# Extreme Programming

El Extreme Programming (abreviado XP) es una metodología de desarrollo de Software.

Esta metodología coge el nombre a que lleva al extremo las prácticas más beneficiosas para la ingeniería del Software.

Es una de las más características entre las llamadas “metodologías ágiles” , ya que su forma de actuar así lo es.

Existen metodologías en las que se separan las etapas de desarrollo y no se puede volver a las anteriores (como las lineales) o que sus tiempos son muy extensos.
XP tiene se asemeja al modelo de espiral a que sus etapas se van repitiendo y una de sus etapas es junto al cliente.
Sin embargo, se parece al incremental a que en cada *“release”* o “incremento” se debate entre todo el equipo y el cliente lo realizado hasta ahora. Esto es esencial, ya que el XP tiene como objetivo conseguir el software con la mayor calidad posible.

Esta metodología tiene cinco valores fundamentales: *comunicación, simplicidad, feedback, valor y respeto.*

El equipo *entero* o *completo* (*“Whole Team”*) lo forman todo aquel que participe en el proyecto, incluido y necesario, un representante del cliente que estará junto al equipo cada día como si fuese uno más.

XP “vive mucho en el presente” , por lo antes mencionado, que los ciclos de entregas son muy cortos. Por esto existe un “planning game”, que consiste en quedar cada dos semanas (o un tiempo similar) el equipo *completo* y discutir sobre lo realizado y sobre lo que no ha podido ser realizado, sus modificaciones … junto con el cliente.
Estudiando el rendimiento obtenido en la entrega recién realizada viendo los requisitos que se han podido cumplir, se planifica la siguiente entrega junto al cliente.

El cliente debe realizar pruebas (*“Customer tests”*) que será la comprobación de los requisitos descritos en el *planning game* anterior.

Aquí se deben decidir aspectos sobre el Software que hay que dar más o menos importancia, ya que el tiempo es crucial para no estancar el proyecto; el cliente debe exigir y adaptar sus necesidades a lo visto.
Por tanto, al final de este tiempo se planifica las tareas, los equipos y el coste de la siguiente entrega.

Otro de los canones de XP es la *simplicidad* (“simple design”) ; el código no debe ser complicado de leer, más vale un código largo pero simple que uno corto y potente que no sea capaz de entenderse. Esto es debido a un problema más que conocido, y es que uno/varios programadores crean un Software de calidad, pero que es tan complicado que es imposible extenderlo o modifcarlo por otro que no estuviese en ese equipo. Esta simplicidad puede verse en aspectos tan comunes como el nombre de las variables. No debe ser un problema tener variables largas como “contador_bucle_conceptos_hotel” si

con ello nos ahorramos leer 80 lineas más para entender su significado por contexto. Los IDE actuales con las funciones de autocompletado nos hacen esta cuestión una trivialidad.

Una de las características que más impacta sobre el resto de metodologías de desarrollo de Software es el trabajo *por parejas* (*"Pair programming"*). Esto puede chocar de primeras, ya que el trabajo de un solo programador esta gastando dos. Esto al comienzo puede no ser beneficioso, pero pasadas unas o dos entregas y se acostumbre el equipo *completo* es algo muy poderoso.
Mientras escribe uno el otro le va siguiendo y discutiendo cada línea que vea sospechosa o extraña. Esto puede describirse como un *debugger in real-time* , es decir, un corrector que llega donde no llega el IDE, a la lógica y funcionamiento del código.

Las parejas se van rotando tras cada entrega. Esto fomenta a cada miembro sepa sobre el trabajo de todos los demás. Por ejemplo, si tras un cambio, un miembro detecta la necesidad de una función que recuerda ya haberla hecho antes, sabe perfectamente como reciclar ese código.

Respecto a esto, también añadir que cualquier equipo de parejas puede modificar cualquier módulo del sistema. Esto no es problema debido a las pruebas unitarias y de regresión que se hace tras cada pequeña entrega.

A la hora de cambiar de equipos y trabajar con otros miembros, cada un tiene su forma de pensar y ver las cosas, así que durante las reuniones, también se habla y modifica la forma en la que se realizará o vivirá la aplicación. Puede verse como una *metáfora* ("*Metaphor*").
Normalmente se recurre a una frase poetica como "las ramas de un arbol" o "las nubes del cielo".

Para esta metodología también nombramos el *valor* (*"courage"*). Este aspecto también es muy importante y complementario, ya que la decisión no la tomas solo tú, sino que también te apoya un compañero. Con valor nos podemos referir a la hora de modificar código. Si estamos seguro de que cambiando algo nuestro sistema puede funcionar mucho mejor, hagamoslo. Si no funcionase o fuese deficiente, en la siguiente entrega (que sera siempre cercana) se le pasaría las pruebas unitarias y de regresión.

Muchas veces un código se modifica y se dejan restos de versiones anteriores por si hubiese que volver atrás. Todo eso eso sobra, ya que quitaría legibilidad al Software y calidad, si decidimos modificar algo, que se aplique de verdad.

Uniendo esto a la simplicidad podemos hablar sobre los comentarios. Comentarios excesivamente largos no sirven, ya que son costosos de modificar después. Cuando se plantea un archivo, durante su creación pueden ocurrir muchas modificaciones que contradigan los comentarios. Por tanto, en XP no hay comentarios, solamente los estrictamente necesarios, como pueden ser las entradas y salidas del archivo. Si aplicamos lo dicho antes sobre las variables, el código debe hablar él mismo.

Añadimos el ultimo valor mencionado, ya que esta relacionado con el valor: el *respeto*.

Cada miembro debe saber valorar cuando tener valorar, no vale romper código de buenas a primeras sin estudiar a priori durante varios casos de que de verdad esa acción va a ser

beneficiosas, ya que eso que vas a cambiar ya fue alguien que penso uno/varios compañero/s tuyo/s y que debes saber valorar. Además, cualquier modificación o mejora que hagas y salga defectuosa hará relentizar el ritmo del proyecto y por tanto el de tus compañeros.