Gestión y procesos en empresas de software

Juan Palacio

Rev. Nov - 2005

SINOPSIS

Las empresas que desarrollan software no pueden ignorar que su negocio es un negocio de software, y que el modelo que cada una adopte para las actividades de desarrollo y mantenimiento tiene implicaciones relevantes en la eficiencia general del negocio.

El problema que pueden encontrar quienes deciden implantar métodos más eficientes, es caer en la desorientación ante el abanico de modelos de calidad, de procesos y de técnicas de trabajo desplegado en la última década; o abrazar al primero que se presenta en la puerta de la organización como "solución" de eficiencia y calidad

Este artículo dibuja un mapa de situación general como orientación para directores técnicos, responsables de departamentos informáticos y personal directivo de organizaciones de software.

La visión general que ofrece, muestra cómo en su totalidad un entorno de producción es un entorno sistémico, en cuyo diseño global los componentes tienen que encajar y funcionar armónicamente, alineados con las características, cultura y estrategia de la organización.

Amenaza u oportunidad

Según como afrontan las organizaciones el desarrollo del software, éste puede comportarse como factor de riesgo o amenaza para el negocio; o por el contrario como una poderosa oportunidad de negocio.
Todas las empresas quieren producir más rápido, mejor y con menores costes, y sin duda esto es posible porque la naturaleza del software no es origen de riesgos y problemas, sino una fuente de oportunidades.

Cada vez más directivos están comprendiendo que la forma de gestión del desarrollo de software, puede hacer de la materia prima de su negocio un material arisco de resultados impredecibles, o una ventaja competitiva.

La evolución hacia entornos de ingeniería del software requiere cambios severos en la organización, así como el convencimiento, implicación y empuje de la dirección. Pero sobre todo el diseño de un modelo de producción propio que sepa aprove-char la personalidad de la organización, y responder a las particularidades de su negocio.

El software ha estado generando los mismos problemas en los últimos 40 años, y quien no cambia la forma de hacer las cosas, sigue tropezando en ellos.

El problema que pueden encontrar quienes deciden implantar métodos más eficientes es caer en la desorientación ante el abanico de modelos de calidad, de procesos y de técnicas de trabajo desplegado en la última década, o abrazar al primero que se presenta en la puerta de la organización como "solución" de eficiencia y calidad.

Trabajar sin ningún método es una opción, pero no una metodología

Las organizaciones que desarrollan o mantienen software pueden optar por trabajar "a la buena de Dios", o por seguir una metodología. Hacerlo a la buena de Dios no es tan raro. Es la primera forma que se empleó para desarrollar programas:
"Aquí tenemos unos ordenadores, aquí unos señores a los que les encanta leer los manuales de programación, y se trata sencillamente de conseguir que estas máquinas terminen imprimiendo facturas (o haciendo lo que sea)".

En realidad, cuando en la segunda mitad del siglo pasado la industria del hardware puso en la calle máquinas que se podían programar, poco más se podía hacer; y los pioneros de nuestra profesión, atraídos por el encanto de diseñar y construir artefactos útiles, y verlos funcionar, fueron los primeros en remangarse frente al teclado y decir a su cliente con una sonrisa:"en un par de meses esto estará funcionando".
Fueron también los primeros en pasar noches en vela programando, prometiendo una y otra vez que "tan sólo es cuestión de un par de días más."

Aunque las cinco décadas de vida del software ya han sido suficientes para experimentar los excesos y los errores de la infancia y la juventud, la resistencia al cambio es el mejor aliado de la inercia, y por eso se produce una cierta impermeabilidad a la experiencia.

En cualquier caso es una opción: trabajar sin ninguna metodología.

SEI, (Software Engineering Institute) por el rigor de su documentación y la circunspección de su imagen no puede llamar a esta forma de producir como "a la buena de Dios", y en su lugar afirma que es la forma propia de organizaciones "poco maduras".
En la escala que de 1 a 5 emplea para determinar el grado de madurez de una organización, y en consecuencia el nivel de garantía que ofrece en cuanto a calidad, predictibilidad en los resultados y eficiencia

en el desarrollo de software, este tipo de organizaciones se quedan, lógicamente, en el 1.
Que, ¡ojo cuidado! no quiere decir que necesariamente van a producir mal software, de forma ineficiente y con retraso, sino que las probabilidades de cumplir las tres facetas es baja.
Hay equipos que lo consiguen, pero son pocos. La razón es sencilla, los resultados en estos casos son tan buenos como las personas que los desarrollan, y lo cierto es que los buenos programadores escasean.
A este modo de producción se le ha venido a llamar "programación heroica", porque todo el peso del resultado descansa sobre el "saber-hacer" de los programadores.
El éxito o fracaso de las organizaciones que trabajan sin metodologías depende del conocimiento tácito de su personal, pero teniendo en cuenta que se trata del conocimiento que cada uno traía ya de la calle, o del que adquiere *motu proprio*, porque estamos hablando de "ninguna metodología", lo que implica que como mucho los procesos de formación de la empresa llegan al "ahí tienes manuales y libros, por si hubiera algo que no sabes".

Este es sin duda el modelo con el que empiezan muchas empresas "start-up": un equipo de emprendedores con talento, capaces de construir sistemas de software con calidad.
Los resultados serán tan buenos como ellos los 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.
Y esta situación dibuja el guión común a todas las pequeñas empresas de desarrollo de soft que surgen de cero, impulsadas tan sólo por el empuje de sus emprendedores: pueden llegar todo lo lejos que la combinación del talento y de la capacidad de marketing de sus creadores les permitan.
O sea, en ocasiones acabarán cerrando, y en otras alcanzarán un nivel de estabilidad tan mediocre o tan brillante como dicha combinación les permita. Y si una vez logrado ese nivel de equilibrio desean proyectar mayor crecimiento a su organización se enfrentan a un reto importante:
pasar de personas que saben hacer software a empresa que sabe hacer software. Salto que supone que no van a ser ya ellos, sino su empresa quien deberá producir de forma eficiente y repetible en todos sus proyectos, resultados de calidad. Y esta es otra aventura en la que muchos fracasan teniendo que regresar a la casilla de salida, o si el "roto" del intento ha sido muy grave, terminar cerrando, o como mal menor dejándose adquirir por algún competidor más o menos grande del sector.

La teoría de la gestión empresarial recomienda para esta travesía tomar como medio de transporte la gestión por procesos.

Sí, no es nada nuevo, y leerlo a estas alturas puede dar la sensación de estar desayunando con el periódico del día anterior.
Aparte de las excelencias recogidas en el manual del consultor sobre la transversalidad de los procesos, que atravesando a toda la organización dibuja modelos organizacionales sin rigideces departamentales, y alinea con sus flujogramas a toda la empresa en el mismo sentido con visión centrada en el cliente, y bla, bla, bla.
Lo cierto es que la gestión por procesos encierra cuatro factores que hacen posible pasar de grupo de emprendedores a empresa:

- **Repetibilidad de resultados**. Al conseguir que la calidad del resultado sea consecuencia del proceso, producir aplicando el mismo proceso garantiza la homogeneidad de los resultados.
- **Escalabilidad**. Es una consecuencia de la repetibilidad. No sólo un equipo consigue resultados homogéneos en todos los proyectos, sino que los obtienen todos los equipos.
- **Mejora continua**. Al aplicar meta-procesos que trabajan sobre los propios procesos de producción, midiendo y analizando los resultados se obtienen los criterios de gestión necesarios para aplicar medidas que mejoran de forma continua la eficiencia y calidad de los procesos base, y por tanto de los resultados.
- Un **know-how propio**, consiguiendo finalmente una empresa que sabe hacer, porque su modelo de procesos termina conteniendo un activo valioso de la organización: la clave para hacer las cosas bien, con eficiencia y de forma homogénea.

No sólo son procesos

Los procesos marcan pautas para realizar el trabajo, pero sin las personas y las herramientas (tecnología) necesarias, lo que se puede producir con ellos es más bien poco.
Las únicas combinaciones válidas para formar sistemas capaces de producir resultados son:

Personas + tecnología	Producción heroica
Personas + procesos + tecnología	Producción basada en procesos

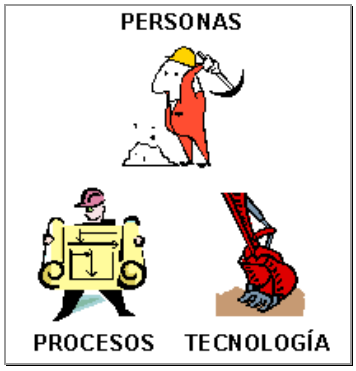

Figura 1

Un ejemplo que seguro todos hemos realizado alguna ves, y que ilustra las diferencias entre ambos sistemas de producción es el montaje de un mueble en *kit*.

Para enfrentarnos a esta tarea tenemos, por un lado a las personas: nosotros. Y por otro la tecnología: el destornillador (manual o eléctrico), los tornillos, pasadores, la cola de montaje, etc. Y si dejamos el proceso a un lado (el manual con las instrucciones de cómo proceder) tendremos un sistema perfecto de producción heroica: personas + tecnología; del que cabrá esperar los resultados propios de estos entornos.

La productividad y la calidad no son homogéneas. Hay quien ensambla la estantería o el mueble, en cuestión de minutos, y quien necesita toda una tarde.
Algunos obtienen estanterías robustas y perfectas, y otros terminan el montaje con peor fortuna, afirmando que la estantería está bien montada pero que las piezas venían mal cortadas de fábrica, o que quien diseñó ese mueble era un perfecto inútil, o que los tornillos son de mala calidad, o quizá sin ni siquiera percatarse de los defectos con los que ha dejado el mueble.

En definitiva esta forma de trabajo produce resultados, y puede resultar más o menos apropiada para realizar hobbys o bricolajes domésticos, pero para una empresa de montaje de muebles no resultaría viable un sistema de producción que no pudiera garantizar eficiencia en los costes, y resultados con una calidad homogénea.

Hay dos formas de evitar estos problemas:

- Introducir un tercer elemento en el sistema de producción: los procesos.
- Trabajar sólo con personas y tecnología, pero empleando sólo a los mejores ebanistas, y proporcionándoles las mejores herramientas.

Como veremos a continuación generalmente la mejor solución no suele consistir en adoptar una u otra opción, sino una mezcla de ambas, con mayor o menor peso específico en una u otra, en función de las características del negocio.

Al añadir procesos al sistema se consigue que todas las personas ejecuten el trabajo siguiendo las mismas pautas, y que los parámetros de ejecución y los resultados puedan ser medidos de forma homogénea en toda la organización.
De esta forma se reduce drásticamente la hetero-geneidad de los resultados, tanto en la eficiencia de los tiempos invertidos, como en la calidad obtenida, que empieza a ser proporcional a la capacidad de los procesos que se hayan diseñado.

Pero no hay que dejarse deslumbrar por los procesos y olvidarse de los otros componentes.
Basar una empresa sobre un sistema de producción de dos elementos: personas y tecnología plantea limitaciones; pero basarlo en un entorno exclusivo de procesos, o de procesos y tecnología, simplemente no es posible.

La realidad de los procesos es cierta, pero en el triángulo personas - procesos - tecnología cada elemento actúa con un peso específico, diferente en función del tipo de producción, e incluso de las características de personalidad de cada empresa.
Por eso, llegados a este punto, el ámbito de la teoría generalista se acaba, y cada sector y cada empresa, más que importar una metodología estándar sacada del manual del perfecto consultor, debe comenzar por el *gnosce te ipsum*, y a partir de ahí analizar y descubrir la potencia de cada uno de los tres elementos de la producción para componer el diseño de un sistema de producción a su medida.
Los gestores o consultores "estándar" no existen; bueno, perdón, sí que existen, pero obtienen los resultados estándar que todos conocemos.

Relevancia del capital estructural y del capital humano

El capital estructural está formado por los bienes que quedan en la empresa cuando las personas se van a sus casas: patentes, licencias, cartera de clientes, equipos, maquinaria, vehículos, etc.
El capital humano es el compuesto por el valor que la empresa emplea en su negocio y que resulta inseparable de las personas.

Todas las industrias necesitan ambos tipos de capitales para elaborar los productos o servicios de su negocio, pero la relevancia con la que cada uno de ellos puede influir en el resultado final es muy diferente de unas empresas a otras.
Es frecuente que para las empresas dedicadas a la fabricación de productos el componente estructural sea más crítico que para las dedicadas a los servicios, en las que el elemento humano tiene más relevancia.
Pero este no es un principio universal, y dentro de la misma industria o del mismo sector, la estrategia y la táctica de cada empresa también influyen en los niveles

de relevancia que van a tener el capital humano y el estructural.

Veámoslo con un ejemplo, comparando la relevancia de ambos en dos empresas del mismo sector: hostelería.
Por un lado pensemos en un exquisito restaurante de cocina vasca (por ejemplo), y por el otro en un establecimiento de *PhonoPizza* (también por ejemplo).
Al margen de consideraciones gastronómicas, ambos negocios tienen perfectamente claros su identidad, personalidad y segmento; ambos tienen también su propio patrón de atributos de calidad sobre el que medirse.

En el segundo modelo la empresa tiene una serie de procedimientos para garantizar de forma continua la calidad de sus resultados (calidad y repetibilidad), por lo que éstos dependen mucho menos del conocimiento tácito de sus cocineros, y en su mayor parte se deben a los procesos y la tecnología empleada.

Los hornos regulan automáticamente el tiempo y la temperatura, los ingredientes se producen también en base a unos procesos, y se distribuyen de forma masiva a todos los establecimientos en estuches, ambientes frigoríficos y medios de transporte en base a unos procesos que son los principales responsables de los resultados.

Por esta razón resulta mucho más fácil, o en lenguaje empresarial, menos costoso, reemplazar al personal de cocina de un establecimiento de *PhonoPizza* que al de un restaurante de cocina vasca.

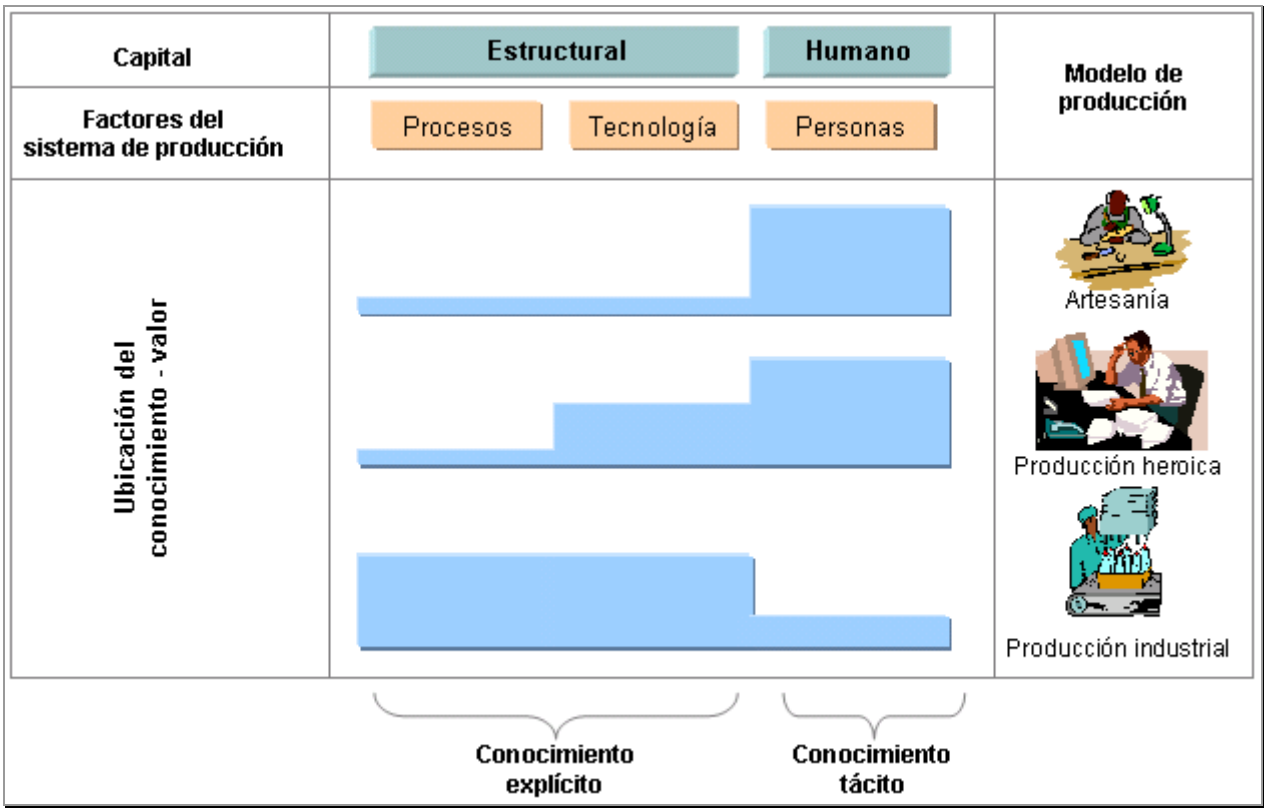

Figura 2

Las barras de color azul de la figura 2 representan de forma gráfica el valor, que para una empresa determinada, aporta cada elemento a su sistema de producción.

Las combinaciones posibles para cada negocio son muchas, y factores como la innovación en la tecnología, en los procesos o la formación del personal pueden abrir la horquilla de potencial de cada elemento.

La homogeneidad y calidad de los resultados, y la eficiencia del sistema en su conjunto serán consecuencia del equilibrio logrado.

Por supuesto una franquicia de hamburguesas y comida rápida podría incluir, además de tecnología y procesos eficientes, la contratación o formación de gourmets de alta cocina.; pero de esta forma su sistema de producción no estaría aprovechando la naturaleza de su producto como ventaja competitiva.

La obsesión por la excelencia mal entendida, o la desorientación sobre las funciones del propio puesto lleva a muchos gestores a centrar sus esfuerzos en la incorporación de modelos o técnicas, bien de procesos, bien de recursos humanos o de innovación tecnológica; o de todos a la vez, por razones como que "es lo último en gestión", o lo estudiado en el último seminario, o lo que con tanta vehemencia le ha "vendido" el último pseudo-consultor que visitó su empresa, o lo leído en ese libro de gestión tan interesante.

Esta es al fin y al cabo, la razón del escepticismo y las críticas que finalmente verterán sobre modelos, métodos o técnicas contrastadas y eficaces; cuya debilidad no radica en ellas sino en una interpretación incorrecta, o en una implementación en un grado incorrecto, o sobre sistemas para los que no resultan apropiadas.

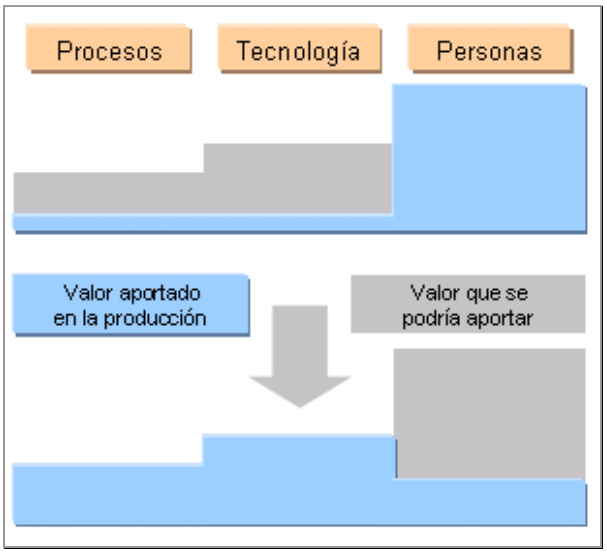

Figura 3

Conseguir que el sistema de procesos, tecnología y personas formen una ventaja competitiva frente a la competencia, y no un reto más del negocio, no es fácil, ni puede importarse con la implantación "prêt-à porter" de un modelo estándar.

Como refleja la figura 3, la clave del éxito radica en conseguir que cada factor pueda aportar el mayor valor hasta el límite de su mejor relación eficiencia / calidad.

En esta tarea nunca está dicha la última palabra, y la labor de innovación constante en procesos y tecnología puede ir ampliando los límites de valor a un factor para conseguir un nuevo equilibrio con mejores parámetros de eficiencia y calidad en el sistema.

Gnosce te ipsum

"No hay mayor inmoralidad que ejercer un oficio que no se conoce"
Napoleón

Lo cierto es que un gestor o un directivo necesita combinar los conocimientos de *management* con los del propio negocio, sector e industria en la que se mueve.
Quien se sienta en el puente de mando de una embarcación, por su puesto debe ser un buen "gestor" y contar con las habilidades y conocimientos para gobernar una nave, pero para realizar la travesía necesitará conocer las características de la propia nave, de la ruta y de las condiciones de la mar.

Muchas veces los problemas que desde los puestos directivos se transmiten a la organización tienen su origen en la carencia de alguna de estas habilidades:

- Decisiones más o menos fundadas en principios de gestión, que han ignorado las características de la empresa, o las particularidades del sector.
- Decisiones de conocedores del negocio sin experiencia en management.

En cierto modo es como poner a los mandos de una embarcación a un capitán que desconoce cómo es el barco que tripula y las características de la ruta que va a seguir; o como apostar al timón a un tripulante veterano del navío, sin conocimientos de pilotaje.

Según las características y dimensiones de la embarcación, y las vicisitudes que presente la travesía, alcanzar puerto puede ser un reto sin solución.

Conocer la empresa

Los aspectos de la empresa que se deben conocer son, en general, comunes en todos los sectores, y poco puede aportar este artículo a un gestor, que él no conozca ya:
Para conseguir eficiencia en su estrategia de producción esta deberá ser consecuente y estar alineada Con:

La visión: Qué quiere llegar a ser la organización y cuál es su meta.
Estrategia:, foco y plan de negocio, contemplando el mercado, las fortalezas y debilidades propias, etc. Cuál es la ruta que se ha perfilado para alcanzar esa meta.
Sin saber adónde se va, difícilmente se puede diseñar la estrategia para lograr el objetivo, y sin estrategia también es difícil que un gestor pueda ejecutar la táctica adecuada, que es la que deberá incluir, entre otras cosas, el modelo de producción empleado.

Por supuesto se puede zarpar y navegar a la deriva para ver en qué puerto se acaba, pero si no se es amante de las emociones fuertes; y si se siente respeto por el capital de los accionistas, y el trabajo de toda la tripulación, es mejor considerar que la lotería no es una inversión, sino un juego de azar.

Industria del software

Para todas las industrias hay marcos de trabajo compuestos por normativas y estándares más o menos estables y consensuados, que en forma de modelos ayudan a mejora o evaluar la calidad de sus sistemas de producción.
Al mismo tiempo cada ingeniería tiene delimitadas sus áreas de conocimiento, y reguladas las técnicas de trabajo para ofrecer las garantías necesarias en la construcción de sus respectivos artefactos.
Así por ejemplo, una empresa de arquitectura, o de ingeniería aero-espacial, naval o nuclear . tiene estándares y conocimientos estables, que si aplica sistemáticamente aportan garantías contrastadas a la robustez de las estructuras de sus edificios o de las naves que construye.

Estos conocimientos, que sirven de referencia y ayuda a los entornos de desarrollo, provienen de dos áreas:
1.- Currículo de técnicas y conocimientos de las respectivas ingenierías.
2.- Modelos, normas y estándares de calidad y procesos aplicables a cada industria.

Una diferencia importante entre la industria del software y otras industrias de ingeniería es la falta de consenso o inmadurez en los dos puntos anteriores, que deja a las empresas de nuestro sector en un cierto grado de orfandad a la hora de buscar ayuda en modelos, técnicas o patrones de referencia.

Por la relativa juventud del software en nuestra industria no hay un consenso acerca de cómo éste dee ser producido. El siguiente texto extraído del artículo "Criticism of software engineering" de wikipedia expone con acierto 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.

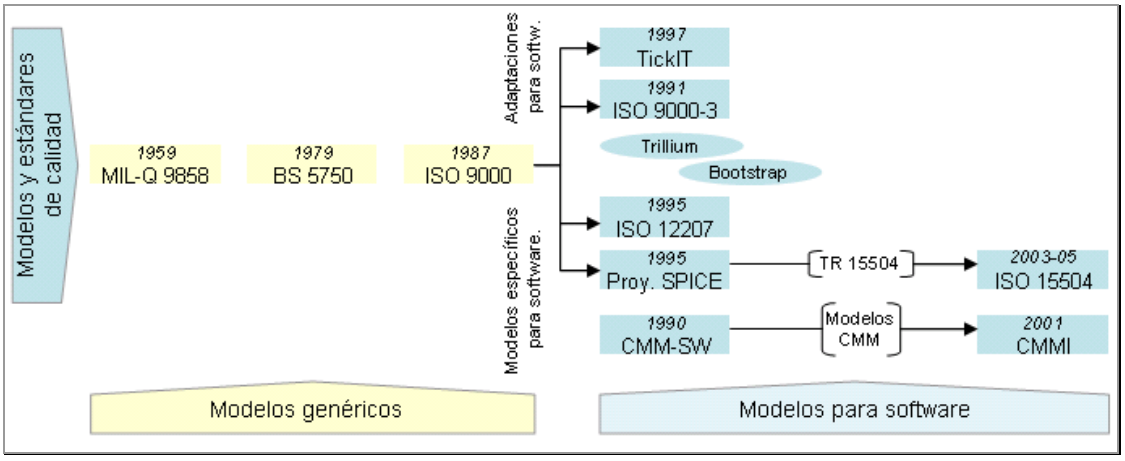

Fig. 4

Modelos y estándares de calidad

La figura 4 es un mapa de situación simplificado con la foto de los principales marcos y modelos de procesos que pueden servir de referencia a las organizaciones de software; y las principales referencias de sus orígenes y evolución.

Los procesos de fabricación de los años 50 hicieron necesaria la normalización de los procesos de producción para garantizar la consistencia de los resultados sobre unos parámetros o requisitos previamente determinados.

La norma que se considera como punto de arranque de los posteriores estándares, marcos y modelos que se han extendido a todas las industrias es la desarrollada por EE.UU. en 1959: "Quality Program Requirements" MIL-Q-9858, que, inicialmente diseñada para el ámbito militar, 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, pero de forma paralela diferentes países y 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 en realidad la precursora de ISO 9000.

Las normas y estándares que fueron surgiendo de los años 60 a los 90 dieron respuesta a las garantías de homogeneidad, calidad y predectibilidad que los entornos de fabricación, y especialmente el desarrollo de sistemas complejos requería al agrupar en proyectos la construcción de artefactos que imbricaban tecnologías e ingenierías diferentes: mecánica, aero-espacial, naval, atómica, etc.
En estos sistemas se requería la integración, cada vez con mayor protagonismo, de una ingeniería que en base al conocimiento que había desplegado en esos años, no podía considerarse aún ni asentada ni consensuada: la ingeniería del software.
Algunos de los fiascos que se produjeron en determinados sistemas por la introducción del software sin que éste pudiera dar garantías de resultados a la altura de ingenierías más maduras, son ya parte de la historia de nuestra profesión.

Esbozar este marco histórico resulta necesario, porque es la causa de la actual sopa de letras capaz de marear al gestor más dispuesto con sólo ennumerarla: ISO 900-3, TicIT, ISO 12207, CMM-SW, ISO 15504, CMMI, ASD, XP, SCRUM, DSDM, PP, MSF…

Tomando un poco de perspectiva (tan sólo 3 décadas), al software se le ha juntado la necesidad de encajar en marcos de calidad para ofrecer garantías de perfección, eficiencia y consistencia en los resultados; con la creación de su propia base de conocimientos técnicos para desarrollar y mantener software (requisitos, codificación, pruebas, diseño, etc.)

En este punto es interesante diferenciar entre marco o modelo de procesos y técnica o base de conocimiento técnico.
La primera línea tiene como objetivo fijar **qué** es lo que hay que hacer para ofrecer garantías en los proyectos, y la segunda **cómo** debe realizarse.

Un marco de procesos determinará, por ejemplo, la necesidad de gestionar adecuadamente las modif.-caciones 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 el marco de procesos no dirá, por ejemplo, que deba emplearse un sistema basado en documentos o en base de datos para la gestión de los requisitos; o si el diseño 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 dar la cobertura necesaria a las particularidades de nuestro producto, donde 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. CMM, y las normalizaciones de ISO 12207 y 15504.

CMM

SEI (Software Engineering Institute), un auténtico peso pesado en la normalización de los procesos del software, desarrolló su línea de trabajo sobre el concepto de "madurez" de las organizaciones para producir software.
Por madurez se entiende la capacidad que tiene la organización para asegurar la calidad de sus proyectos (fecha, coste y funcionalidad), la homogeneidad (siempre y en todos sus proyectos) y la capacidad de aprendizaje de sus 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 ha demostrado que en muchas organizaciones ha resultado eficaz.
Este modelo crea una escala de cinco niveles para determinar la madurez de una organización.

- Nivel 1.- INICIAL
  Pertenecen a él las empresas que no basan el desarrollo en procesos definidos, y no aplican técnicas de gestión de proyectos.
  El modelo se refiere a entornos de producción caóticos con técnicas de programación heroica cuyos resultados no son predecibles y 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 la capacidad de los procesos de desarrollo, de forma que pueden predecir de forma cuantificable 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, surgieron otros para la medición y mejora de la capacidad de los procesos, todos ellos centrados en áreas relacionadas 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 el modelo SW-CMM se actualizó en otro que facilitaba su implantación de forma conjunta y simultánea con otros modelos (SE-CMM o IPD-CMM). Fue el nuevo modelo 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:
CMMI al igual que CMM tiene dos utilidades. Puede servir tanto como guía para la mejora en una organización, o como criterio para evaluar su nivel; 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 las mediciones de los procesos.

De esta forma los modelos CMMI presentan 2 versiones:

- Versión escalonada. Guía a la organización sobre 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 su nivel de capacidad.

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 cada una de las áreas de proceso contempladas por el modelo.

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 el estándar internacional que ha determinado cuál es el ciclo de vida de un proyecto de software, y cuáles los procesos y tareas que intervienen en cada una de sus fases.

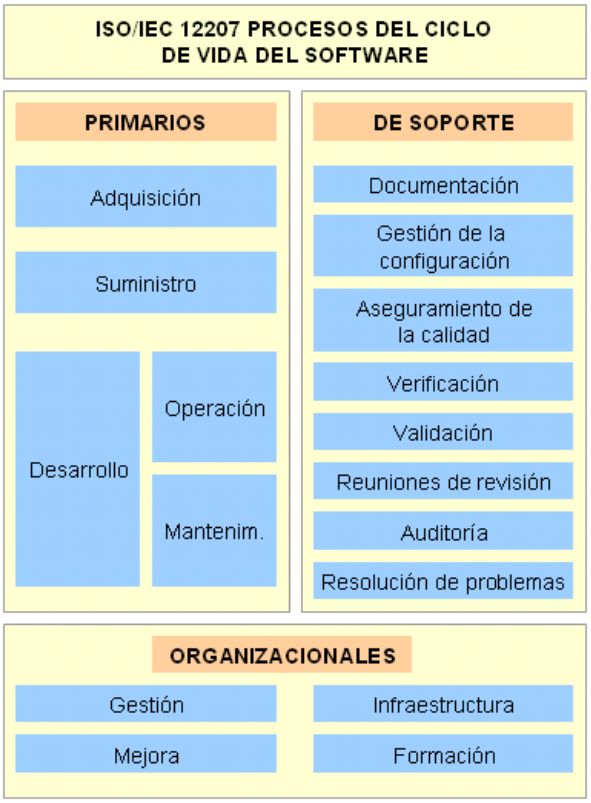

Figura 5

A grandes rasgos, 12207 concluyó considerando que el ciclo de vida de un sistema de software comienza en el momento que se concibe su idea o necesidad.

Momento en el que ya es necesario comenzar a actuar de manera ortodoxa para describir el ámbito del problema, las soluciones posibles, etc.

El ciclo de vida comprende también 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 objetivo. 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 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.

Modelos nuevos sobre una profesión recién estrenada

El uso de modelos de procesos y calidad para asegurar la eficiencia, aptitud y consistencia en los resultados, y para asentar pautas de mejora continua, es relativamente novedoso, y lo habitual es aplicar estos marcos sobre entornos de fabricación, con conocimientos profesionales ya maduros.

El software, sin embargo, es una industria que trabaja con un conocimiento profesional escasamente asentado. Y a la aparición de marcos y modelos de calidad se suma el movimiento paralelo de propuestas de currículos profesionales.

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.

Con este esquema se podrá estar más o menos de acuerdo, pero es el que emplea la sociedad para dar el rango de profesión a los trabajos no regulados. Para cambiar el título de curandero por el de médico, el de sacamuelas por odontólogo, o el de alquimista por el de químico.

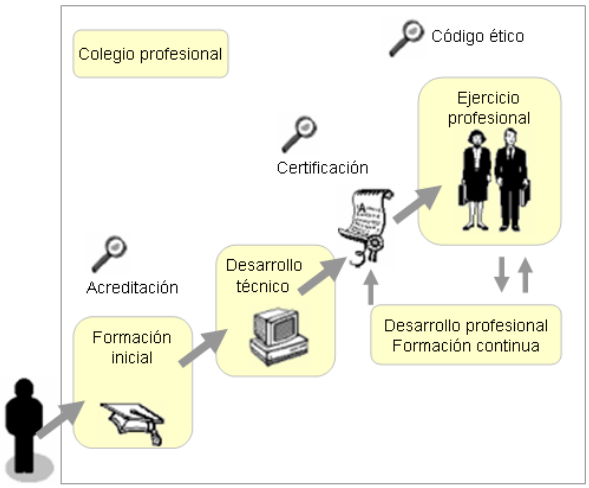

Fig. 6
(Gary Ford y Norman Gibbs 1996)

En esta línea han estado trabajando un buen número de organizaciones profesionales, agrupados 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 lo tierna que está la profesión. O mejor dicho, de que se trata de una profesión no asentada.

La heterodoxia: métodos ágiles

Ya hemos visto que a la relativa novedad de gestión por procesos, en nuestra industria se suma como dificultad adicional el producirse de forma simultánea al asentamiento de la profesión.

Y para conseguir el "más difícil todavía", hay que contar con una tercera tendencia de normalización. O, quizá habría que llamarla de des-normalización:

Los métodos ágiles.

A finales de los 90, reputados profesionales con renombre y eco en diferentes foros técnicos, comenzaron a cuestionar las metodologías formales, que representadas por CMM e ISO 15504, y respaldadas por la autoridad y los medios de sus respectivas organizaciones, estaban configurando una ingeniería del software basada en los procesos.

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 cliente, 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.

Manifiesto Ágil

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.

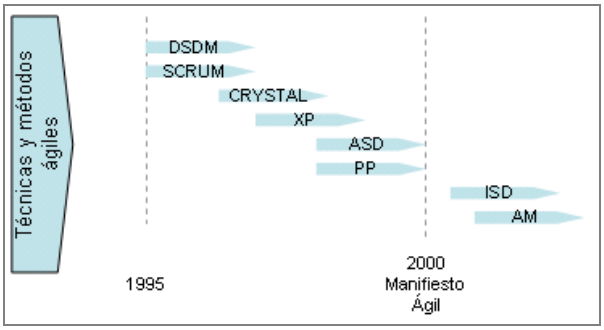

Fig. 7

El manifiesto ágil surgió con espíritu de respuesta desafiante y beligerante contra los métodos basados en procesos.

Los propios integrantes del manifiesto se auto-califican como "anarquistas organizacionales", y en estos años, desde uno y otro lado se han lanzado 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 auto-denominadas 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 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

No es propiamente un método o metodología de desarrollo, e implantarlo como tal resulta insuficiente.
Scrum define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP que, centrados en prácticas de tipo técnico, carecen de ellas.

Los principios de Scrum son:

- Equipos autogestionados.
- Una vez dimensionadas las tareas no es posible agregarles trabajo extra.
- Reuniones diarias en las que los miembros del equipo se plantean 3 cuestiones:
  - ¿Qué has hecho desde la última revisión?
  - ¿Qué obstáculos te impiden cumplir la meta?
  - ¿Qué vas a hacer antes de la próxima reunión?
- Iteraciones de desarrollo de frecuencia inferior a un mes, al final de las cuales se presenta el resultado a los externos del equipo de desarrollo, y se realiza una planificación de la siguiente iteración, guiada por cliente.

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 del entorno de desarrollo:

- 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 la cultura de sus ***Principios Fundamentales***, 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 (Rational Unified Process)

Es un proceso de Ingeniería del Software que proporciona una visión disciplinada para la asignación de tareas y responsabilidades en las organizaciones de desarrollo de software.
RUP es un "modelo-producto" desarrollado y mantenido por 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".

Métodos ortodoxos y métodos ágiles

Se dice que hay dos formas de abordar los proyectos:
1.- 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.
2.- Invertir mucho 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 sin duda esta es una visión simplificada, refleja las diferencias entre los modelos ortodoxos, o basados en procesos, y las metodologías ágiles; así como las razones de su desencuentro.

Los métodos formales (CMMI, ISO15504) abogan por el segundo modelo y 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 a construirlo, y en las prácticas formales de la gestión de proyectos, para trabajar sobre una planificación previa, y controlar su seguimiento.
Para estos modelos, se alcanza mayor nivel de madurez, cuantas más actividades del ciclo de vida del sistema de software se realicen según los procesos previamente diseñados e implantados en la organización.

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 sea la conformidad o no de lo producido, y las modificaciones propuestas por los usuarios.
Aparte de esta base común, también comparten 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, y con mayor o menor proximidad del extremo.

Los métodos formales llevados hasta su máximo nivel de madurez se situarían en el extremo de la ortodoxia, (lado izquierdo de la fig. 8) si bien es cierto que las representaciones continuas de estos modelos permiten adoptar sólo las prácticas para determinadas áreas de procesos, y llevarlos hasta el nivel de capacidad que se estime oportuno, por lo que un conocedor del modelo puede emplearlo como guía para realizar una implantación relativamente ágil, más o menos adecuada a las características de la organización.

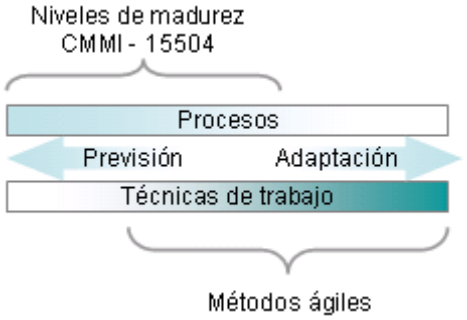

Fig. 8

Los métodos ágiles se sitúan más o menos cerca del extremo opuesto. Los más extremistas (quizá PP o XP) por la no utilización o incluso aversión a los procesos, se encuentran más en la categoría de prácticas de programación, o técnicas de trabajo (cómo hacer las cosas) que en la de modelos de procesos (qué cosas hay que hacer)

La circunstancia más o menos marcada en cada caso de "oposición" entre métodos ágiles y métodos formales, puede inducir al error de considerar a todos ágiles como similares o equivalentes.

En el lado opuesto sí que ocurre, y las similitudes entre CMMI e ISO 15504 son notables, pero con los ágiles ocurre sin embargo que los niveles de reconocimiento y rigor alcanzan diferencias notables entre ellos, así como la cobertura que llegan a alcanzar sobre determinadas áreas de procesos.

El oficio del gestor

Con lo visto hasta ahora tenemos ya el mapa de situación.
No es que sea importante conocerlo. Es necesario para no andar desorientados entre los árboles, y tener la visión del bosque en su conjunto.

Ya solo queda trazar la ruta y guiar el camino. ¡Casi nada!. Pero nadie ha dicho que la gestión de entornos

de desarrollo de software sea fácil. Basta con echar un vistazo a nuestro alrededor.

El mayor o menor grado de éxito en la calidad y eficiencia en el desarrollo de software depende del equilibrio que se consiga al conjugar los siguientes factores:

- Características de los proyectos de software gestionados.
- Visión de la organización.
- Cultura de la organización.
- Diseño y gestión del equilibrio productivo personas-procesos-tecnología.

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 desarrollado en un proyecto Open Source por iniciativa del propio grupo de desarrolladores; o el diseñado para asistir el pilotaje de un avión comercial.

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 varias decenas de personas durante un par de 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 entre el desarrollo de un sistema open-source sin ánimo de lucro, o de un sistema de gestión para una empresa que depende de él para abrir su negocio, es muy diferente.

El nivel de integridad del sistema, o daño que pueden producir los fallos del sistema, también 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.

Las 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 que debe dar beneficios?
¿Es el departamento de sistemas de una entidad bancaria que centra su misión en la seguridad?
¿Es una asociación de ingenieros para el desarrollo de un proyecto Open-Source?

Para alcanzar su misión, ¿Qué papel juega la innovación y vanguardia tecnológica?.
¿Se dedican a mantener operativos sistemas ya desarrollados?.
¿Programan 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 garantizan 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…?

El éxito: ¿Gestión o azar?

"Para todo problema complejo hay siempre una solución simple, que es errónea"

George Bernard Shaw

Cuando se desconocen las variables que actúan sobre un sistema, y su influencia, se habla de que presenta un comportamiento aleatorio ("caótico").
Por definición, no es posible predecir cuáles serán los resultados al actuar sobre un entorno caótico.

La acción externa modificará algunas variables, pero al no saber cuántas más están actuando, y cuál es su repercusión, no se pueden predecir los resultados.
Cuantas más variables se conocen, éste se va tornando más predecible y menos caótico, como ha ido ocurriendo, por ejemplo, en las últimas décadas con la predicción meteorológica.
La diferencia entre un sistema caótico y un sistema complejo es que en el primero son tantas las fuerzas que actúan que resulta materialmente imposible predecir su resultado (Ej: comportamiento de las bolas en un bombo).

En sistemas complejos, aunque son varias las fuerzas, sí que resulta posible su conocimiento, o el conocimiento de las más relevantes, y actuar con probabilidades de certidumbre muy altas sobre las consecuencias.
Desde este punto de vista, un entorno de desarrollo de software es un sistema complejo en el que se pueden combinar y poner en interacción múltiples elementos, y con diferentes pesos y formas de gestión (Véanse en el artículo los modelos y técnicas que pueden conjugarse, y los factores que apuntan las preguntas del apartado anterior).

Aunque sobre los sistemas complejos, a diferencia de los aleatorios, sí es posible actuar con factores de predictibilidad elevados, si desconocemos su ecuación de fuerzas, se presentan de comportamiento tan aleatorio como los sistemas caóticos.
Para ser más exactos en esta apreciación, lo cierto es que un sistema complejo parece caótico, a quien desconoce cuáles son sus variables y su relación, pero esto no es muy frecuente que ocurra, ya que si alguien sabe que hay variables aprehensibles y que conociéndolas podrá actuar con mayor seguridad, no es normal que continúe jugando al azar.
El hombre, por su naturaleza necesita seguridad, y si sabe que las reacciones de un sistema responden a un mecanismo más o menos complejo, no es normal que prefiera ignorarlo.

Por eso el error más habitual consiste en actuar bajo el fenómeno de la "superstición", tomando como referencia variables falsas, o creyendo que son tan sólo una o dos las fuerzas que deciden el comportamiento del sistema.
De esta forma, al igual que el jugador de ruleta supersticioso se aferra a su camisa de la suerte, por la razón de que le dio un par de noches afortunadas, un gestor se puede aferrar a determinadas prácticas de gestión, porque una vez funcionaron con éxito, o porque el último libro de *management* las considera "mano de santo".

Por este camino también puede acabarse maldiciendo la ruleta, perdón, la gestión; cayendo en el escepticismo sobre los consultores, las teorías de *management*, de comportamiento organizacional, los modelos de calidad, etc.
Porque claro, uno siempre hace las cosas bien. Por eso si los resultados salen mal el problema está fuera de él.

Conclusiones

Para diseñar y gestionar entornos de producción eficientes se debe trabajar con pensamiento sistémico, comprendiendo que todos los factores implicados en la producción de software componen un sistema interrelacionado, imbricado y alineado con la misión de la organización.

Lo contrario, y probablemente más habitual consiste en actuar desde una visión fáctica e inmediata, buscando 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.

La multiplicidad de técnicas, modelos, herramientas, modos de gestión, es consecuencia de la diversidad de organizaciones y tipos de proyectos posibles.

Evitar la gestión lineal

El error habitual es trabajar con gestión lineal y a-sistémica, administrando soluciones sintomáticas para cada situación.

La gestión lineal, carece de la perspectiva necesaria, y responde de forma azuzada a la presión del día a día.
Sin esa perspectiva no se ve el sistema, y el marco de trabajo 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 (allá cada cual con el estilo de gestión que aplique) 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… (también aquí cada cual sintonizará mejor con unas u otras decisiones.).

Etcétera.

Para cada necesidad se abre el botiquín, y se busca 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 muy tentadores que se deben evitar:

- Aplicar soluciones enfocadas en resultados en el corto plazo.
- Obtener mediciones y datos que demuestren 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 apariencia 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, absolutos con resultados inmediatos.

Un modelo de procesos CMMI con nivel 5 de madurez puede garantizar resultados en línea con la misión de una gran 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 guardar coherencia, de forma que se potencien, y no se contrarresten.

Un "*know-how*" adecuado, basado en el conocimiento tácito de las personas, compaginado con equipos desmotivados no es una buena combinación.

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 atino 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 del producto del negocio: 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 híbridos 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.

Modelos y prácticas de producción o de calidad hay muchos: EFQM, Six Sigma, CMMI, ISO 15504, ISO 9000, DSDM, MSF, XP, PP, etc.

Teorías y técnicas para la gestión de las personas hay muchas: evaluaciones de desempeño, gestión por competencias, coaching, Maslow, Taylorismo, Herzberg, Mc Gregor, Vroom, Shein, etc.

La tecnología ofrece múltiples herramientas CAD de ayuda en la ingeniería del software (Requisite Pro, Visio, Subversión, CVS, MSTS, Racional Rose, etc) así como plataformas de desarrollo, operativas, lenguajes, compiladores....

Aquí hay de todo, como en botica; pero con el mismo criterio que las medicinas, hay que considerar que aunque todos los remedios son buenos, ni todos sirven para todos, ni deben combinarse al azar.

El conocimiento general de la naturaleza del software y del marco de nuestra industria son ingredientes necesarios tanto para abordar el diseño de procesos con recursos propios, como para evaluar con acierto a una asesoría externa que pueda ofrecer una orientación objetiva; porque a la consultoría le ocurre lo que a nuestra profesión, que no está normalizada, y en el peor de los casos uno puede terminar con un curandero o un traumatólogo, cuando lo que necesita, a lo mejor es un alergólogo.

Bibliografía

JENNIFER STAPLETON, DSDM Business Focused Development, DSDM Consortium, 2003

KEN SCHWABER 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

Derechos