimo.
- 3.- Desarrollar estándares de codificación, para comunicar ideas con claridad a través del código.
- 4.- Desarrollar un vocabulario común, para comunicar las ideas sobre el código con claridad.

PRÁCTICAS DE DESARROLLO

- 1.- Adoptar un método de desarrollo basado en las pruebas para asegurar que el código se comporta según lo esperado.
- 2.- Programación por parejas, para incrementar el conocimiento, la experiencia y las ideas.
- 3.- Asumir la propiedad colectiva del código, para que todo el equipo sea responsable de él.
- 4.- Integración continua, para reducir el impacto de la incorporación de nuevas funcionalidades.

PRÁCTICAS DE NEGOCIO

- 1.- Integración de un representante del cliente en el equipo, para encauzar las cuestiones de negocio del sistema de forma directa, sin retrasos o pérdidas por intermediación.
- 2.- Adoptar el juego de la planificación para centrar en la agenda el trabajo más importante.
- 3.- Entregas regulares y frecuentes para satisfacer la inversión del cliente.
- 4.- Ritmo de trabajo sostenible, para terminar la jornada cansado pero no agotado.

SCRUM

Scrum es un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos autoorganizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

Este modelo fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlett-Packard (Nonaka & Takeuchi, The New New Product Development Game, 1986)

En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo, con el avance en formación de scrum de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para proyectos con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

En 1995 Ken Schwaber presentó “Scrum Development Process” en OOPSLA 95 (Object-Oriented Programming Systems & Applications conference)(SCRUM Development Process), un marco de reglas para desarrollo de software, basado en los principios de scrum, y que él había empleado en el desarrollo de Delphi, y Jeff Sutherland en su empresa Easel Corporation (compañía que en los macrojuegos de compras y fusiones, se integraría en VMARK, y luego en Informix y finalmente en Ascential Software Corporation).

OTROS MÉTODOS ÁGILES

Familia de métodos Crystal

La familia de metodologías Crystal ofrece diferentes métodos para seleccionar el más apropiado para cada proyecto. Crystal identifica con colores diferentes cada método, y su elección debe ser consecuencia del tamaño y criticidad del

proyecto, de forma que los de mayor tamaño, o aquellos en los que la presencia de errores o desbordamiento de agendas implique consecuencias graves, deben adoptar metodologías más pesadas.

Los métodos Crystal no prescriben prácticas concretas, y se pueden combinar con técnicas como XP.

ASD

(Adaptative Software Development) Método que como alternativa a los procedimientos formales, aborda el desarrollo de grandes sistemas con el uso de técnicas propias de las metodologías ágiles.

No se trata de una metodología, sino de la implantación de una cultura en la empresa, basada en la adaptabilidad.

PP

(Pragmatic Programming) Pragmatic Programming es la colección de 70 prácticas de programación, comunes a otros métodos ágiles, cuya aplicación resulta útil para solucionar los problemas cotidianos.

AM

(Agile Modeling) Agile Modeling es la presentación de un nuevo enfoque para realizar el modelado de sistemas,(diseño) y basado en los principios de los métodos ágiles remarca la conveniencia de reducir el volumen de la documentación.

ISD

(Internet-Speed Development) Es el más reciente de los métodos ágiles, surgido como respuesta para las situaciones que requieren ciclos de desarrollo muy breves con entregas rápidas.

Se centra en el talento de las personas sobre los procesos.

ISD

es un entorno de gestión orientado al negocio.

FDD

(Feature Driven Development) Prescribe un proceso iterativo de 5 pasos, con iteraciones de dos semanas.

El punto de referencia son las características que debe reunir el software, y se centra en las fases de diseño e implementación del sistema.

MÉTODOS DE PROPIEDAD COMERCIAL

MSF (Microsoft Solutions Framework)

MSF es la metodología empleada por Microsoft para el desarrollo de software.

Hasta su versión 3 (principios de 2005) MSF se definía como un marco de desarrollo flexible para adaptarse a las necesidades de cada proyecto, en oposición a lo que sería una metodología prescriptiva; porque parte de la base de que no hay una estructura de procesos óptima para las necesidades de todos los entornos de desarrollo posibles.

El marco MSF se asienta sobre unos principios fundamentales que definen la cultura de la organización:

- Fomento de la comunicación abierta.

- Trabajo en torno a una visión compartida.
- Apoderar a los integrantes del equipo (“empowerment”)
- Establecimiento de responsabilidades claras y compartidas.
- Centrar el objetivo en la entrega de valor para el negocio.
- Permanecer ágiles y esperar e cambio.
- Invertir en calidad.
- Aprender de la experiencia.

Para la aplicación de estos principios en los procesos y en las personas, MSF define un modelo de equipo y un modelo de procesos.

Sobre los modelos, y trabajando con los principios fundamentales de la cultura, las disciplinas que establece para el desarrollo del software son:

- Gestión de proyectos.
- Gestión de riesgos.
- Gestión de la mejora del talento.

Si bien, MSF despliega la gestión de proyectos y la gestión de riesgos con algunas diferencias sobre las visiones clásicas de estas áreas.

El marco de desarrollo incluye también conceptos clave, prácticas contrastadas y recomendaciones para la ejecución de las tareas concretas en el desarrollo de software.

En 2005, el desarrollo del nuevo producto de Microsoft “Visual Studio 2005 Team System” ha ganerado la evolución de MSF hacia la nueva versión 4.0 con dos líneas paralelas:

- Microsoft Solutions Framework (MSF) for Agile Software Development.
- Microsoft Solutions Framework (MSF) for CMMI Process Improvement.

RUP (Racional Unified Process)

Es un proceso de Ingeniería del Software que proporciona una visión disciplinada para la asignación de tareas y responsabilidades en las organizaciones.

RUP es un “modelo-producto” de Racional Software, integrado en su conjunto de herramientas de desarrollo, y distribuido por IBM.

RUP integra un conjunto de “buenas prácticas” para el desarrollo de software en un marco de procesos válido para un rango amplio de tipos de proyectos y organizaciones.

Las principales buenas prácticas cubiertas son:

- Desarrollo iterativo.
- Gestión de requisitos.
- Uso de arquitecturas basadas en componentes.
- Uso de técnicas de modelado visual.
- Verificación continua de la calidad.
- Gestión y control de cambios.

En su visión estática, el modelo RUP está compuesto por:

- Roles: analista de sistemas, diseñador, diseñador de pruebas, roles de gestión y roles de administración.
- Actividades: RUP determina el trabajo de cada rol a través de actividades. Cada actividad del proyecto debe tener un propósito claro, y se asigna a un rol específico. Las actividades pueden tener duración de horas o de algunos días; y son elementos base de planificación y progreso.
- Artefactos: Son los elementos de entrada y salida de las actividades. Son productos tangibles del proyecto. Las cosas que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables…)

- Disciplinas: son “contenedores” empleados para organizar las actividades del proceso. RUP comprende 6 disciplinas técnicas y 3 de soporte.
    - Técnicas: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas y desarrollo.
    - Soprte: gestión de proyecto, gestión de configuración y cambio, y entorno.
- Flujos de trabajo: son el “pegamento”de los roles, actividades, artefactos y disciplinas, y constituyen la secuencia de actividades que producen resultados visibles.

En su visión dinámica, la visión de la estructura del ciclo de vida RUP se basa en un desarrollo iterativo, jalonado por hitos para revisar el avance y planear la continuidad o los posibles cambios de rumbo.

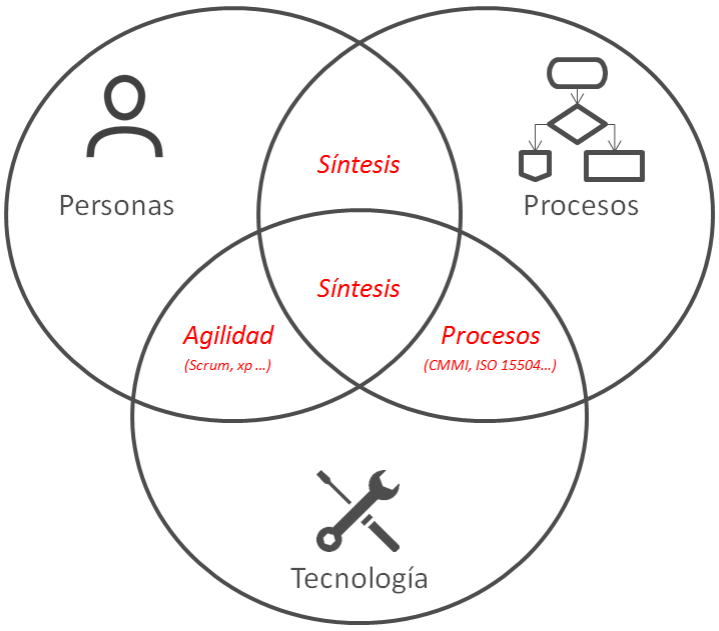
Cuatro son las fases que dividen el ciclo de vida de un proyecto RUP:

1.-Inicio. Es la fase de la idea, de la visión inicial de producto, su alcance. El esbozo de una arquitectura posible y las primeras estimaciones. Concluye con el “hito de objetivo.

2.-Elaboración. Comprende la planificación de las actividades y del equipo necesario. La especificación de las necesidades y el diseño de la arquitectura. Termina con el “hito de arquitectura”.

3.-Construcción. Desarrollo del producto hasta que se encuentra disponible para su entrega a los usuarios. Termina con el “hito del inicio de la capacidad operativa”.

4.-Transición. Traspaso del producto a los usuarios. Incluye: manufactura, envío, formación, asistencia y el mantenimiento hasta lograr la satisfacción de los usuarios. Termina con el “hito de entrega del producto”.



Aportación de **conocimiento** y **áreas** para producción basada en procesos, agilidad o en una síntesis adecuada.

Gestión predictiva y gestión ágil

Se dice que hay dos formas de abordar los proyectos:

- Dedicar escaso tiempo a la preparación y planificación, e invertir la mayor parte del tiempo en el desarrollo, probando y rectificando hasta conseguir el resultado adecuado.
- Invertir tiempo y esfuerzo en la preparación y planificación para conseguir que el desarrollo se ejecute en el menor tiempo y con los menores incidentes posibles.

Aunque esta es una visión simplificada, refleja las diferencias entre la gestión de proyectos predictiva empleada tradicionalmente por los modelos de procesos, y la gestión introducida por las metodologías ágiles; así como las razones de su desencuentro.

Los métodos formales (CMMI, ISO15504) basan el éxito del proyecto en la planificación y el rigor normativo en los procesos de su desarrollo.
Hacen especial hincapié en la importancia de los requisitos para conocer con el mayor detalle el sistema antes de comenzar su construcción, y en las prácticas formales de la gestión de proyectos, para controlar el seguiminto de la planificación y evitar desviaciones.

Por otra parte, y con mayor o menor proximidad al extremo contrario, se encuentran las denominadas metodologías ágiles, que tienen como común denominador un modelo de desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos, y predisposición para el cambio y la adaptación continua según sean las sugerencias modificaciones propuestas por los usuarios durante el desarrollo. Además también trabajan con menor rigidez en los procesos, aunque las diferencias en este punto son significativas entre cada método.

Considerar que hay dos formas de realizar los proyectos es una visión simplificada, pero refleja el hecho de que en la realidad, cada técnica se define en uno u otro bando con mayor o menor proximidad de los extremos.

La base de conocimiento gestión de proyetos predictiva más conocida es PMBOK® (Project Management Body of Knowledge) desarrollada por PMI (Project Management Institute)

Características de los proyectos de software.

El concepto sistema de software es muy amplio, y tan sistema de software es el integrado en un teléfono celular, como el diseñado para asistir el pilotaje de una aeronave.

En algunos casos se tratará de gestionar una normativa contable u otro negocio o entorno suficientemente conocido, en el que un proceso formal de requisitos inicial puede capturar con bastante detalle todas las funcionalidades esperadas. En otros, la novedad o complejidad del entorno en el que se integrará el sistema, hacen difícil, si no imposible descubrir qué debe hacer el software si no es a través de prototipado o desarrollo evolutivo.

El tamaño también importa. Un sistema puede requerir el esfuerzo de un programador durante un par de meses, o de un equipo de decenas de personas durante meses o años.

Por las características del proyecto quizá la funcionalidad alcanzada en cada versión, así como la precisión en el cumplimiento de las fechas puede ser poco trascendente o vital para el negocio del cliente.

En este aspecto, la diferencia de criticidad o nivel de integridad entre unos sistemas y otros es muy diferente, y determina el rigor de los procesos que deben aplicarse en su desarrollo. Un error en el aplicativo de una entidad bancaria puede suponer pérdidas económicas o de imagen que comprometan la continuidad de su negocio. Errores en otros sistemas pueden producir pérdidas humanas. La presencia de errores en la página web de un pequeño comercio tiene repercusiones de menor alcance.

Visión, misión y negocio de la organización.

Algunas preguntas relativas a la organización que ayudan a diseñar el aludido equilibrio son:

- ¿Para qué desarrolla software la organización?
- ¿Es una empresa con un producto definido?
- ¿Es el departamento de sistemas de una entidad bancaria que centra su misión en la seguridad?
- Para alcanzar su misión, ¿Qué papel juega la innovación y vanguardia tecnológica?
- ¿Se dedican a mantener operativos sistemas ya desarrollados?
- ¿Programa sistemas poco innovadores, sobre plataformas contrastadas, para negocios y entornos conocidos?
- ¿Apuesta por la innovación, implementación sobre dispositivos nuevos, o diseñan soluciones para entornos novedosos?.

Cultura de la organización

- ¿Se trata de una organización con niveles jerárquicos y funciones claramente definidos?
- ¿Cuáles son los niveles de confianza, delegación, empowerment y responsabilidad?
- ¿Clima laboral, motivación?

Diseño y gestión del equilibrio productivo personas-procesos-tecnología

La estrategia productiva.

¿Qué tecnología emplea para el desarrollo?. Equipos, red, sistemas de pruebas, plataformas operativas, plataformas de desarrollo, pruebas, herramientas CAD…?

¿Qué porcentaje o peso del conocimiento necesario para la ejecución de los proyectos lo garantiza la ejecución de los procesos, y cuánto las personas?.

¿Son coherentes las políticas o procesos de formación, contratación, planes de carrera, diseño del modelo de procesos, calidad y mejora…

Evitar la gestión lineal

Para todo problema complejo hay siempre una solución simple, que es errónea”
George Bernard Shaw

Para diseñar y gestionar entornos de producción eficientes se debe trabajar con un enfoque sistémico que contemple todos los factores implicados en la producción de software, con la perspectiva de un sistema interrelacionado, imbricado y alineado con la misión de la organización.
Se deben evitar estrategias simples que azuzadas por los problemas del día a día busquen relaciones lineales causa–efecto, sin apreciar la relación y armonía del sistema, que las integra y les da sentido, y que no se ciñe al área de desarrollo, sino a la organización en su conjunto.
No se trata por tanto de conocer cuál es la mejor metodología para el diseño de bases de datos, cuál el mejor modelo de procesos o cuáles las mejores técnicas de gestión de recursos humanos.
Sin esa perspectiva se trabaja con una visión miope, y la realidad se presenta como un campo de batalla desordenado que requiere actuaciones en cada uno de los frentes que presenta.
Si los proyectos se retrasan será preciso sugerir, animar o presionar a las personas para que alarguen sus jornadas de trabajo, o restrinjan los tiempos de café…
Si programamos con mayores costes que nuestra competencia, habrá que contener los salarios, o contratar más barato, o aplicar procesos de calidad para reducir los tiempos, o…
Para cada problema se abre el botiquín, y se aplica un apaciguador de síntomas. Las soluciones obvias no funcionan.
En el mejor de los casos introducen mejoras en el corto plazo que terminarán por empeorar la situación.
La presión del día a día ofrece dos atajos tentadores que se deben evitar:

- Aplicar soluciones enfocadas sólo en el corto plazo.
- Obtener mediciones y datos que aparenten un buen comportamiento.

Con carácter general, las soluciones a corto hipotecan el futuro. son sintomáticas, y generan resistencia, de forma que la próxima vez que se vuelva a generar un problema similar, el remedio sintomático tendrá menos eficacia.

Cerrar rápido un proyecto, hará más duro su mantenimiento (sistema peor diseñado, mayor densidad de errores…).

Presionar a los programadores para cumplir fechas es también una solución a corto, que no ofrece soluciones a largo. Si se mantiene la presión como una norma, generará desmotivación y degradación en la calidad de lo producido.

Las mediciones sobre los procesos son una herramienta útil para comprender el funcionamiento general del sistema y trabajar en su mejora. Emplearlas como justificación ante instancias superiores, es también una búsqueda de la solución a corto, interesada en que las cosas pinten bien. De esta forma las organizaciones pueden llegar a agonizar con una imagen de salud envidiable.

Gestión sistémica

La eficiencia es el resultado de la idoneidad y equilibrio de todos los componentes de la producción.

¿Cuál es el mejor modelo de procesos para el desarrollo de software?.
¿La cultura de empresa más adecuada?
¿Las mejores técnicas de gestión de RR.HH.?
¿Las mejores plataformas de progrmación?
Para la gestión de las personas, los procesos y la tecnología, no hay métodos simples ni absolutos con resultados inmediatos.

Un modelo de procesos CMMI con nivel 5 de madurez puede garantizar resultados en línea con la misión de una empresa integradora de grandes sistemas críticos, pero resulta excesivamente pesado para una start-up de diseño web

El modelo “bueno” no es Scrum, o CMMI o XP, sino el que mejor encaje en su sistema.

Las técnicas empleadas, los modelos de calidad, la tecnología, la cultura de la empresa, la gestión de las personas… deben mantener coherencia, de forma que se potencien, y no se contrarresten.

Un “know-how” adecuado, basado en el conocimiento tácito de las personas no funciona con equipos desmotivados.

Un método ágil puede resultar idóneo para el tamaño de equipos y de empresa, pero si sus procesos no cubren las facetas contractuales de la adquisición, generará problemas en la validación del producto.

A modo de síntesis, las claves para conseguir organizaciones eficientes de desarrollo de software son:

Personalidad de la organización. Esta es la referencia final con la que deben alinearse y sincronizarse todas las variables que operan juntas para lograr los objetivos. Cuanto más nítida sea la visión, misión, estrategia, segmento y objetivos de la organización, con mayor tino y precisión se podrán imbricar los componentes del sistema y orientar la gestión de su funcionamiento.

Conocimiento de la propia empresa. Sus objetivos, debilidades y fortalezas. Relevancia del capital estructural y del capital humano. Cuál es su configuración actual y cuál la óptima para la personalidad de la organización.

Este es el conocimiento que orientará el diseño y la gestión del sistema de desarrollo (en realidad de todos los subsistemas de la organización).

Conocimiento de la industria. Características de la materia prima: el software. Las áreas de conocimiento implicadas en su desarrollo, las técnicas de construcción, los métodos y procesos posibles para su desarrollo y mantenimiento (ISO, CMM, Métodos Ágiles…)

Es la información de referencia, con el conocimiento acumulado por nuestra industria. Su comprensión y visión general ayuda a seleccionar las prácticas más adecuadas para la organización, o da el conocimiento de causa necesario para el diseño de modelos propios.

Gestión sistémica. Guiar las decisiones de gestión desde la perspectiva general del sistema que compone la organización. Huir de las soluciones sintomáticas, y evaluar sus idoneidad más allá del corto plazo, y su coherencia con el sistema en su conjunto.

Revisión y adaptación. El mercado, el entorno tecnológico, la misma base de conocimiento de nuestra industria están en continua evolución. Es necesaria una tarea de vigilancia del entorno para investigar, desarrollar e innovar, tanto en productos como en los propios procesos y formas de trabajo.

Bibliografía

IKUJIRO NONAKA & HIROTAKA TAKEUCHI, New New Product Development Game, Harvard Business Review Jan, 1 1986
JENNIFER STAPLETON, DSDM Business Focused Development, DSDM Consortium, 2003
KEN SCHWABE y MIKE BEEDLE, Agile Software Development with Scrum, Prentice Hall, 2002
JIM HIGHSMITH, Agile Project Management, Addison Wesley, 2004.
KEN SCHABER, Agile Project Management with Scrum, Microsoft, 2004
GARY POLLICE, LIZ AUGUSTINE, CHRIS LOWE y JAS MADHUR, Software Development for Small Teams a RUP-Centric Approach, Addison Wesley, 2004
BARRY BOEHM y RICHARD TURNER, Balancing Agility and Discipline, Addison Wesley, 2004
GARY FORD y NORMAN GIBBS, Mature Profession of Software Engineering, SEI, 1996
ROBERT L. GLASS, Facts and Fallacies of Software Engineering, Addison Wesley, 2002
CARNEGIE-MELLON UNIVERSITY CMMI for Software Engineering, 1.1, 2002
IEEE COMPUTER SOCIETY Guide to the Software Engineering Body of Knowledge, 2004
DENNIS M. AHERN, AARON CLOUSE y RICHARD TURNER, CMMI Distilled: A Practical Introduction to Integrated Process Improvement, Addison Wesley, 2003
MARY BETH CHRISSIS, MIKE KONRAD y SANDY SHRUMI, CMMI Guidelines for Process Integration and Product, Addison Wesley, 2003
HAN VAN LOON, Process Assessment and ISO/IEC 15504 A Reference Book, Springer, 2004

Autor y derechos