

Metodologías Del Desarrollo de Software

El desarrollo de software es uno de los sectores tecnológicos más competitivos y no es algo nuevo, ya que durante muchos años lo ha sido, sin embargo ha tenido una evolución constante en lo que se refiere a las metodologías o bien, las formas en las cuales se realiza la planeación para el diseño del software, básicamente con el objetivo de mejorar, optimizar procesos y ofrecer una mejor calidad.

Sin embargo, antes de hablar acerca de metodologías y todo este tema tan amplio, analicemos a detalle brevemente ¿Qué es un método? y para que lo acompañemos también veamos ¿qué es una metodología? Seguramente los términos te suenan familiar, sin embargo el saber que significan de forma correcta es indispensable.

¿Qué es un Método?

Un Método se compone de diversos aspectos que nos permitirán conseguir una meta o lograr un objetivo. Se define más claramente como un conjunto de herramientas, las cuales utilizadas mediante las técnicas correctas, permiten la ejecución de procesos que nos llevarán a cumplir los objetivos que buscamos. En pocas palabras y aunque esto lo puedes encontrar como tal en internet, es un conjunto de herramientas, técnicas y procesos que facilitan la obtención de un objetivo.

¿Qué es una Metodología?

En el desarrollo de software, una metodología hace cierto énfasis al entorno en el cual se plantea y estructura el desarrollo de un sistema. Como lo mencioné al principio, existen una gran cantidad de **metodologías de la programación** que se han utilizado desde los tiempos atrás y que con el paso del tiempo han ido evolucionando. Esto se debe principalmente a que no todos los sistemas de la información son compatibles con todas las **metodologías**, pues el **ciclo de vida del software** puede ser variable. Por esta razón, es importante que dependiendo del tipo de software que se vaya a desarrollar, se identifique la **metodología para el diseño de software** idónea.

¿En qué consisten las Metodologías de Desarrollo de Software?

Una Metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Regularmente este tipo de metodología, tienen la

necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto comprendan perfectamente la metodología y en algunos casos el **ciclo de vida del software** que se pretende seguir.

Aunque actualmente existen mucha variedad en **metodologías de programación**. La realidad es que todas están basadas en ciertos enfoques generalistas que se crearon hace muchos años, algunos **tipos de metodologías de desarrollo de software** que se utilizaron e inventaron al principio de nuestra era tecnológica y son las que veremos a continuación.

¿Cuáles son modelos del Ciclo de vida del Software tradicionales?

Como les mencioné hace un momento, regularmente, **cada metodología de desarrollo de software tiene un enfoque bien marcado**, estos enfoques no son para nada nuevos y se siguen utilizando para la planeación y desarrollo de software aún en nuestros tiempos, así que vamos a ver cuáles son cada uno de ellos y aprenderemos cómo funcionan.

1. Metodología en cascada: Framework lineal.

El **modelo de desarrollo de Software** en cascada es una **metodología de la programación** muy antigua. Si bien su creador nunca lo menciona como **metodología en cascada**, el funcionamiento y lineamiento de los procesos de la planeación, son exactamente iguales. Básicamente, el estilo del modelo en cascada es que no podrás avanzar a la siguiente fase, si la anterior no se encuentra totalmente terminada, pues no tiene por qué haber vuelta atrás. Vamos a ver cuáles son las **fases de desarrollo de software** del modelo en cascada, para que te puedas dar una idea.

Etapas Modelo Cascada






Recuperado de:

http://3.bp.blogspot.com/_psAq9gFP8jk/SwY6dVMRYjI/AAAAAAAAAFE/1hNKDdYurqE/s1600/cascada.jpg

1.1. Análisis de Requisitos. El primer nivel del modelo cascada, es el análisis de requisitos. Básicamente lo que se documenta aquí, son los objetivos de lo que el software debe hacer al terminar el desarrollo, sin entrar en detalles de la parte interna, los cuales se verán durante el proceso. Sin embargo es importante señalar que una vez avanzado el paso de análisis, no puede haber vuelta atrás, pues la **metodología para el diseño de software** con la cual se trabaja no lo permitirá.

1.2. Diseño del Sistema. Todo lo que conlleva el armado de un diseño para el sistema que vayas a utilizar, es lo que continua después del análisis de requisitos. Aquí se elaborará lo que es la estructura del sistema y se determinarán las especificaciones para cada una de las partes del sistema que se planea desarrollar. Siendo del mismo modo, una fase a la cuál ya no se podrá volver después de haber bajado al nivel 3.






1.3. Diseño del Programa. En este punto, aún no ingresamos a lo que es la escritura de código, sin embargo ya se realizan los algoritmos que se van a utilizar en la programación. Si bien recuerdas, un algoritmo no necesariamente es código, simplemente tomas una hoja de papel y escribes el algoritmo que vas a utilizar. Esto es precisamente lo que se realiza en el paso número 3.

1.4. Codificación. Seguramente como programador, esta es la parte que estabas esperando, pues ahora es momento de empezar a escribir todo el código que será necesario para el desarrollo del software. Para este punto, la velocidad y el tiempo que se requiera, dependerá mucho del lenguaje de programación que vayas a utilizar. Pues algunos lenguajes de programación permiten utilizar componente, bibliotecas e incluso algunas funciones para reutilizar código, las cuales podrán acelerar el proceso de codificación en gran manera.

1.5. Ejecución de Pruebas. La codificación ha terminado y ahora es momento de verificar que nuestro sistema es realmente funciona, antes de que el cliente empiece a utilizarlo. Este es precisamente el objetivo de la fase 5 de pruebas. Aquí es recomendable que intentes mover lo más que se pueda tu software, con el objetivo de dañarlo intencionalmente, de este modo, si supera las pruebas de daño realizadas por tí, entonces estará listo para el usuario final.

1.6. Verificación. Después de haber realizado una gran cantidad de pruebas en la Fase 5, debemos migrar a la verificación. Esta fase consiste en la ejecución del Software por parte del usuario final. Si la fase cinco se realizó correcta y profundamente, el software no tendrá ningún tipo de problema y el usuario final quedará satisfecho con el resultado.



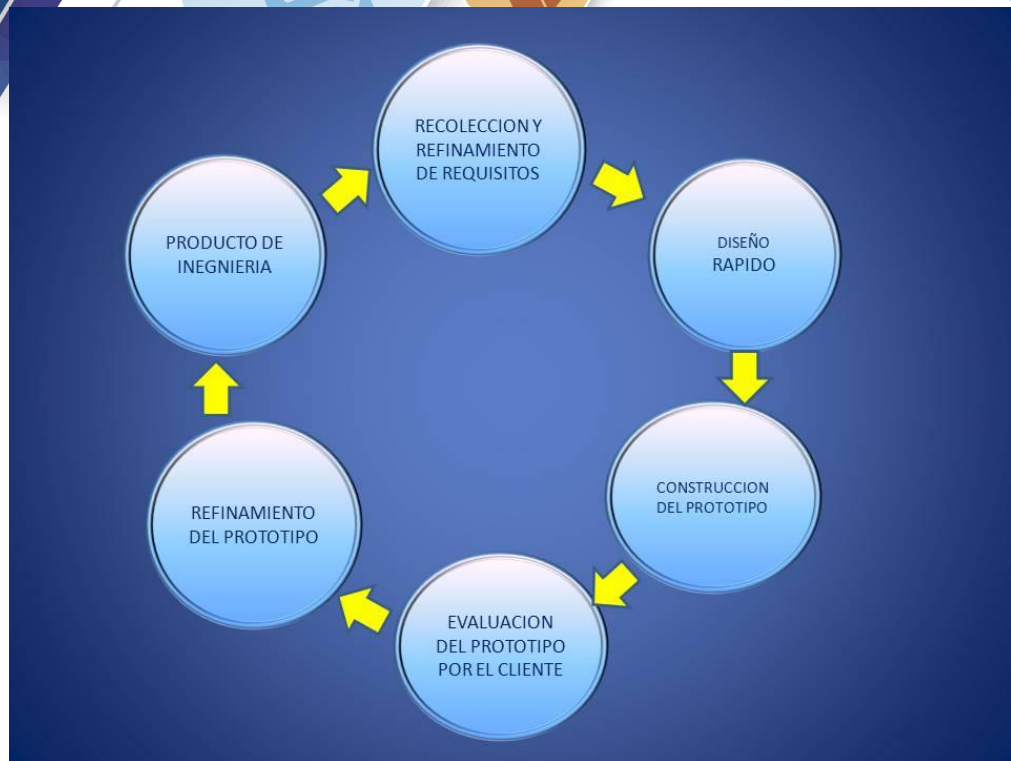
1.7. **Mantenimiento.** Seguramente te has dado cuenta, de que las aplicaciones o el software actual, constantemente se está actualizando. Esto se debe principalmente a que se le da mantenimiento al software, se solucionan errores, se quitan algunos bugs, se añaden funcionalidades, todo después de que el usuario final ya lo ha probado y utilizado en su fase final. Esta es posiblemente una de las fases más tediosas del **modelo de desarrollo de software**, pues debes estar atento a los comentarios de los usuarios, para ver que cosas son las que no funcionan correctamente y a las cuales hay que darles mantenimiento

2. Método de Prototipos

Esta **metodología de la programación** todavía sigue siendo la favorita de muchos. Consiste básicamente en que en base a los requerimientos y necesidades que tiene el cliente, se realiza de forma rápida un prototipo, este no vendrá completo ni mucho menos terminado, pero si permitirá contar con las bases necesarias para que cualquier programador pueda seguir trabajando en el hasta llegar al código final.

Por si no lo sabes aún, un prototipo es una versión no terminada del producto que se le entregará al cliente o usuario final. Esto nos genera cierta ventaja en el desarrollo de productos similares con funciones distintas, por ejemplo. Supongamos que vas a desarrollar un proyecto para 3 clientes distintos, ambos con una estructura idéntica pero con funcionalidades muy distintas, entonces lo que hacemos es crear un prototipo base y entorno a el mostrarlo a nuestros clientes para que de ahí se empiecen a desarrollar las demás funciones.

Mejor vamos a ver cuales son las **etapas de desarrollo de software** por las cuales tendrás que pasar, en caso de utilizar la metodología de prototipos.




Recuperado de: <http://1.bp.blogspot.com/-vHj-IQyj-o0/TaACL9ZhsOI/AAAAAAAAACY/q3NblQxCssg/s1600/PROTOTIPOS01.jpg>

2.1. Planeación. A diferencia de otras metodologías, la planeación debe ser muy rápida, en esta fase no puedes demorarte mucho, pues recuerda que solamente será un prototipo por el momento.

2.2. Modelado. Nuevamente, una fase que deberá ser suficientemente rápida como para que nos nos quite nada de tiempo. Hacer el modelado será simple y te sigo recordando que solamente es un prototipo, al menos por ahora.

2.3. Elaboración del Prototipo. Ya que contamos con la planeación de lo que vamos a realizar y el modelado rápido, entonces es momento de elaborar el prototipo. Para esta instancia, ya no te diré que lo debes hacer rápido, puesto que te tomará el tiempo que tenga sea necesario elaborarlo, recuerda que este ya se muestra al cliente, así que ya es una fase importante.



2.4. Desarrollo. Posterior a contar con el prototipo elaborado y mostrado al cliente, es momento de comenzar el desarrollo. Este te tomará una gran cantidad de tiempo, dependiendo del tamaño del proyecto y el lenguaje de programación que se vaya a utilizar.


2.5. Entrega y Retroalimentación. Una de las cosas con las que cuenta el modelo de prototipos, es que una vez entregado el proyecto, debemos darle al cliente cierta retroalimentación sobre como utilizarlo y ciertamente es una fase que se encuentra dentro de las **etapas de desarrollo de software** esta metodología.

2.6. Comunicación con el Cliente. Es importante que una vez entregado el proyecto, tengamos cierta comunicación con el cliente, básicamente para que nos indique si el proyecto es correcto o si desea agregarle ciertas funciones, nuestra metodología lo permite. Si fuera en modo cascada, entonces sería algo realmente imposible de hacer.

2.7. Entrega del Producto Final. Por último, solamente quedará entregar el sistema elaborado mediante esta metodología. Aquí tendrás la ventaja de que el código es reutilizable, para que así con el prototipo ya puedas simplemente empezar de nuevo y con una buena base de código que te acelerará el proceso.

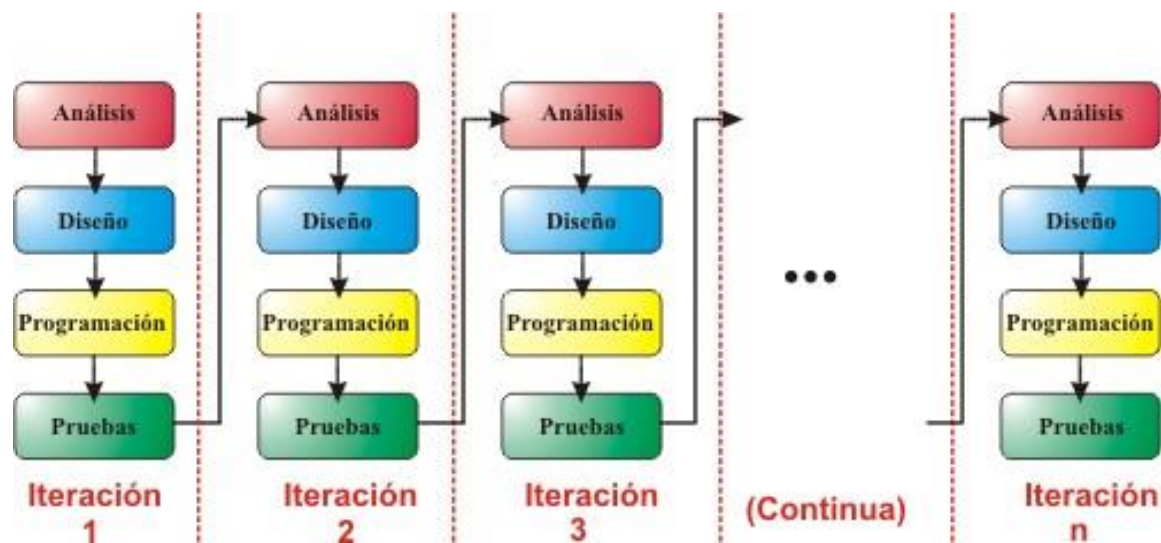
3. Modelo Incremental o Iterativo y Creciente

El modelo Incremental, es una **metodología de la programación** muy utilizada hoy en día, pues su comodidad de desarrollo permite que te obtenga un producto final mucho más completo y exitoso. Se trata especialmente de la combinación de los modelos lineal e iterativo o bien, modelo de cascada y prototipos. Básicamente consiste en completar varias iteraciones de lo que es el modelo de cascada, pero sin completar ninguna, haciendo iteraciones lo que se hace es crear una evolución en el



producto, permitiendo que se agreguen nuevas especificaciones, funcionalidades, opciones, funciones y lo que el usuario requiera después de cada iteración.

- En pocas palabras, el Modelo Incremental repite el modelo de cascada una y otra vez, pero con pequeñas modificaciones o actualizaciones que se le puedan ir agregando al sistema. De este modo el usuario final se ve sumamente sumergido en el desarrollo y puedes proporcionarle un resultado óptimo.



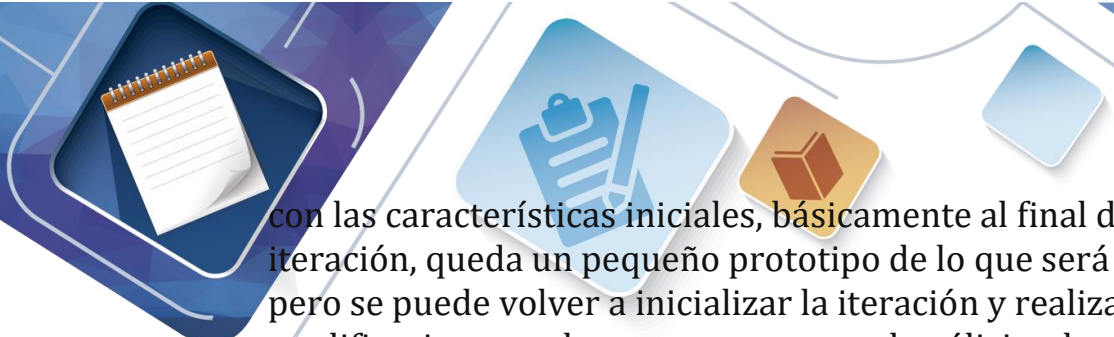
Recuperado de: http://2.bp.blogspot.com/_kgOxUnOZkAE/TJau6zS-XpI/AAAAAAAAADM/rrAX1QeUjU/s1600/iterativo.jpg

Fases del Modelo Incremental

El modelo iterativo o incremental, cuenta con algunas **fases de desarrollo de software** que realmente no tienen mucha complejidad, vamos a verlas:

3.1. Inicialización. como en todo modelo de desarrollo, debe haber una inicialización, aquí se puede hablar de dar una idea, de tener algunos requisitos que se buscan en el proyecto y ciertas especificaciones que se pueden manejar. No es necesario que se haga una lista total de requerimientos pues recordemos que el proyecto se basa en iteraciones, así que el proyecto puede ir evolucionando poco a poco.

3.2. Periodos de Iteración. Durante el desarrollo del proyecto, es cuando damos inicio a las iteraciones. La primera iteración se realiza




con las características iniciales, básicamente al final de la primer iteración, queda un pequeño prototipo de lo que será el proyecto, pero se puede volver a inicializar la iteración y realizar modificaciones en los procesos, como el análisis y las especificaciones o funciones que el usuario final requiere para su sistema. El número de iteraciones que se realicen son ilimitadas y dependerá tanto del desarrollador como del usuario final. Si nuestro objetivo es que el cliente o la persona que necesita el trabajo quede completamente satisfecha, entonces nos veremos en la necesidad de hacer la cantidad de iteraciones que se requieran al final del día.

3.3. Lista de Control. Es importante que conforme se vaya realizando cada iteración, se vaya llevando un control del mismo en una lista. Como si fuera un programa que recibe actualizaciones constantemente. Cada una de las actualizaciones o iteraciones deberá ser documentada y de ser posible, guardada en sus respectivas versiones, para que sea sencillo volver atrás, en caso de que una iteración no sea exitosa o el usuario ya no la requiera.

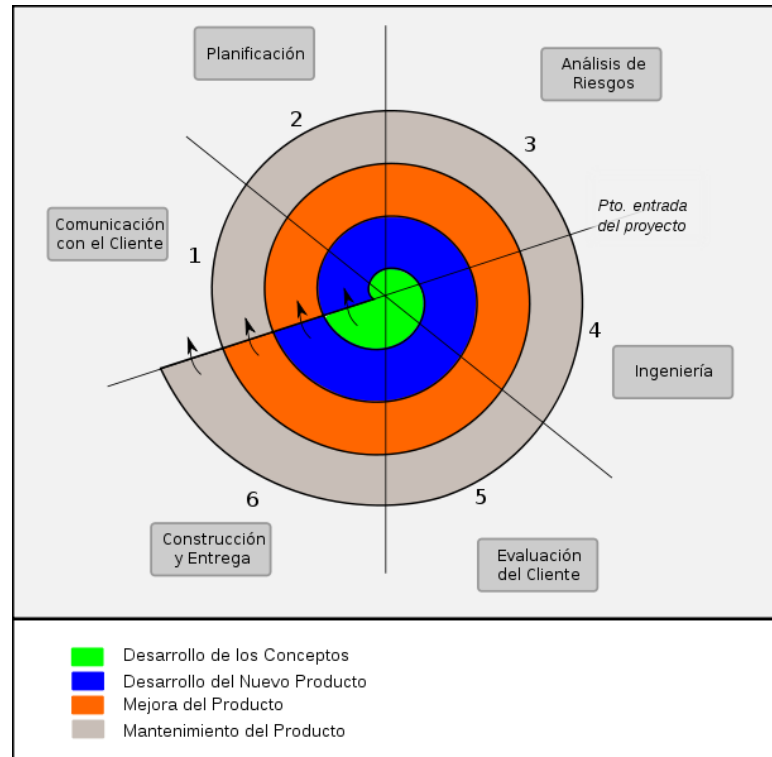
4. Modelo en Espiral

El modelo en espiral, fue utilizado y diseñado por primera vez por Barry Boehm en 1986. Se trata nuevamente de una combinación entre el modelo lineal o de cascada y el modelo iterativo o basado en prototipos, sin embargo a este sistema lo que debemos añadirle es la gestión de riesgos, algo que en los modelos anteriores ni siquiera se menciona.

Este modelo, **consiste en ciertas fases que se van realizando en modo de espiral, utilizando procesos de la misma forma en que se utilizan en el modelo de cascada**, sin embargo aquí estos no son obligatorios y no llevan precisamente el orden establecido. Básicamente se trata de un modelo evolutivo, que conforme avancen los ciclos, irá incrementando el nivel de código fuente desarrollado, un incremento en la gestión de riesgos y por supuesto un incremento en los tiempos de ejecución y planificación del sistema, esto es lo que tiene el modelo en espiral.



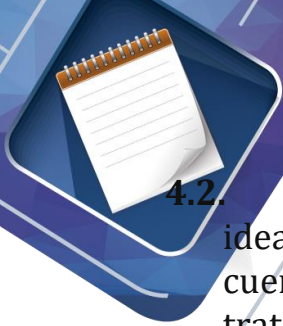


Para que tengas una idea más clara, **el modelo en espiral es principalmente utilizado para el desarrollo de grandes proyectos como la creación de un sistema operativo**. Sin embargo necesitas de ciertos requisitos, como el hecho de contar con personal completamente capacitado para las funciones que se requieran. Mejor veamos cuales son las las fases o tareas dentro del modelo de espiral.



Recuperado de:

http://1.bp.blogspot.com/_S5W0bf4uECM/TL5u0oqW4MI/AAAAAAAAABI/wk8pbHQALHU/s1600/espiral2.png


4.1. Determinar Objetivo. Es importante que siempre consideres una planeación inicial, esta solo se realizará una vez. Sin embargo el proceso de determinar objetivos se hará constantemente durante cada iteración que se vaya realizando con el modelo espiral. Esto se debe a que poco a poco se irá incrementando más el tamaño del manual de usuario, los requisitos, las especificaciones e incluso las restricciones. Todo esto entra en lo que es la tarea de objetivos y con cada vuelta en el espiral entraremos a esta tarea, la cual como todas las demás, es fundamental.

4.2. Análisis de Riesgo. Una etapa donde incluso una lluvia de ideas podría ayudar, el análisis de riesgos. Aquí deberás tener en cuenta todo aquello que pueda dañar a tu proyecto, ya sea que se trate de ciertas amenazas o de posibles daños que se puedan ocasionar, teniendo además un Plan B por así decirlo, para que en caso de que ocurra algo inesperado, tener a la mano la solución para continuar con el proyecto. En esta fase del modelo espiral, podemos agregar lo que son la creación de prototipos, pues siempre es bueno tener un respaldo de nuestro código, se esta forma en caso de que algo malo suceda, volvemos a la versión anterior. Así que cada vez que vayamos a ingresar a la fase de pruebas e implementación, será necesario contar con un prototipo que nos respalde.

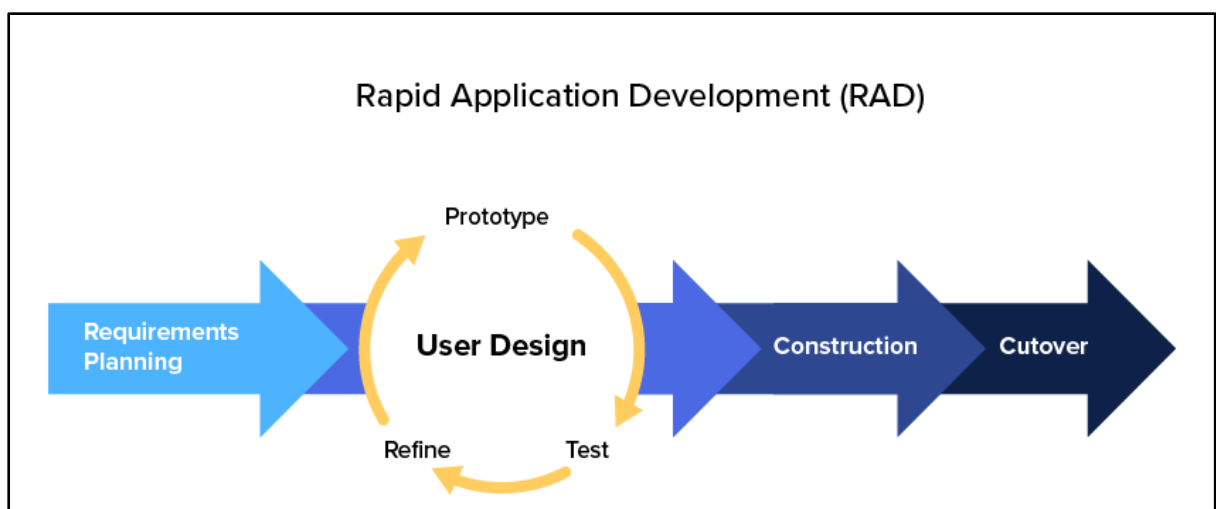
4.3. Desarrollar, Validar y Probar. Básicamente en esta fase, la forma en que se estará desarrollando el proyecto, dependerá del análisis de riesgos, pues siempre se va a ir desarrollando el proyecto enfocándose en los riesgos que podemos evitar en nuestro software, es decir, si la situación de riesgo más obvia se encuentra en la interfaz del usuario, entonces hay que trabajar con prototipos para este enfoque, creando evoluciones proporcionales, para ir evitando ese tipo de riesgos. Esto no significa que ignoremos el resto del proyecto o del desarrollo, sin embargo el modelo en espiral si acomoda un poco más las prioridades al momento, independientemente de todo lo demás. Por lo que siempre en cada vuelta o iteración que se le de al modelo de espiral, tu avance siempre dependerá del análisis de riesgo, hasta que este sea mínimo y el desarrollo pueda continuar de forma normal.

4.4. Planificación. Antes de proceder a realizar otra iteración o vuelta al espiral, debemos prestar atención a lo que sucedió en la vuelta anterior. Debemos analizar detalladamente si los riesgos encontrados ya tuvieron solución, lo cual debe ser lo ideal, puesto que ahora habría que analizar más especificaciones y más requisitos del sistema para continuar con el desarrollo. Básicamente la fase de planificación, nos servirá para determinar el avance de nuestro proyecto y indicar hacia donde vamos a dirigirnos en la próxima iteración.



5. RAD: Desarrollo Rápido de Aplicaciones (Rapid Application Development)

A diferencia de otras **metodologías para el desarrollo de software**, la **metodología RAD** o desarrollo rápido de aplicaciones, no cuenta con una serie de fases ordenadas por así decirlo. Aunque si está basada en lo que es el modelo de cascada y la creación de prototipos, sin embargo el proceso es muy independiente a contar con ciertas fases estipuladas como los modelos que hemos visto anteriormente. Así que vamos a ver los principios del modelo RAD.




Recuperado de: <https://kissflow.com/wp-content/uploads/2018/07/Rapid-application-development>

¿Cuáles son los principios básicos del Modelo RAD?

Del mismo modos que modelos anteriores, **el Modelo RAD, está basado en el uso de las iteraciones y principalmente en el manejo de prototipos**. Sin embargo a diferencia del resto, la **metodología RAD** hace uso de las Herramientas CASE, las cuales permitirán acelerar el proceso considerablemente.

Del mismo modo que el modelos espiral y el de prototipos, **RAD se subdivide en pequeñas secciones, las cuales se irán optimizando y de esta forma se irá avanzando mucho más rápido** que con grandes segmentos que pueden volverse difíciles dentro de un proceso acelerado como lo este este modelo.



Una de las ventajas del RAD, es que el enfoque y las prioridades van hacia la fase de desarrollo, a diferencia de modelos como el espiral, que se enfoca en que los riesgos al momento sean mucho menores. Acá con el RAD, se hace lo contrario, si hay riesgos reducimos los requerimientos para reducir los riesgos, no como en el espiral, que entre más riesgos, más requisitos aportamos para que se incremente. Acá la idea es reducir tiempos y no riesgos, o si, talves también, pero la reducción de riesgos es totalmente inversa al modelo espiral.

Por supuesto, como en todos los modelos, vas a requerir de ciertos factores documentados, de preferencia todo lo que se pueda, para que en caso de que alguien más venga a continuar con este proyecto, tenga a la mano toda la información que necesita y con unas cuentas lecturas pueda empezar a desarrollar lo que se había quedado pendiente en un determinado momento.


¿Cuál es tu metodología tradicional preferida?

Si bien hemos visto métodos muy diversos, información muy variada y **ciclos de desarrollo de software** con distintos enfoques. La realidad es que al final tu siempre decidirás como trabajar y con que tipo de metodología te sentirías más a gusto. Obviamente depende de muchos factores, principalmente del tamaño del proyecto, pues modelos como el espiral, son especialmente para proyectos grandes y modelos como el RAD, son enfocados en proyectos pequeños, sin tanto requisito pero manejando siempre cierto nivel de calidad, el cual siempre debes considerar.

A continuación, **analizaremos lo que son las metodologías ágiles de desarrollo de software más importantes**, las cuáles a diferencia de las metodologías tradicionales, funcionan mas como una combinación de estas para lograr un objetivo. Su finalidad siempre será el crear software de una forma más rapida de lo que se venia logrando con las metodologías de antaño. Así que vamos a ver un poco de información referente a lo que son las metodologías ágiles, como funcionan y que se necesita para implementarlas.

6. Metodologías Ágiles

Con el paso del tiempo, estaba claro que las metodologías tradicionales, simplemente no se iban a acoplar con las nuevas tecnologías, los nuevos



lenguajes y sobretodo los programadores modernos. Es por eso que **desde principios del Siglo, se han desarrollado lo que son las metodologías ágiles**. Una metodología ágil, consiste principalmente en trabajar con menos documentación de la que, como vimos, las metodologías tradicionales utilizan en todo momento.


Existen una gran cantidad de **metodologías ágiles de desarrollo de software** y todas las vamos a ver a continuación. Sin embargo antes hay que comprender en que consiste detenidamente la metodología ágil, para lo cual contamos con el manifiesto ágil. Un documento en el cuál se resume la filosofía de este enfoque de desarrollo, así seguramente después de leer esos puntos, nos quedará aún mas clara la idea de hacia donde se pretende llegar y principalmente Cómo se pretende llegar a los objetivos.



Recuperado de: <https://iswugaps2crystalclear.files.wordpress.com/2014/09/fondo.png>

Manifiesto Ágil

Al Individuo y las Interacciones del Equipo. Una de las cosas que sabemos muy bien, es que las personas son quienes consiguen los éxitos dentro de un proyecto de software. Sin embargo también está claro que si el equipo de trabajo falla, entonces todo se viene abajo y el enfoque




que había de éxito se convierte en incertidumbre. A diferencia de si el equipo funciona perfectamente, se tienen mas cerca los objetivos, aún cuando no exista una lista de procesos establecida como se hacia con las metodologías de antaño.

Con este primer valor del **desarrollo ágil**, se pretender hacer ver, que en realidad no importa que el equipo de trabajo no sean las personas más brillantes del sector. Con que sean personas que saben hacer bien el trabajo que se les asignará es más que suficiente. En este caso, las herramientas aunque son importantes para incrementar el rendimiento, también es cierto que si hay herramientas de más y que no sirven de nada en un principio, lo mejor es dejarlas de lado o estas nos quitarán valioso tiempo que no tendremos.

Básicamente el enfoque, es que no se intente crear un entorno de trabajo desde un principio, tratando de que los desarrolladores se adapten a ese entorno que nosotros deseamos, pues este tipo de proyectos suelen tardar mucho tiempo en desarrollarse, algunos incluso jamás terminan y se quedan estancados. El enfoque del **desarrollo ágil** nos dice, que es mejor formar primero un buen equipo de trabajo y posteriormente entre ellos vayan creando su propio entorno. Este proceso ayudará mucho más a la metodología ágil y por supuesto, la adaptación será un proceso fugaz.

Software funcional en lugar de demasiada documentación. Crear documentación, es una de las actividades que con las metodologías tradicionales, absorbían una gran cantidad de tiempo. Si nos acercamos a analizar las metodologías de antaño, descubriremos que en cada una de ellas, realizar la documentación era una parte vital. Sin embargo en las **metodologías ágiles**, esto ha cambiado, pues existe una regla que dice de la siguientes forma: “No producir documentación, almenos que sean sumamente necesarios al momento para tomar una decisión”. Por lo que además se extiende hacia el enfoque de que la documentación debe ser corta y breve, nada sumamente extenso que requiera de una gran cantidad de tiempo de lectura.

Colaboración con el Cliente en lugar de hacer Contrato. Una de las cosas importantes dentro de las **metodologías ágiles**. Es que cambia el modo en que se trabajaba con el cliente anteriormente. Y es que en las metodologías de antaño, el trabajo consistía en tener una reunión previa con el cliente para analizar los requerimientos del sistema, aquí se analizaban las limitaciones del proyecto y se establecían los costos. Lo que generaba cierta barrera de bloqueo entre las posibilidades de



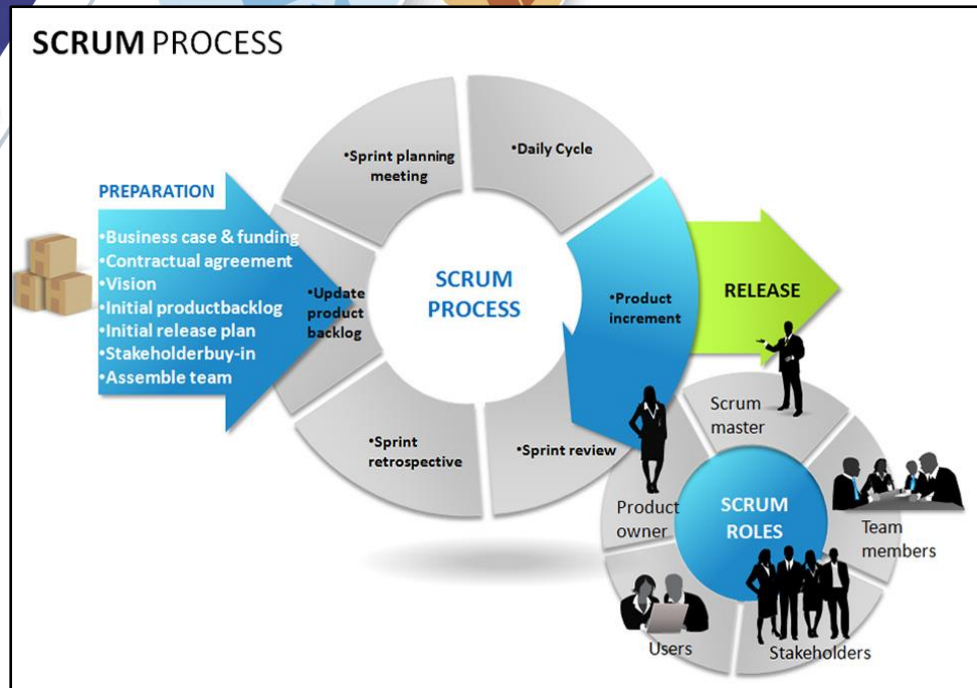
desarrollar algo funcional para otras cosas o solo para el objetivo establecido en la reunión. Esto al final podía ser desastroso y dificultar la llegada al éxito por parte del proyecto.

✓ **Posibilidad de Hacer cambios de planes a medio proyecto.** Suena mas o menos a lo que vimos en el punto anterior, pues básicamente **la idea es evitar lo que es la planeación extensa y empezar a crear código que permita expansión.** Recordemos que con las metodologías tradicionales, se acostumbraba a enlistar los requisitos del sistema y el desarrollo iba enfocado solamente a eso, lo cual ya no permitía que a medio desarrollo hubiera cambios, pues era un código poco moldeable y si se requerían nuevas cosas, en algunas metodologías lo idea era volver a empezar.

7. Metodología Scrum

Para que tengas una idea rápida, para que un proyecto ingrese al marco de lo que es el **modelo Scrum**, debe contar con las siguientes características:

- ✓ **Desarrollo Incremental.** Una metodología ágil sin desarrollo incremental, no puede ser considerada Scrum. Con incremental hago énfasis a olvidarnos de la planeación y de la ejecución de las líneas sin salirnos de lo pre establecido, pues con una metodología Scrum, el desarrollo se irá incrementando poco a poco, sin importar el orden en el cual se lleven a cabo los procesos.



Recuperado de: <https://www.cprime.com/wp-content/static/images/resources/ScrumOverviewResized.jpg?ca3ce9>

- ✓ **Calidad de las personas.** Básicamente la calidad de un producto, no será analizada en base a la calidad de cada uno de los procesos llevados a cabo. Al contrario, la calidad dependerá de las personas, la auto organización y el conocimiento de los equipos de trabajo.

- ✓ **Adiós al Secuencial y Cascada.** Aquí en el **modelo Scrum**, hay algo a lo que se le denomina, solapamiento. Esto consiste en que no importa en que proceso te encuentres, si un proceso necesita ser trabajado, vuelves a el para realizar lo que tienes que hacer, a diferencia de las metodologías cascada o secuencial, donde no había vuelta atrás. Acá afortunadamente no hay ningún problema con eso y la ventaja es que se ahorran tiempos.

- ✓ **La comunicación es Fundamental.** Una de las cosas que se realizan, son los equipos de trabajo, sin embargo acá la ventaja que tendrás es que podrás estar en constante comunicación con los otros equipos de trabajo, nadie está envuelto en su propia burbuja y toda la información que se maneje o lleve a cabo, será comunicada sin problema.

¿Cómo funcionan los Procesos Scrum?

La metodología Scrum, es bastante amigable y fomenta lo que es el trabajo en equipo en todo momento, con la finalidad de conseguir los objetivos de una forma rápida. Veamos ahora cuales son los procesos con los cuales funciona la metodología, empezando por el Product Backlog, el cual nos permitirá llegar a los Sprints, a continuación te explicaré de que te estoy hablando.

- ✓ **Product Backlog.** El Product Backlog no es más que una lista de las funcionalidades del producto a desarrollar. Este debe ser elaborado por el Product Owner, puesto que más adelante les explicaré. Sin embargo no se trata de una lista cualquiera hecha con escritos y nadamás. El Product Backlog debe estar ordenado de acuerdo a las prioridades del sistema de mas a menos, con la idea de que las cosas con mayor prioridad sean las que se realicen antes de cualquier cosa. De forma concreta, digamos que el objetivo base del Product Owner, es que nos de respuesta a la pregunta “¿Qué hay que hacer?”.
- ✓ **Sprint Backlog.** Una ves que ya contamos con el Product Backlog terminado, entonces aparecerá el primer **Sprint Backlog**. Pero ¿Qué es el **Sprint Backlog**? Consiste básicamente en seleccionar algunos de los puntos escritos en el Product Backlog, los cuales procederán a ser realizados. Sin embargo en este punto el Sprint Backlog tiene como requisito marcar el tiempo en que se llevará a cabo el Sprint.
- ✓ **Sprint Planning Meeting.** Antes de iniciar un Sprint, el cual es la fase de desarrollo, se realiza lo que es un Sprint Planning Meeting. En este proceso del Scrum, es una reunión que se realiza para definir plazos y procesos a efectuarse para el proyecto establecido en el Product Backlog. Algo importante que debes saber, es que cada Sprint, se compone de diversos features, que no son otra cosa mas que procesos o subprocesos que se deben realizar, puede ser la creación de un logo, la gestión de contenido, el diseño visual, etc. Todo dependerá del proceso que se desee llevar a cabo.

✓ **Daily Scrum o Stand-up Meeting.** Cuando un Sprint está en proceso, después de haber hecho la planeación del proyecto mediante plazos y procesos, entonces entramos a lo que son los Daily Scrum o Stand-up Meeting. Aquí básicamente lo que se hace son reuniones diarias mientras se está llevando a cabo un Sprint, para responder las siguientes preguntas: ¿Que hice ayer?, ¿Qué voy a hacer hoy, ¿Qué ayuda necesito?. Aquí entra en función el Scrum Master, un puesto que igual mas adelante les explicaré. Pero el será el encargado de determinar la solución de los problemas y cada complicación que suceda.

✓ **Sprint Review.** El Sprint Review, es básicamente una reseña de lo que fue el Sprint. Consiste específicamente en la revisión del Sprint terminado y para este punto ya tendría que haber algo que mostrarle al cliente, algo realmente visual o tangible para que se pueda analizar un cierto avance.

✓ **Sprint Retrospective.** Para concluir, el Sprint Retrospective, permite al equipo analizar los objetivos cumplidos, si se cometieron errores, visualizarlos y tratar de no cometerlos nuevamente mas adelante. Básicamente también sirve este proceso para lo que son la implementación de mejoras.

Equipos que Componen los Procesos Scrum

Durante el punto anterior, te describí los procesos que se llevan a cabo en la **Scrum Metodología**, y en varios puntos mencioné ciertos equipos que son encargados de algunos aspectos importantes. Por eso a continuación veremos cuales son los equipos que conforman la metodología Scrum y con los cuales se trabajará arduamente, obvio, cada quien con sus respectivas responsabilidades.

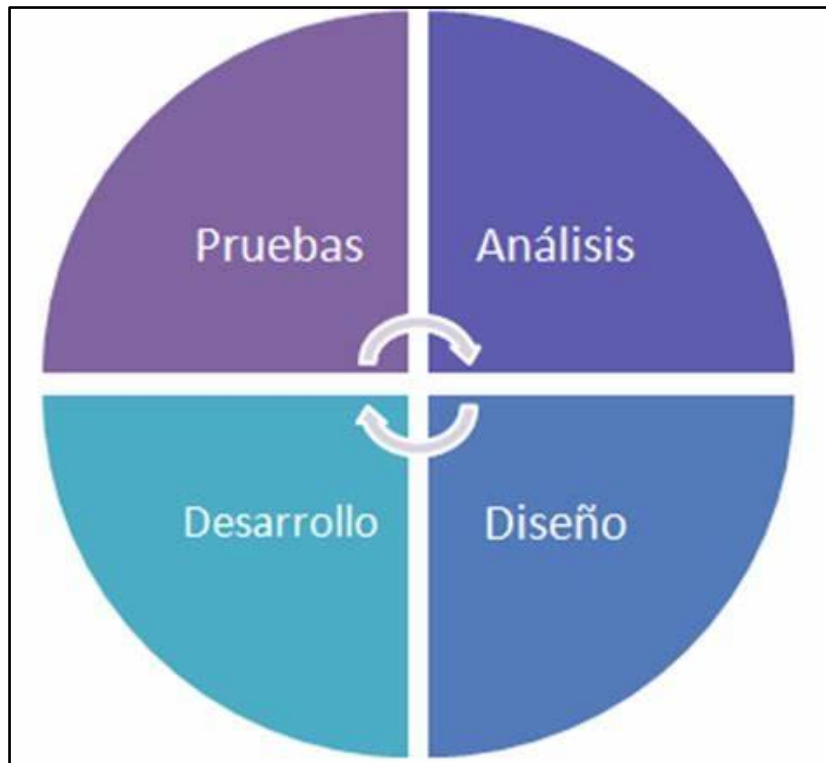
✓ **Product Owner.** Si se trata de tener un líder de proyecto, entonces el Product Owner lo será. Básicamente son los ojos del cliente, será la persona encargada del proyecto y de visorear que se lleve a cabo de tal forma que cumpla las expectativas de lo que se espera.

- ✓ **Scrum Master.** Ahora bien, para cada reunión realizada, siempre debe estar un líder, en este caso el Scrum Master será el líder de cada una de las reuniones y ayudará en los problemas que hayan surgido. Será básicamente como un “facilitador” el cual minimizará obstáculos, sin embargo no los omitirá. En realidad el Scrum Master debe ser una persona empapada de conocimientos sobre el lenguaje o lenguajes bajo los cuales se llevará a cabo el proyecto, de lo contrario no tendría como ayudar a solucionar problemas.
- ✓ **Scrum Team.** Básicamente es el núcleo de la metodología Scrum, pues es el equipo de desarrollo, encargado de lo que es la codificación del software y de cumplir los objetivos o metas propuestas por el Product Owner.
- ✓ **Cliente.** Aunque no lo creas, el cliente también forma parte del equipo, hablamos de eso hace un rato, cuando comenté que no es como en las metodologías tradicionales donde al cliente se le pedían requerimientos y se le daba un costo total. En la metodología Scrum, el cliente tiene la capacidad para influir en el proceso, debido a que siempre estará empapado de el, ya sea que proponga nuevas ideas o bien haciendo algún tipo de comentario.

8. Metodología XP

Si hablamos de metodologías de la programación sin mencionar a la Metodología XP, es como no hablar de nada en absoluto. **Esta metodología es posiblemente la mas destacada de las metodologías ágiles y esto se debe a su gran capacidad de adaptación ante cualquier tipo de imprevisto que surja.** Pues la idea no es mantener ciertos requisitos desde que se está elaborando el proyecto, sino que durante el proceso, estos vayan cambiando o vayan evolucionando gradualmente sin complicaciones. Básicamente los creadores de esta metodología XP, consideran que es mejor adaptarte en el proceso a los requisitos que vayan apareciendo, que iniciar con requisitos y desarrollar un proyecto en base a eso.

Si queremos ver la metodología con otra perspectiva, se podría decir que la metodología XP o **metodología de programación** extrema, como también se le conoce. **Es la combinación de las demás metodologías, solamente que se van utilizando de acuerdo a como sea necesario**, por eso es considerada como la más destacada de las metodologías ágiles. Así que es momento de entrar en detalles y vamos a ver cuales son los valores que conforman a la **metodología de programación XP**.




Recuperado de: <https://managementplaza.es/wp-content/uploads/2016/07/Ciclo-XP.png>

Valores de la Metodología XP

Como toda metodología, **la programación extrema cuenta con algunos valores que son fundamentales para que se lleve a cabo como debe ser**. En algunas otras metodologías estos puntos los conocíamos como principios básicos, es realmente lo mismo solo que acá los mencionan como valores, veamos cuales son.


8.1. Comunicación. Del mismo modo que otras metodologías como la Scrum, el cliente tiene una gran intervención, pero obviamente la comunicación dependerá de mas factores. Por ejemplo. El código fuente de los programadores debe transmitir esa comunicación a todo el equipo, por eso las variables amigables. De igual forma, se

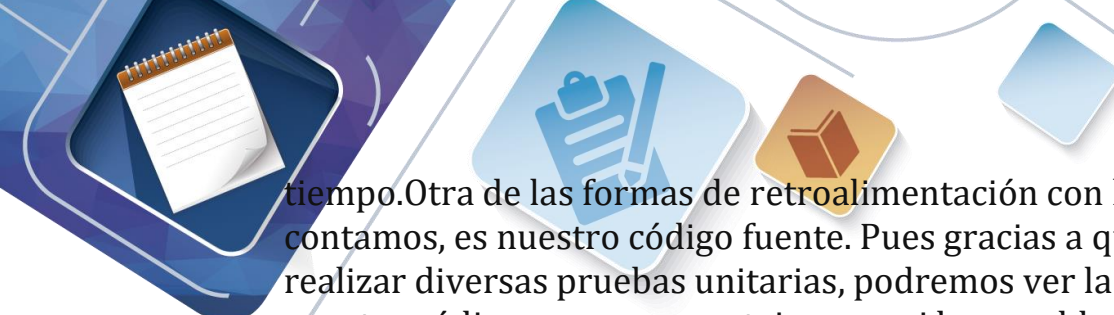


deben documentar las cosas mas relevantes, independientemente de que sean comentadas en el código, pero es importante tener un documento extra para explicaciones extensas, de lo contrario el código se verá infestado de escrito. En cuando a la comunicación entre personas, los programadores se comunican constantemente ya que trabajan en parejas, la comunicación que se tiene con el cliente debe ser constante, pues recordemos que incluso el forma parte del equipo de trabajo y es responsabilidad del cliente, comunicarnos algunas actualizaciones que requiera en el proceso, nuevas ideas, soluciones a problemas o sencillamente algún problema que el vea. Todo debe ser comunicado, esta parte es realmente fundamental para el desarrollo de un producto exitoso.

8.2. Simplicidad. El primero de los valores de la **metodología de programación XP**, es la simplicidad. Ya te haz de imaginar en que consiste, puesto que la idea es que el desarrollo sea velóz, por lo cual todas las cuestiones de diseño se simplifican al máximo, lo mismo sucede con las lineas de código, si se pueden simplificar, se hacen, además de que regularmente el mismo código es donde va la documentación comentada, de esta forma nos evitamos el estar haciendo documentación extra. Por esta razón, además es importante que en el **ciclo de desarrollo de software** mediante la metodología XP, las variables, métodos y clases, tengan nombres amigables y relacionados, de este modo no solamente se ayuda el equipo de trabajo, el cual de por si ya se debe conformar de dos personas por equipo, si no que además, cuando una persona nueva ingrese al proyecto, será muy rápida su adaptación.

8.3. Retroalimentación. El hecho de que el cliente se encuentre involucrado en el proyecto, ayudará inicialmente con la retroalimentación. Pues conforme pasan los días y se va analizando el código por pequeñas etapas, el cliente puede ir corrigiendo, agregando, quitando o excluyendo algunas cosas, esa es la ventaja de la programación por periodos cortos de tiempo, es decir, imagina que llevas varios meses desarrollando el proyecto y cuando vas con el cliente, el proyecto no le gusta y desea hacer tantos cambios que te llevará una eternidad. Precisamente eso es lo que se trata de evitar con la metodología XP, que se tiren varios meses de trabajo a la basura y mejor se vaya optimizando en pequeños lapsos de





tiempo. Otra de las formas de retroalimentación con la cual contamos, es nuestro código fuente. Pues gracias a que podemos realizar diversas pruebas unitarias, podremos ver la salud de nuestro código, una gran ventaja, pues si hay problemas o errores, siempre estaremos a tiempo de realizar modificaciones y soluciones. A diferencia de un proyecto sumamente extenso, donde seguramente la cantidad de errores será tan impresionante que volver a empezar podrá ser una opción.

8.4. Valentía. Hay elementos donde el coraje o la valentía de los programadores será fundamental. Por ejemplo el dar solución a los problemas frente a los cuales se enfrente. El pasar por la eliminación de código fuente en el programa desarrollado, aún cuando todo ese código haya tomado una gran cantidad de tiempo en hacerse o que tal el hecho de diseñar para hoy y no para mañana. Muchos lo hacen con esa idea en ocasiones, pero con un poco de valentía y coraje que forman parte de los valores de la metodología, seguramente el éxito llegará de manera anticipada.

8.5. Respeto. Originalmente, este quinto valor no se encontraba en la metodología XP, sin embargo es importante mencionarlo pues hoy en día ya lo conforma. El respeto es importante para que haya una buena comunión entre los programadores del equipo. Nunca hay que denigrar a nadie ni agregar o ofender, pues una autoestima alta en el equipo garantizará un trabajo mucho más eficiente. Por esta razón, cosas como el código fuente, las modificaciones, los fallos obtenidos, los problemas o la solución de problemas, son procesos que se deben respetar para que el ambiente de trabajo también sea óptimo.

Referencias Bibliográficas

Anonimo. (s.f.). *OK HOSTING*. Obtenido de <https://okhosting.com/blog/metodologias-del-desarrollo-de-software/>

