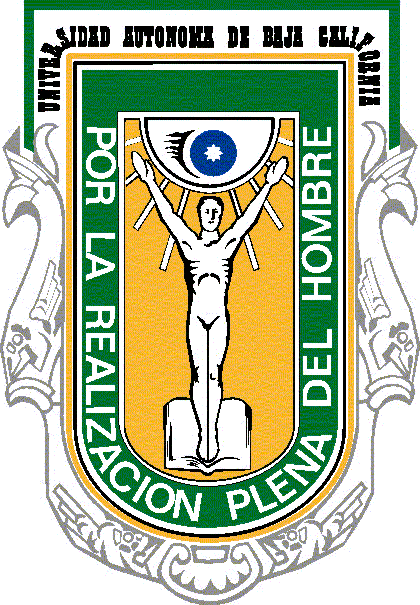
UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA



*“xtreme programmins (xp)”*

Ingeniería de Software

José Luis Alejandro Gomero Portilla

Alumnos: René Efrén Deveze Astorga

Erika V. Santoyo Soto

Grupo 361

**INTRODUCCION:**

XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico

**DEFINICION:**

Es una metodología de desarrollo de la [ingeniería de software](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) formulada por [Kent Beck](http://es.wikipedia.org/wiki/Kent_Beck), autor del primer libro sobre la materia, (*Extreme Programming Explained: Embrace Change* (1999)). La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

**CARACTERISTICAS:**

* **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.
* [**Pruebas unitarias**](http://es.wikipedia.org/wiki/Prueba_unitaria)**continuas**, frecuentemente repetidas y automatizadas, incluyendo [pruebas de regresión](http://es.wikipedia.org/wiki/Pruebas_de_regresi%C3%B3n). Se aconseja escribir el código de la prueba antes de la codificación.
* [**Programación en parejas**](http://es.wikipedia.org/wiki/Programaci%C3%B3n_en_pareja): se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. La mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
* Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
* **Corrección de todos los**[**errores**](http://es.wikipedia.org/wiki/Error_de_programaci%C3%B3n) antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
* [**Refactorización**](http://es.wikipedia.org/wiki/Refactorizaci%C3%B3n)**del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
* **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
* **Simplicidad en el código**: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

**CICLO DE VIDA:**

**Fase de la exploración:** En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.   
Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.  
**Fase del planeamiento:** se priorizan las historias de usuario y se acuerda el alcance del release. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma. La duración del cronograma del primer release no excede normalmente dos meses. La fase de planeamiento toma un par de días. Se deben incluir varias iteraciones para lograr un release. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución. La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema está listo para producción.  
**Fase de producción:** requiere prueba y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente. En esta fase, los nuevos cambios pueden todavía ser encontrados y debe tomarse la decisión de si se incluyen o no en el release actual. Durante esta fase, las iteraciones pueden ser aceleradas de una a tres semanas. Las ideas y las sugerencias pospuestas se documentan para una puesta en práctica posterior por ejemplo en la fase de mantenimiento. Después de que se realice el primer release productivo para uso del cliente, el proyecto de Xp debe mantener el funcionamiento del sistema mientras que realiza nuevas iteraciones.  
**Fase de mantenimiento:** requiere de un mayor esfuerzo para satisfacer también las tareas del cliente. Así, la velocidad del desarrollo puede desacelerar después de que el sistema esté en la producción. La fase de mantenimiento puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.  
**Fase de muerte:** Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la  documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

**ROLES:**

### Programador

Escribe las pruebas unitarias y produce el código del sistema.

### Cliente

Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar el mayor valor de negocio.

### Tester

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

### Tracker

Es el encargado de seguimiento. Proporciona realimentación al equipo. Debe verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

### Entrenador (coach)

Responsable del proceso global. Guía a los miembros del equipo para seguir el proceso correctamente.

### Consultor

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

### Gestor (Big boss)

Es el dueño del equipo y el vinculo entre clientes y programadores. Su labor esencial es la coordinación.

**Ventajas:**

Programación organizada.

Menor taza de errores.

Satisfacción del programador.

**Desventajas:**

Es recomendable emplearlo solo en proyectos a corto plazo.

Altas comisiones en caso de fallar.

**Bibiografia:**

<http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html>

<http://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema>

<http://extremeprogramming.host56.com/ARTICULO5.php>