Tema 3. PRUEBAS CONTRA ESPECIFICACIÓN DE SOFTWARE

El desarrollo de software, es una de las áreas de la tecnología a donde muchos ingenieros y licenciados en informática suelen acercarse. También denominado como el proceso del desarrollo de software, el ciclo de vida del software nos permite entablar una serie de procedimientos mediante las cuales se procede para la realización del mismo. De hecho gracias a este término, es que se han ido creando los métodos del ciclo del software, que no son otra cosa más que metodologías que indican distintos pasos a seguir para el desarrollo de un producto. Es por eso que a continuación, vamos a ver algunos **ejemplos del ciclo de vida del software basándonos en diversos términos**, autores, definiciones e incluso metodologías de gran relevancia que durante muchos años y hasta el día de hoy, se siguen usando.

Proceso Básico del Ciclo de Vida de un Sistema

Si bien, es cierto que **existen diversas metodologías y formas de desarrollar software, la realidad es que hay modelos tan antiguos que ya son como básicos al momento del ciclo de vida de un software**. Un ejemplo de esto, es el modelo en cascada para el proceso de desarrollo de un sistema, con el cuál veremos a ciencia cierta el proceso básico, del cual muchos modelos más se empezarán a desarrollar.

• **Planificación**. El primer punto importante en el ciclo de vida de software, es analizar brevemente los requerimientos que el cliente pide para la elaboración del sistema que necesita. Esta etapa requiere se cierto conocimiento para poder entender la idea que el cliente propone, además de que regularmente debes tomar nota con cada uno de los puntos importantes que se te solicitan, de este modo puedes hacer una planificación al momento y llegar incluso a determinar los tiempos de desarrollo que te llevará, antes de proceder a entregar el producto final. Un punto importante por el cual la planificación siempre debe estar en los ciclos de vida del software. Es porque el cliente se imagina su producto final de una forma tan abstracta, que necesitas hacer que ponga los pies en el suelo para obtener resultados que se acerquen más a la realidad.

• **Implementación**. Una vez que hemos platicado con el cliente y tenemos lo que es un análisis de requerimientos, necesidades y funcionalidades por parte de una aceptación en ambas partes, entonces procedemos con lo que es el ciclo de vida de desarrollo de software. Para este punto, existen una infinidad de metodologías de desarrollo de software, que nos ofrecen la posibilidad de trabajar de distintas formas. Más adelante hablaremos más específicamente de ellas, sin embargo la implementación, es básicamente la parte donde los programadores empiezan a codificar o desarrollar el sistema que se necesita, básicamente se trata del ciclo de vida del desarrollo de sistemas, sin importar el lenguaje de programación mediante el cual se vayan a elaborar.

• **Pruebas**. Una vez que el sistema se va desarrollando, es importante para el ciclo de vida del desarrollo del software, que se realicen ciertas pruebas conforme se vaya avanzando. La idea es que no se termine el desarrollo para poder hacer pruebas, si no que mucho antes, durante el proceso de creación, estas ya se puedan ir ejecutando. Las pruebas nos van a permitir ver si el sistema que se está desarrollando es funcional, si tiene algunos errores, si le faltan ciertas cosas para funcionar correctamente, pues básicamente para avanzar al siguiente punto del ciclo de desarrollo de software, será necesario haber pasado las pruebas correctamente.

• **Documentación**. Muchas metodologías de lo que es el ciclo de vida software, van creando documentación, conforme se va avanzando en el desarrollo del sistema. Sin embargo algunas otras prefieren no hacer la documentación hasta el final. Ahora sí que sea cual sea la metodología que elijas, la documentación siempre será importante, pues considera que no siempre vas a estar tú y tu equipo disponibles y cuando otro equipo llegue a programar lo que ustedes hicieron, será indispensable que haya una documentación de la cual se puedan basar, para poder empezar a desarrollar nuevamente el sistema incompleto.

• **Despliegue**. Ya casi llegando a lo que son las últimas etapas del desarrollo de software, nos encontramos con el Despliegue. Este no es otra cosa, más que el momento en que el sistema ya está terminado y ha sido aprobado para que se elabore el producto final. ahora será el momento de distribuirlo y celebrar, pues gracias al equipo de trabajo es como se habrá llegado a esta fase. Lamentablemente, de las etapas de desarrollo de software, esta es a la cual muchos nunca llegan. Pues una gran cantidad de software incompleto se queda en el camino debido a distintos puntos o motivos. Puede ser que el equipo no se unió, el cliente declinó, el proyecto no fue funcional, etc. Así que de haber llegado a esta fase de desarrollo de software, tu como tu equipo deberán sentirse orgullosos y es momento de volver a desarrollar un proyecto más.

• **Mantenimiento**. La última de las fases del desarrollo de software, es el mantenimiento. Que creías, que nunca más verías al software que hicieron, terminaron y distribuyeron. Pues claro que si lo volverías a ver, pues es momento de darle mantenimiento. Acá además se pueden agregar lo que son las actualizaciones, dependiendo del tipo de desarrollo. Si el equipo siguió trabajando con el software desarrollado y encontraron formas de hacerle mejoras, entonces parte del mantenimiento será actualizarlo a la versión final en todo momento.

Paradigmas de los Modelos del Ciclo de Vida del Software

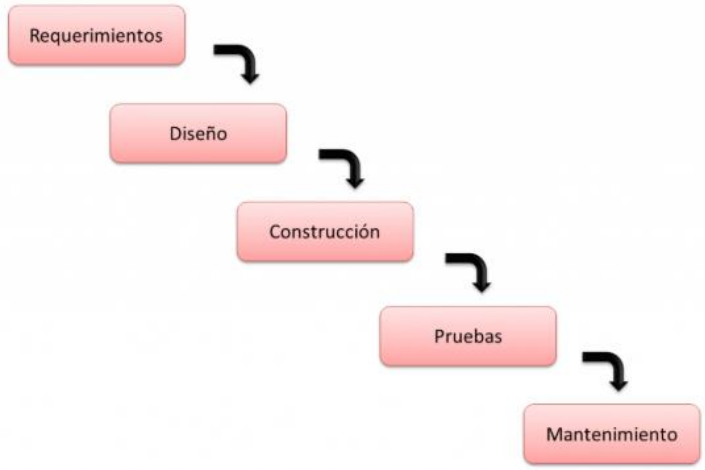
Una de las cosas principales, que se deben elegir al momento de empezar un proyecto de desarrollo de software, son precisamente las etapas del desarrollo de software. Si bien, nos queda claro que no todos tenemos las mismas ideas y no todos pensamos de la misma manera, afortunadamente ya existen modelos preestablecidos bajo los cuales podemos elaborar nuestro proyecto. Es por eso que a continuación les mostraré cuales son algunos de los paradigmas de los Modelos del ciclo de vida de desarrollo de sistemas. Bajo los cuales podemos encontrar una gran cantidad de modelos distintos para desarrollar software, veamos.

• **Paradigma Tradicional**. Existen algunas metodologías del ciclo de vida de desarrollo de sistemas, que se manejan a la antigua, a estas también se le conocen como paradigmas tradicionales. Si bien, es verdad que las metodologías actuales están basadas con fundamentos de lo que fueron los paradigmas tradicionales, hoy en día ya hemos evolucionado, sin embargo los paradigmas tradicionales ahí se mantienen. Estos paradigmas, se caracterizan principalmente por ser lineales sin vuelta atrás, es decir, se trataba de completar cada proceso de principio a fin, hasta que quedara listo para avanzar a la segunda fase del ciclo del software. Esto generaba grandes dificultades y pérdidas de tiempo si se encontraba algún error en una fase avanzada, pues el proceso a realizarse era, volver atrás y volver a pasar nuevamente por las fases que ya se habían hecho y reestructurar de acuerdo a las modificaciones, pero todo con un proceso lineal, lento y tardado.

• **Paradigma Orientado a Objetos**. Una de las genialidades más exquisitas, es el desarrollo de software mediante programación orientada a objetos. Con esta forma del ciclo de vida de los sistemas, lo que se pretende es que el código fuente sea reutilizable para otros proyectos o mini proyectos alternos relacionados con el programa base, pues se utilizan Clases.

Básicamente la etapas de desarrollo de software en el paradigma orientado a objetos, se conforma principalmente lo que es la creación de clases, seguido del análisis de requisitos, un paso fundamental para determinar no solamente la duración del desarrollo, si no también los costos al final del proyecto. Y por supuesto el diseño, pues ya con el paradigma orientado a objetos, el diseño es mucho mejor que con un paradigma tradicional.

• **Paradigma de Desarrollo Ágil**. Los modelos de ciclo de vida ágiles, son de los más utilizados hoy en día. El objetivo de este paradigma, es el desarrollo de proyectos en poco tiempo. Para lo cual, se hace una eliminación de procesos tediosos, se agilizan las fases de desarrollo, las iteraciones se hacen en un corto periodo de tiempo, los riesgos se desechan y se evitan para no tener que lidiar con ellos y siempre se da solución a los problemas de forma rápida. Si algo demora mucho en dar solución, lo mejor es dejarlo de lado y seguir avanzando. Una de las principales diferencias del paradigma de desarrollo ágil con los paradigmas anteriores, es que el cliente se ve involucrado en el proyecto durante el desarrollo de este. A diferencia del paradigma tradicional donde el cliente solo está al principio, de igual forma en el paradigma orientado a objetos sucede lo mismo. Acá el cliente interfiere, da mejoras, propone ideas y se mantiene al tanto del desarrollo del producto. Lo que ayuda aún más, pues el producto final se realiza de forma satisfactoria en un menor lapso de tiempo.



<u>Ciclo de Vida del Software en las distintas Metodologías</u>

El ciclo de vida de un proyecto se software, empieza cuando se da la recolección de requerimientos para el programa a desarrollar y termina cuando el producto ha quedado completado y es entregado al cliente que lo pidió. Sin embargo en el intermedio, hay una gran cantidad de fases por las cuales se tiene que pasar y cada metodología tiene fases distintas en su ciclo de desarrollo de programas, es por eso que a continuación, veremos cómo están compuestas cada uno de los modelos de ciclo de vida del software, sin entrar a escenarios profundos, pues son tantas metodologías por mencionar que nos podríamos llevar todo el día.

Modelo en Cascada

Una de las metodologías más antiguas en lo que es el ciclo de vida de un modelo informático, es el modelo de cascada. Esta metodología es lineal y consta de algunas fases que hay que seguir y completar para poder avanzar a la fase siguiente. No es precisamente la mejor metodología, pero si se utiliza de forma correcta los resultados pueden ser muy buenos. Está compuesta por las siguientes fases:

1. Requerimientos
2. Diseño

3. Implementación y Desarrollo
4. Integración
5. Pruebas o Validación
6. Despliegue o Instalación
7. Mantenimiento

Como puedes ver, **el ciclo de vida de un programa realizado bajo la metodología en cascada, es extenso pero muy bien estructurado**. El detalle aquí es que no puedes saltarte fases ni volver a repetirlas, por ejemplo.

Si se realiza un análisis de requerimientos, avanzamos a diseñar el programa y ya estamos en el desarrollo y de momento el cliente nos dice que desea modificar los requerimientos, digamos que por tratarse del modelo en cascada, no es posible volver atrás. Por lo tanto se tendría que reiniciar el proyecto o bien concluirlo y ver cómo queda el software al final.

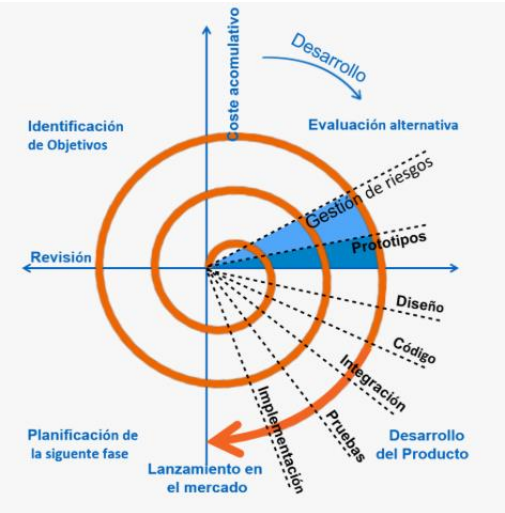
Como te mencionaba, **es una metodología lineal en cascada y si no se completa cada una de las fases al 100%, no es posible avanzar a la fase que sigue**, así es como funciona y se debe seguir al pie de la letra, por muy exagerado que esta parezca.

Modelo en el Espiral

El modelo espiral en ingeniería del software tiene un enfoque muy distinto al modelo de cascada, principalmente porque su enfoque va dirigido hacia el análisis de riesgos. El modelo de ciclo de vida en espiral, consiste en realizar diversas iteraciones, pasando por cada una de sus fases una y otra vez. A diferencia del modelo cascada que no tiene vuelta atrás, en el modelo en espiral se pueden hacer las iteraciones que se consideren necesarias y estas son sus fases principales:

1. Determinación de Objetivos
2. Análisis de riesgos
3. Desarrollo y Pruebas
4. Planificación

Entre las principales ventajas de desarrollar un proyecto con el modelo espiral, es que los riesgos se van disminuyendo conforme avanzan los ciclos o iteraciones, de hecho no puedes avanzar a un ciclo nuevo, si no se ha dado solución a todos los riesgos latentes. Lamentablemente el modelo es realmente costoso y para que puedas tener un alto nivel de eficacia en la evaluación final de tu proyecto con este ciclo de vida, necesitas que tu equipo tenga un gran nivel de conocimientos y si es posible buena experiencia para superar cualquier riesgo al cual se puedan enfrentar.



Modelo Iterativo o por Prototipos

Uno de mis modelos de ciclo de vida de antaño que realmente es de mis favoritos, es el modelo iterativo. ¿La razón?, se maneja a base de prototipos, es decir. Es uno de los primeros ciclos de vida que permitían que el código fuente fuera reutilizable, sin embargo con el modelo iterativo no solo es utilizable, si no que para muchos, estos prototipos pueden llegar a ser el producto final que siempre quisieron, lo cual lo hace realmente relevante y destacable, por encima del resto de los modelos de antaño que puedas encontrar.

Básicamente, las fases del ciclo de vida del sistema, son las siguientes:

1. Inicialización
2. Iteración
3. Lista de Control

Una de las principales ventajas del modelo iterativo, es que la retroalimentación a los usuarios se proporciona desde muy temprano, haciendo que adentrarse en el proyecto sea demasiado sencillo. Por supuesto que el hecho de contar con iteraciones nos da ciertas ventajas, pues con cada iteración realizada, se van separando las partes complejas de el, permitiendo más el acceso al software. Y por supuesto, un sistema creado mediante el ciclo de vida iterativo, tiende a no fallar casi, lo cual es garantía de satisfacción para el cliente en este caso o para la empresa que está implementando esta metodología.

Modelos del Ciclo de Vida del Desarrollo Ágiles

Las tendencias, con el paso del tiempo suelen cambiar para bien y en el caso de las metodologías del ciclo de vida desarrollo de software no es la excepción. Y un claro ejemplo de esto, son los modelos de desarrollo ágil. **Estos procesos se caracterizan por estar basados en las etapas del ciclo de vida del software tradicionales, pero combinándolas con algunas técnicas y siendo aún más solapadoras** en cuando al orden que se deben ejecutar. Bueno no les diré más, mejor vamos a ver brevemente cuales son algunas de ellas, las más conocidas y populares, claro y la mejor de todas.

Modelo Scrum

El ciclo de vida del sistema, puede agilirse si se utiliza la metodología Scrum, uno de los modelos del ciclo de vida del desarrollo del software más populares y mas recientes, bueno no tanto, pero si más que los de antaño. El modelo Scrum, se encuentra basado en lo que es el desarrollo incremental, es decir, conforme pasen las fases y las iteraciones, mayor va a ser el tamaño del proyecto que se esté desarrollando, es por eso que uno de los principales requisitos para llevarlo a cabo, es que tu equipo de desarrollo sea de calidad. Teniendo una alta calidad en el equipo, tendremos garantizado un excelente funcionamiento.

Como te mencionaba al principio, **el modelo Scrum, deja de seguir metodologías lineales, podemos despedirnos del modelo cascada y secuencial, pues ahora procedemos a solapar las fases y no importará en que momento tengas que volver atrás**, siempre habrá un equipo de trabajo de buena calidad, que tenga ese soporte para aguantar los cambios que son ciertamente normales dentro de la metodología Scrum. Por último, como ingrediente vital tenemos la comunicación, y es que acá olvídate de las tendencias de ese jefes que te tienen envuelto en una burbuja desarrollando. Con el modelo scrum podrás estar comunicado con tu equipo de trabajo en todo momento, para estar al tanto de los sucesos.

Ahora veremos brevemente, cuales son los procesos que el modelo Scrum utiliza:

1. Product Backlog
2. Sprint Backlog
3. Sprint Planning Meeting
4. Daily Scrum o Stand-up Meeting

5. Sprint Review
6. Sprint Retrospective

Estas son las fases del ciclo de vida del software en esta metodología, el cuál básicamente **consiste en realizar un análisis de los requerimientos del sistema (Product Backlog**), señalar cuales serán los objetivos a corto o mediano plazo dentro de un **sprint**, osea, la fase de desarrollo. Posteriormente los desarrolladores harán lo suyo, se realizan algunas pruebas y se retroalimenta de acuerdo a lo conseguido al terminar la última fase. Recuerda que aquí, se pueden añadir nuevas cosas en todo momento, pues el modelo Scrum no se bloquea en ninguna de sus fases.

Modelo Kanban

El modelo Kanban, es uno de los modelos más visuales de las metodologías ágiles. **Consiste en la creación de un tablero con etiquetas, donde se seccionan cada una de las fases de su desarrollo**, además se clasifica de acuerdo a los equipos de trabajo y se les asignan objetivos a corto, mediano y largo plazo.

Entre las ventajas de este modelo del ciclo de vida del software, **destaca el hecho de no tener un orden tal cual, de hecho todas las fases comienzan a trabajar a la par, no hay tiempos de espera y básicamente su objetivo es que los desarrolladores y programadores estén trabajando todo el tiempo**. Si concluyes con las fases del proyecto que te corresponde, seguramente tendrás que avanzar en fases del nuevo proyecto que está por venir.

Por supuesto, la metodología Kanban, también requiere de un equipo totalmente capacitado, pues solamente de esta forma se podrán lograr los objetivos. Así que aquí les muestro las fases del proceso del ciclo de vida de un sistema, mediante la metodología japonesa Kanban:

1. Definir el Flujo de Trabajo
2. Fases del Ciclo de Producción
3. Stop Starting, start finishing
4. Tener un Control

Si aún no estás seguro de si implementar la metodología Kanban o no, te cuento. La empresa Toyota, ha sido una de las primeras en implementar la metodología, incrementando la eficiencia y productividad en un alto porcentaje. Considera como principal ventaja, que el producto final quedará terminado en un periodo de tiempo mucho más corto, que con cualquiera de las metodologías vistas al principio.

Modelo XP o Programación extrema

Posiblemente la más destacada de las metodologías ágiles para los ciclos de vida de un software, es la metodología XP o modelo de programación extrema. A diferencia del resto de las metodología del mundo, habidas y por haber, esta es adaptable de acuerdo a las necesidades y requerimientos que se tengan que implementar, con la ventaja de que podemos hacer uso de cualquier modelo anterior para el desarrollo y de inmediato salirnos y programar otras cosas, es muy solapador y permite mucha más libertad en el equipo de trabajo que el resto de los modelos.

Además, si querías una diferencia aún mayor, en la metodología de programación extrema, el cliente se encuentra involucrado en el proceso de desarrollo, lo que hace que al final el producto pueda estar terminado en un menor tiempo, pues evitamos muchas pérdidas de tiempo, elaborando cosas que no son y que en la revisión al cliente no le agradarán, acá el cliente va viendo lo que se va desarrollando y tiene la libertad de proponer cambios, ideas, requerimientos o actualizaciones sin ningún problema.

Los valores que componen a al modelo de programación extrema, son los siguientes:

1. Comunicación
2. Simplicidad
3. Retroalimentación
4. Valentía
5. Respeto

Esta serie de valores, son de suma importancia para que se pueda llevar a cabo un proyecto de alta calidad. Cada uno de ellos, tiene su razón de ser y existir, por ejemplo, la comunicación, la cual debe estar incluso con el cliente y ni hablar del resto de los equipos de trabajo. La simplicidad corresponde al hecho de no hacer cosas que quiten mucho tiempo, la idea es terminar rápido y las cosas que sean muy tardadas es mejor dejarlas de lado. **La retroalimentación es vital, más cuando los equipos de trabajo deben ser de dos personas, siempre es bueno aprender cosas nuevas de nuestro compañero de trabajo** y esto seguramente todos lo hemos vivido alguna vez.

La valentía es un valor integrado como programador, pues deber ser valiente para afrontar los cambios que se vengan, tomar decisiones radicales y en todo momento mantener esa fuerza que tanto a ti como a tu equipo de trabajo los debe mantener a tope. Y por supuesto el respeto, esto es en todo el equipo de trabajo, hasta el cliente debe tener un margen de respeto por el equipo de desarrollo. Con estos valores la metodología tendrá una buena formación, pero vamos a ver cuáles son las características principales de la programación extrema:

1. Tipo de Desarrollo Iterativo e incremental
2. Pruebas Unitarias
3. Trabajo en Equipo
4. Trabajo junto al cliente
5. Corrección de Errores
6. Reetructuración del Código
7. El Código es de todos
8. Código simple es la clave

Como te puedes dar cuenta, pasos como hacer el código simple o las pruebas unitarias para prevenir errores y tener que darle muchas vueltas al código, además de que las fases se segmentan en pequeñas porciones, para que si hay errores, estos puedan ser modificados fácilmente.
Sin embargo seguramente notaste que no hay documentación por ningún lado, esto es porque con tanta comunicación, en realidad no es necesaria, sin embargo dentro del código se van dejando comentarios con las cosas que otro programador no pueda llegar a entender y se utilizan variables o clases entendibles para que todo el mundo tenga la facilidad de comprenderlo. Ya solamente en caso muy necesario, se procede a hacer documentación breve para algunas partes, pero regularmente no se utiliza de forma tradicional.