Programación Extrema y Calidad.
Estudio de Compatibilidad XP – CMM

Mg. Rodolfo Bertone[1], *A.C. Ariel Pasini*[2], *Mg. Hugo Ramón*[3]

III-LIDI[4] - Facultad de Informática - UNLP

Resumen

La Programación Extrema (XP) es una práctica de Ingeniería de Software que permite el desarrollo de aplicaciones desde una perspectiva diferente a lo que plantean la mayoría de las metodologías clásicas respecto de la construcción del software.

Por su concepción XP puede ser visto como una forma un tanto caótica de trabajo donde se pone énfasis en resolver el problema del presente sin planificar hacia el futuro mediato. Desde esta perspectiva un enfoque de calidad podría verse como una expectativa ambiciosa que no cae dentro de los rasgos de la metodología/práctica.

En este trabajo se presenta a XP y se lo compara contra un estándar de calidad como el plantado por le SEI a través de CMM. Además, se concluye bajo que contextos XP es compatible con CMM permitiendo alcanzar niveles aceptables de calidad en la construcción del software.

Palabras Clave

Ingeniería de Software. Programación Extrema. Calidad. SQA.

---

[1] Profesor Adjunto Dedicación Exclusiva – Facultad de Informática UNLP
mail: pbertone@lidi.info.unlp.edu.ar
[2] Jefe de Trabajos Prácticos Dedicación Exclusiva - Facultad de Informática UNLP
mail: apasini@lidi.info.unlp.edu.ar
[3] Profesor Adjunto Dedicación Exclusiva – Facultad de Informática UNLP
mail: hramón@lidi.info.unlp.edu.ar
[4] Instituto de Investigación en Informática LIDI. Facultad de Informática. UNLP. Calle 50 y 115. La Plata. Buenos Aires. Argentina. TE +54 221 4227707

Introducción

Programación Extrema

Una definición inicial de Programación Extrema plantea a XP como una forma ligera, eficiente, de bajo riesgo, predecible, científica y "amena" de desarrollar software [1]. XP se diferencia de otros métodos para la construcción de software en conceptos como [16]:

- ✓ Realimentación por parte del usuario en ciclos cortos y continuos.
- ✓ Enfoque de planificación incremental, la cual genera un plan global de desarrollo, el cual se espera que evolucione a lo largo de la vida del proyecto.
- ✓ Capacidad para programar en forma flexible la implementación de funcionalidades, respondiendo a las necesidades cambiantes de negocios.
- ✓ Confianza en pruebas de software automatizadas, definidas y escritas por programadores y clientes para controlar el proceso de desarrollo, permitiendo la evolución del sistema y captando los defectos lo antes posible.
- ✓ Confianza en la comunicación oral, las pruebas y el código fuente para obtener la estructura e intención del sistema.
- ✓ Confianza en el proceso de diseño evolutivo que perdura mientras lo haga el sistema.
- ✓ Confianza en la colaboración muy correlacionada entre todos los programadores involucrados.

Las cualidades anteriores presentan a XP como una disciplina de desarrollo de software, debido a que hay determinadas reglas, normas o preceptos que se deben seguir para su aplicación. Por ejemplo, no es posible decidir si las pruebas se llevan a cabo o no, simplemente deben hacerse porque son parte vital del método.

XP está diseñado para trabajar con proyectos que puedan llevarse adelante con un número limitado de programadores, que no estén fuertemente condicionados por el entorno de cómputo existente y en donde un trabajo razonable de ejecución de pruebas puede hacerse paulatinamente y tiempos acotados (horas).

Una característica de XP consiste en que el trabajo se realiza de a pares. Esto es, cada grupo de trabajo está compuesto por series de dos programadores los cuales se complementan y ayudan para generar la solución a cada problema. De esta forma, mientras uno programa, el otro realiza o genera casos de prueba que se utilizarán para evaluar los resultados de cada iteración [6]. En general, en XP:

- ✓ Las parejas de programadores trabajan juntos.
- ✓ El desarrollo es guiado por pruebas. Primero se generan los casos de prueba y luego se procede con la codificación. El desarrollo de cada iteración finaliza cuando todas las pruebas funcionan y no se puede generar otro caso particular [7].
- ✓ Las parejas no solo ejecutan los casos de pruebas, sino también se encargan de la evolución del diseño del sistema. Los cambios no se restringen a un área particular. Las parejas añaden valor al análisis, diseño, implementación y prueba del sistema [8] [9].
- ✓ La integración sigue inmediatamente al desarrollo, incluyendo las pruebas de integración.

Los siguientes principios, pueden establecerse como lo pilares fundamentales sobre los que la metodología XP se encuentra basado [18]:

- ✓ Realimentación rápida: el tiempo entre una acción y su realimentación resulta crítico. Los negocios aprenden la forma en que el sistema puede contribuir a su mejora y realimentan al mismo en tiempos muy reducidos (semanas en lugar de meses o años). Los programadores aprenden como mejorar el diseño, implementación y prueba del sistema, y realimentan lo aprendido en término de "horas", en lugar de días o semanas.
- ✓ Asumir "lo sencillo como base de resolución de un problema. Aquí, XP entra en contradicción con las técnicas clásicas para el desarrollo de software donde el reuso es un concepto afianzado [4]. Sin entrar en discusiones detalladas, la tendencia actual es desarrollar para el futuro, pensando en la reutilización de componentes. Si todo módulo nuevo se piensa como un elemento reutilizable, se puede ir construyendo una biblioteca de clases o funciones que permitirá generar software nuevo ensamblando componentes. XP propone hacer un trabajo para el día a día, sin alterar por ello la calidad, manteniendo la confianza en que las habilidades adquiridas por el equipo de trabajo, lo cual permitirá en el futuro añadir más complejidad al problema, si fuera necesario.
- ✓ Cambio incremental: el desarrollo incremental de software no es una idea innovadora, los grandes cambios hechos de una vez raramente tienen a generar el resultado esperado. La diferencia se establece cuando cualquier problema se resuelve a partir de una serie de pequeños cambios. Estos cambios propuestos por XP están relacionado con el diseño, los planes y el equipo de trabajo siempre realizado de manera paulatina y gradual.
- ✓ Aceptar el cambio: la mejor estrategia es aquella que conserva la mayoría de opciones mientras se resuelve el problema más crítico en cada momento del desarrollo.
- ✓ Trabajar con calidad para desarrollar con calidad. Este punto, el centro del trabajo, será discute en el resto de la presentación.

La siguiente *tabla 1* plantea algunos elementos básicos de la metodología XP y como lograrlos en forma teórica y práctica.

Elementos	**En teoría**	En la práctica
Revisión de código	**Revisar el código en todo momento**	Programación de a pares
Testeo	**Testing permanente, involucrando al cliente**	Pruebas de unidad y test funcional
Diseño	**Generar diseño de a parte, para cada regla de negocio**	Refactoring
Simplicidad	**Trabajar siempre con el diseño más simple que soporte la funcionalidad del sistema**	La cosa más simple que pueda funcionar
Arquitectura	**Todos los actores involucrados trabajan para refinar la arquitectura**	Metáforas
Test integración	**Integrar y testear varias veces al día**	Integración continua
Iteraciones cortas	Hacer iteraciones cortas en términos de tiempo (minutos u horas) en lugar de (semanas o meses)	Juego de la planeación

Tabla 1

Capability Maturity Model

El Software Engineering Institute (SEI) ha desarrollado CMM como un modelo que permite definir y construir capacidad organizacional, hasta el punto de obtener un estándar fuertemente aceptado en la comunidad del software. [10]

Los requerimientos de CMM se pueden resumir en una organización de 5 niveles que establecen 52 sentencias que son objetivos del modelo, desarrollado 18 áreas clave de proceso (KPA). Estas prácticas, subprácticas y ejemplos guían al profesional de software para tomar decisiones razonables sobre el proceso que se encuentra implementando.[13]

La *tabla 2* presenta una revisión somera de CMM, sus cinco capas, el enfoque de cada una y las 18 KPA´s [14].

Nivel		KPA
1: inicial	**Gente competente**	
2: repetible	**Proceso de administración de proyectos**	Administración de requerimientos Planeamiento de proyecto de soft Seguimiento de proyectos de soft Administración de subcontratos Planes de SQA Administración de configuración
3: definido	**Proceso de ingeniería y soporte organizacional**	Foco en el proceso de la organización Definición del proceso de la organización Programa de entrenamiento Administración integrada de soft Ingeniería de producto de software Coordinación entre grupos Revisión por pares
4:administrado	**Calidad de producto y proceso**	Administración cuantitativa de proceso Administración de calidad de soft
5:optimizado	Mejora continua de proceso	Prevención de defectos Administración de cambio de tecnología Administración de cambio de proceso

Tabla 2

Las KPA de CMM pueden definirse con los siguientes propósitos [15]:

- ✓ Nivel 2: Repetible
  - o Administración de requerimientos: Establecer un entendimiento común entre el cliente y el equipo a cargo proyecto de software sobre los requerimientos de usuario
  - o Planeamiento de proyecto de software: establecer planes razonables para IS y administración general del proyecto.
  - o Seguimiento de proyectos de software: proveer una forma de monitoreo del estado de avance del proyecto, de manera que la administración pueda actuar efectivamente cuando la performance se desvía de manera significativa del plan establecido.
  - o Administración de subcontratos: elegir contratistas de software calificados y administrarlos efectivamente.
  - o Planes de SQA: proveer un manejo con visibilidad adecuada dentro del producto y proceso de software.

- Administración de configuración: establecer y mantener la integridad del proceso de software a través del ciclo de vida del proyecto.

✓ Nivel 3: Definido
- Foco en el proceso de la organización: establecer responsabilidad organizacional para las actividades del proceso de software que mejoren la capacidad organización general del mismo.
- Definición del proceso de la organización: desarrollar y mantener un conjunto de elementos para el proceso de software que mejoren la performance del mismo a lo largo del desarrollo y provean las bases para un beneficio organizacional.
- Programa de entrenamiento: desarrollar habilidades y conocimiento en los individuos que permitan llevar adelante sus roles de manera eficiente y efectiva.
- Administración integrada de software: integrar las actividades de ingeniería y administración de software dentro de una disciplina coherente y definida.
- Ingeniería de producto de software: uso consistente de proceso de ingeniería bien definido que integre todas las actividades de la IS para producir productos o sistemas correctos y consistentes de una forma efectiva y eficiente.
- Coordinación entre grupos: establecer un camino para el grupo de IS que le permita participar activamente con otro grupos de ingeniería a fin de lograr un producto de software final que se adapte a las necesidades del cliente / usuario.
- Revisión por pares: remover defectos del producto lo antes posible y de la forma más eficiente.

✓ Nivel 4: Administrado
- Administración cuantitativa de proceso: control cualitativo de la performance del proceso de desarrollo de proyectos.
- Administración de calidad de software: cuantificar la calidad de los productos de software y alcanzar metas previamente especificadas.

✓ Nivel 5: Optimizado
- Prevención de defectos: identificar las causad de defectos y prevenir su ocurrencia.
- Administración de cambio de tecnología: identificar nuevas tecnologías (herramientas, métodos y procesos) e introducirlas dentro de la organización de manera ordenada.
- Administración de cambio de proceso: mejora continua del proceso que la organización tiene para desarrollo de sistemas, buscando mejorar calidad del producto, aumentar productividad, disminuyendo el tiempo de desarrollo.

Calidad en la Construcción de Software.

El proceso de desarrollo del software involucra cuatro variables: costo, calidad, tiempo y ámbito. Generalmente, la forma de trabajar con estos factores consiste en dejar tres de ellas para ser definidas por el usuario o cliente y la restante por el grupo de trabajo [1].

La interacción de las variables no es sencilla. Más dinero mejora la relación de trabajo hasta cierto punto. Disponer de un tiempo escaso influirá negativamente en

calidad en primera instancia, luego en el ámbito y posteriormente en el costo. La calidad resulta en la variable más difícil de administrar, usualmente se la sacrifica obteniendo mejoras a corto plazo, pero el costo se transforma en exponencial cuando el plazo se extiende. Por último, el ámbito permite entregar resultados de mejor calidad (mientras el problema no esté resuelto), también permite entregar software más rápido y barato. Resumiendo: no es posible obtener software más rápido aumentando los gastos económicos [12].

Otro factor a tener en cuenta es el costo ligado al cambio. Históricamente el costo de un cambio está ligado al momento del mismo. De esta forma un cambio en etapa de Ingeniería de requerimientos tiene costo nulo, un impacto más alto en análisis, diseño, implementación y prueba; siguiendo un crecimiento cercano al exponencial, el cual se ve más reflejado en con el sistema en producción. [5]

Se ha estudiado en detalle la idea de minimizar el impacto del cambio, en particular en el costo final del proyecto (logrando mejores lenguajes, mejor tecnología de BD, mejores práctica en la codificación, mejores entornos y herramientas, nuevas metodologías de trabajo, mejoras en la elicitación de requerimientos, etc. [3]) . El objetivo buscado es revertir la curva de costo del cambio, desde una exponencial hacia que creciera más lentamente y pudiera tener un comportamiento asintótico a largo plazo.

La metodología XP tiene como premisa este aspecto. Si el costo del cambio crece lentamente con el tiempo, se debería actuar de una forma diferente a lo obrado cuando el crecimiento es exponencial. La toma de grandes decisiones debe postergarse lo más posible, para aplazar el costo asociado a la misma y tener más tiempo para analizar posibles acciones. Solamente se debería implementar lo que se debe hacer (o lo que es necesario), con la esperanza que las necesidades que se anticipan para un futuro no se presenten. Se deberían introducir en el diseño aquellos elementos que obtengan una simplificación del código actual o que llevarse a una escritura más sencilla del mismo.

XP como metodología ayuda a mantener estable y acotada la curva del costo ante cambios de requerimientos. Esto es posible solamente en aquellos contextos donde los cambios de los requerimientos son graduales, consensuados con el usuario, con problemas acotados a contextos específicos del software a modificar. [12]

El problema común en el desarrollo del software es el riesgo. Algunos ejemplos de riesgos comunes existentes la mayoría de los proyectos de software son [2]: retrasos en la planificación, proyectos cancelados, deterioro del sistema, tasa de defectos alta, requisitos mal evaluados, cambios en el negocio, falsas expectativas por parte del usuario, entre otros. ¿Puede XP minimizar los casos de riesgo anteriormente analizados?.

La experiencia obtenida a partir de la observación de desarrollos de software indica que el riesgo que más se presenta es aquel que está relacionado con la planificación temporal y el esquema de tiempo resultante. Generalmente estos esquemas sufren desviaciones en su cumplimiento. XP propone un ciclo de versión corto, pocos meses como máximo, de tal manera que el alcance de cualquier desviación sea limitado, minimizando el efecto sobre la planificación temporal.

Por otro lado, cuando se escucha por primera vez la idea de XP, la mayoría tiende a ver esta metodología como un retroceso. Se tiende a pensar que es volver al pasado, reaccionar ante un pedido rápidamente y usualmente con una solución equivocada e incorrectamente documentada y analizada. Se puede decir que en parte se tiene razón, pero la defensa fuerte que tiene la metodología es que el hecho de reacción

rápida no conlleva una solución equivocada dado que el usuario siempre se mantendrá cercano al proyecto permitiendo corregir los defasajes que pudieran surgir.

Como queda, entonces, el tema de calidad? Como se discutió hasta aquí, XP presenta una solución diferente, con una concepción diferente. Sin embargo, determinados principios, parámetros y características hacen de esta práctica una alternativa donde la calidad puede estar presente. En el apartado siguiente se discute y compara contra el modelo de capacidad del software.

Calidad y XP.

CMM pone énfasis en dos puntos: implementar una metodología de desarrollo efectiva y eficiente, y un proceso de mejora continua. XP por otro lado, especifica un conjunto de prácticas que son efectivas en el contexto de equipos y desarrollo pequeños que se adapten rápidamente a los cambios de los requerimientos. Tomados en conjunto los dos métodos pueden analizar puntos en común, particularmente en conjunción con otras buenas prácticas de IS [19].

La *tabla 3* presenta las 18 KPA´s presentadas anteriormente analizando el grado de satisfacción de la misma por parte de XP.

Nivel	KPA	Satisfacción XP
2	**Administración de requerimientos**	Alcanzado
2	**Planeación de proyectos de software**	Alcanzado
2	**Seguimiento de proyectos de software**	Alcanzado
2	**Administración de subcontratos**	No cubre
2	**Planes de SQA**	Cubre
2	**Administración de configuración**	Cubre
3	**Foco en el proceso de la organización**	Cubre
3	**Definición del proceso de la organización**	Cubre
3	**Programa de entrenamiento**	No cubre
3	**Administración integrada del software**	No cubre
3	**Coordinación entre grupos**	Alcanzado
3	**Ingeniería de producto de software**	Alcanzado
3	**Revisión de a pares**	Alcanzado
4	**Administración cuantitativa de proceso**	No cubre
4	**Administración de calidad de software**	Satisface menos
5	**Prevención de defectos**	Cubre
5	**Administración de cambio de tecnología**	No cubre
5	Administración de cambio de proceso	No cubre

Tabla 3

XP y prácticas CMM de Nivel 2

El KPA de administración de requerimientos se alcanza ampliamente a partir del uso de historias, tener incorporado al cliente en el equipo y mediante integración continua. A pesar que los requerimientos del sistema pueden evolucionar rápida y dramáticamente en el ciclo de vida de un proyecto, XP integra una realimentación continua con el cliente de manera de poder captar lo antes posible sus nuevas necesidades y expectativas. Eso se logra implantando siempre como parte del diseño el desarrollo en ciclos pequeños y simples involucrando a todos los actores (entre ellos el cliente). El entendimiento común se establece y mantiene involucrando continuamente al cliente en la construcción de historias y, posteriormente, seleccionando las mismas para nuevos desarrollo.

El planeamiento de proyectos de software se logra desde XP con el juego de la planeación y con la evolución en ciclos cortos y sencillos. La primera actividad tiene que ver con que el equipo de desarrollo de software realice planeamiento temprano. XP integra el equipo dentro de procesos de compromiso que deben estimar el esfuerzo involucrado para implementar las historias del cliente. Este mantiene el control de las prioridades del negocio seleccionando las historias que se deberán implementar en la siguiente iteración. Por definición, XP es un método incremental y evolutivo. El plan de proyecto no se establece para todo el ciclo de vida del problema, las metáforas que se van definiendo establecen una visión para la dirección del proyecto. Como resultado, los desarrolladores pueden identificar y manejar de manera eficiente los riesgos que puedan aparecer [17].

La siguiente KPA tiene que ver con el seguimiento de proyectos de software. En este caso XP presenta características como la velocidad de desarrollo y la forma de plantar las historias como aproximaciones para su adaptación hacia la calidad. Los procesos de compromiso de XP establecen claramente las expectativas de tanto el cliente como los desarrolladores. El hecho de trabajar 40 horas semanales, si bien no es algo explícitamente solicitado por CMM, establece un marco de trabajo coherente que es considerado como una buena práctica.

Respecto al área clave de proceso que tiene que ver con subcontrato, en la tabla 3 fue indicado como No Cubre. Esto quiere significar que XP no presenta ninguna característica que lo acerque a tal fin de CMM.

La KPA que tiene que ver con SQA es lo que se está discutiendo en este trabajo. En la sección de conclusiones se presentan más argumentos, pero se puede asumir que el tema de calidad, en general, es tenido en cuenta por XP. Cada una de las características que maneja XP se acercan hacia el concepto de calidad. No se debe perder de vista hacia que tipo de sistemas se orienta XP: chicos y medianos, en este entorno se puede considerar que los aspectos mínimos de calidad son alcanzados con políticas de trabajo de a pares, iteraciones cortas, diseños sencillo y por sobre todo pruebas intensivas que acercan al cliente.

Un área clave para calidad tiene que ver con la administración de configuración del software que se realiza. XP logra alcanzar este punto, nuevamente a través de sus características básicas: trabajo de a pares, iteraciones pequeñas y por sobre todo integración continua. De esta forma es posible mantener una configuración de software adecuada.

XP y prácticas de Nivel 3

De las siete áreas del nivel definido, cinco son abarcadas con diferente éxito por XP. Solamente quedan fuera el programa de entrenamiento y la administración integrada del software.

XP cubre, al menos parcialmente, el primer ítem propuesto para el nivel 3: foco en el proceso de la organización. XP centraliza el trabajo en el equipo más que en la organización como lo plantea CMM.

XP no presenta ninguna característica respecto de programas de entrenamiento o administración integrada del software.

Por último, los KPA de Coordinación entre grupos, ingeniería de producto de software y revisión de a pares son tres aspectos que XP cubre ampliamente. La comunicación es la base de XP lo que sirve para maximizar la coordinación entre

grupos. Las metáforas, el diseño simple, el refactoring, codificación estándar, pruebas de unidad y funcional sirven para administrar la ingeniería de producto de software.

Por último, se ha mencionado varias veces en esta presentación los aspectos relacionados con la programación de a pares y, ligado a esto, la revisión de a pares. Con lo cual esta KPA es totalmente cubierta.

XP y prácticas superiores

A partir de los niveles administrado y optimizado XP empieza a alejarse de las prácticas de de CMM. Entre las 5 áreas clave solamente la prevención (y administración) de defectos del nivel 4 está potencialmente cubierta con los mecanismos de prueba de XP. Igualmente las prácticas de mayor nivel de CMM están pensadas para organizaciones donde, por su concepción, no abarcan proyectos de pequeña o mediana envergadura.

Conclusiones – Trabajos Futuros

En general, se puede plantear que XP se enfoca sobre trabajo técnico, mientras que CMM apunta a rasgos de administración. El punto que XP no presenta, y que resulta de mucha importancia para CMM, es el concepto de institucionalización. Esto puede definirse como establecer una cultura de trabajo que sea utilizada en toda la organización.

XP ignora la infraestructura que CMM identifica como clave para lograr una buena práctica en ingeniería y administración, a pesar que implícitamente lo tiene en cuenta. Si se analizan los rasgos comunes contemplados en cada KPA de CMM desde una perspectiva de la práctica que lleva adelante, se puede analizar el nivel de aceptación o cumplimiento por parte de XP hacia ese aspecto. Entonces, se puede asumir que respecto a:

- ✓ Compromiso a ejecutar: XP no presenta soluciones en cuanto a prácticas políticas o de liderazgo.
- ✓ Habilidad a ejecutar: XP presenta estructura de organización, manejo ordenado de recursos y fondos y capacidad de entrenamiento.
- ✓ Medidas y análisis: están presente en la metodología XP.
- ✓ Verificación de implementación: se realiza una administración de negligencias o fallos en la elaboración del proyecto de una forma sistemática y cuidadosa (el mecanismo de prueba XP está orientado a este fin). El aseguramiento de calidad, otra práctica en este rasgo de KPA, se cumple (por todo lo argumentado hasta el momento. La última práctica que se agrega en este contexto, tiene que ver con un seguimiento detallado de problemas, fallos o errores, XP no contempla esta opción, solo se analiza el "presente" del inconveniente y se lo soluciona, pero sin hacer un seguimiento hacia atrás o delante.

Mucho del formalismo que caracteriza el proceso de CMM está orientado hacia grandes proyectos, con múltiples requerimientos generalmente complejos. El conjunto de prácticas XP se torna cada vez más difícil de implementar a medida que los problemas crecen. XP trata con grupos pequeños trabajando con proyectos de pequeña o mediana envergadura. [12]

Características como diseño basado en la arquitectura, diseño generado para el cambio, refactoring, diseño sencillo y pequeño tiene su punto de convergencia hacia la administración sistemática del cambio. El diseño arquitectónico que pone mayor

énfasis en la flexibilidad es la meta para cualquier metodología OO, de esta forma XP puede condecirse dado que tiene el mismo objetivo. Sin embargo, los proyecto tienden a convertirse en ser multidisciplinario, lo que hace de XP una metodología difícil de implantar en estos casos.

Los principios de comunicación y simplicidad impuestos por XP son fundamentales en organizaciones que tengan a CMM como base para calidad. Cuando se define el proceso, las organizaciones deberían capturar el mínima información básica necesaria, estructurar definiciones utilizando buenos principios de diseño de software (abstracción, herencia, por ejemplo) y enfatizar en ideas como reusabilidad. [11]

La principal objeción para utilizar XP para la mejora de procesos es que no se acerca demasiado a técnicas de administración y organización, que son muy enfatizadas por CMM. Implementar la clase de ambiente colaborativo que XP asume requiere gran infraestructura organizacional y de administración.

Existe una idea estructural donde CMM difiere de XP dado que es un proceso riguroso y muy estable no resulta muy adecuada como base de comparación. XP es un proceso que necesita mucha disciplina, está muy bien definido y resulta muy claro en dicha definición. Por esto motivos se puede asumir que CMM y XP son complementarios.

Como conclusiones finales, los autores de este informe consideraban inicialmente a XP como una metodología que contradecía todos los principios establecidos para el desarrollo del software. La idea de trabajar "bajo el caos" no estaba disponible dentro de los posibles entornos de desarrollo practicables. En este contexto resultaba una vuelta al pasado: responder a estímulos puntuales, dejar de lado el todo y pasar a ver las partes, resolver el problema de hoy y no planificar hacia futuro, etc.. La mayoría de los aspectos propuestos por XP entraba en conflicto la lógica del trabajo realizado hasta el momento, la lógica de las metodologías estudiadas y, básicamente, la lógica del conocimiento recibido.

Sin embargo, analizado XP y observado que se trabaja bajo ciertos principios claramente establecidos y conocidos por el entorno de desarrollo; y que los mismos son claros y estrictos, basados en la prueba como elemento de validación y evolución, entonces es posible considerar a XP como otra metodología más, que implanta una forma de solución diferente.

¿Como se evalúa un enfoque de calidad, como el planteado por CMM en el contexto de trabajo de XP? La respuesta a nuestro juicio resulta clara: los niveles 2 y 3 con alcanzables. Como se expuso anteriormente, si hay orden en el trabajo entonces hay posibilidades de llegar a nivel Repetible. Para el nivel Definido varios KPA resultan alcanzables desde XP: trabajo de pares, coordinación y organizaciones. Los niveles superiores, en tanto, no condicen con las metas de XP, el tipo de organización y, por consiguiente, el tipo de problemas que se enfrentan las empresas que garantizan nivel 4 y 5 de CMM no tendrán a XP como metodología de campo.

Referencias bibliográficas

[1] Beck Kent, *Una aplicación de la programación extrema*. *Aceptar el cambio*. Addision Wesley. 2000.

[2] Jones Caper. *Assessment and Control of Software Risk.* Caper Jones. Yourdon Press. 1994

[3] Loucopoulos, P; Karakosas, V. *Systems Requeriments Engineering*. McGraw Hill. Book Company. 1995

[4] Sommerville Ian. *Requeriments Engineering, A good practice guide*. John Wiley. *1997.*
[5] Pressman Roger. *Ingeniería de Software.* Mc Graw Hill. 1999.
[6]Willams, Laurie; Kessler Robert; Cunningham, Ward; Jeffries, Ron. *Strengthening the case for Pair-Programming*.
[7] Constantine, L. *Constantine on Peopleware*. Englewood Cliffs, NJ: Yourdon Press, 1995.
[8] Coplien, J. *A development process Generative Pattern Languaje*. In *Pattern Languages of Program Design*. Addison Wesley 1995. pag 183-237.
[9] Anderson A., Beattlie, Ralph, Beck, Kent et al. *"Chrysler goes to Extremes"* in Distributed Computing, vol. October 1998 pag. 24-28.
[10]M. C. Paulk et al., *The Capability Maturity Model. Guidelines for Improving the Software Process.* Addison Wesley. Reading. Mass., 1995.
[11] M.C. Paulk *Using the Software CMM with Good Judgment*. ASQ software quality professional, vol. 1, no. 3, June 1999, pag. 19-29.
[12] R. Bertone, A. Pasini *Programación extrema y calidad*. Informe Curso de Postgrado de Gestión de Calidad según Normas ISO. Aplicaciones a Software. Facultad de Informática. Junio 2005.
[13] www.sei.cmu.org/cmm
[14] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, *Capability Maturity Model*^{SM} *For Software, Version 1.1* Technical Report CMU/SEI-93-TR-024 ESC-TR-93-177 February 1993
[15] C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.
[16] A. Jackson, Shiu Lun Tsang, A. Gray, C. Driver, S. Clarke, *Behind the rules: XP experiences*. Agile Development Conference, Pag. 87-94. June 2004
[17] L. Williams, R. Upchurch, *Extreme programming for Software Engineering Education?* Frontiers in Education Conference, Vol 1 Pag 12-17. Oct 2001.
[18] J. Nawrocki, B. Walter, A. Woiciechowski, *Toward maturity model for Extreme Programmin.* Euromicro Conference. PRoceedings. 27Th. Pag 233-239. Sept. 2001
[19] D.J. Reifer, *XP and the CMM*, Software IEEE, Vol 20 Issue 3, Pag 14 – 15. May-Jun 2003