Facultad de Tecnologías de la Información y Comunicación.

Escuela de Ingeniería de Sistemas Informáticos.

Metodologías de Desarrollo de Software y Que es MyISAM e INNO DB y una Tabla Comparativa.

Programación III (BSI-090).

Profesor: Lic. Jorge Isaac Vásquez Valenciano.

Estudiante: Anthony Quirós Alfaro.

(20160111305).

Universidad Latina San Pedro.

21 mayo de 2018.

Índice

1. Introducción:

En el presente trabajo vamos a investigar que significa que es un método de desarrollo de software y cuales tipos existen, y así para en los próximos proyectos y trabajos escoger el método más adecuado para nuestras necesidades. También vamos a investigar sobre los significados de los siguientes de MyISAM e INNO DB y que es una tabla comparativa y sus aplicaciones en la programación.

2. ¿Qué son metodologías de desarrollo?

Es el proceso la cual la finalidad es desarrollar productos o soluciones para un cliente, teniendo en cuenta factores como costos, a planificación, la calidad y las dificultades asociadas, es decir que es el proceso que se tiene que seguir a la hora de diseñar una solución o un programa en específico. Tiene que ver, por tanto, con la comunicación, la manipulación de modelos y el intercambio de información y datos entre las partes involucradas. O para ser más precisos, las metodologías de desarrollo de software son enfoques de carácter estructurado y estratégico que permiten el desarrollo de programas con base a modelos de sistemas, reglas, sugerencias de diseño y guías.

3. Tipos de metodología de software:

Las metodologías de desarrollo de software en si no varían en lo tendencial, pero si se puede hablar de modelos de trabajos diferentes. Estos métodos de trabajo se han creado para satisfacer necesidades específicas según los proyectos, ahora vamos a explicar los 5 tipos diferentes de modelos:

3.1 Modelo en cascada:

También llamado Lineal secuencial, es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

3.1.1 Fases del modelo en cascada:

3.1.1.1 Análisis de requisitos:

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada

SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos. Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

3.1.1.2 Diseño del sistema:

Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras. Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

3.1.1.3 Diseño del programa:

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario, así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

3.1.1.4 Codificación:

Es la fase en donde se implementa el código fuente, haciendo uso de prototipos, así como de pruebas y ensayos para corregir errores. Dependiendo del lenguaje de programación y su versión se crean las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

3.1.1.5 Pruebas:

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente y que cumple con los requisitos, antes de ser entregado al usuario final.

Existen varios tipos de Pruebas:

- Pruebas de unidad.
- Pruebas de integración.
- Pruebas de sistema.
- Pruebas de aceptación.

3.1.1.6 Verificación:

Es la fase en donde el usuario final ejecuta el sistema, para ello el o los programadores ya realizaron exhaustivas pruebas para comprobar que el sistema no falle.

3.1.1.7 Mantenimiento:

Una de las etapas más críticas, ya que se destina un 75% de los recursos, es el mantenimiento del Software ya que al utilizarlo como usuario final puede ser que no cumpla con todas nuestras expectativas.

Tipos de Mantenimiento:

- Mantenimiento Preventivo y Perfectivo.
- Mantenimiento Correctivo.
- Mantenimiento Evolutivo.

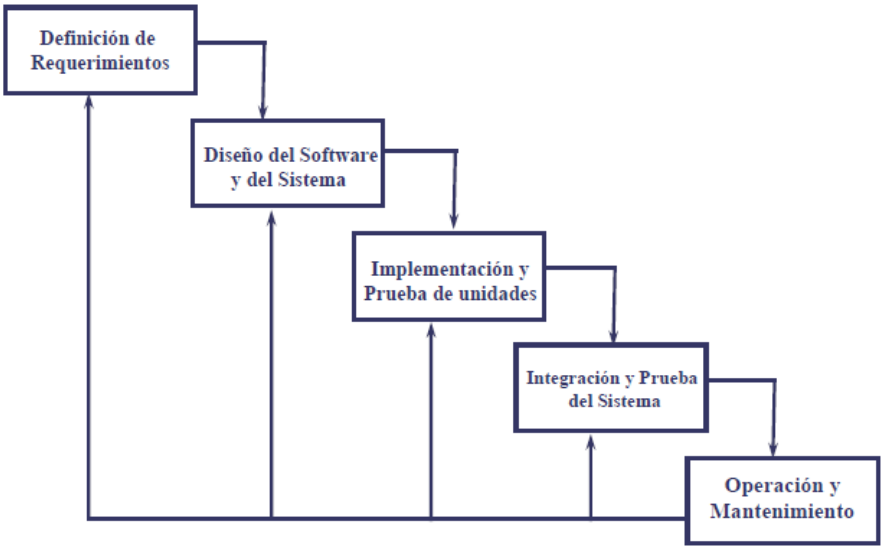
3.1.2 Ventajas del Modelo Cascada:

- Modelo y planificación fácil y sencillos.
- Sus fases son conocidas por los desarrolladores.
- Los usuarios lo pueden comprender fácilmente.

3.1.3 Desventajas del Modelo Cascada:

- En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.
- El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.

- Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.



3.2 Modelo Espiral:

Propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software.

3.2.1 Funcionamiento del modelo:

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

3.2.2 Regiones del modelo:

El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas regiones de tareas. Generalmente, existen entre tres y seis regiones de tareas.

3.2.2.1 Comunicación con el cliente:

Las tareas requerías para establecer comunicación entre el desarrollador y el cliente.

3.2.2.2 Planificación:

Las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.

3.2.2.3 Análisis de riesgo:

Las tareas requeridas para evaluar riesgos técnicos y de gestión.

3.2.2.4 Ingeniería:

Las tareas requeridas para construir una o más representaciones de la aplicación.

3.2.2.5 Construcción y acción:

Las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica).

3.2.2.6 Evaluación del cliente:

Las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación. Cada una de las regiones están compuestas por un conjunto de tareas del trabajo, llamado conjunto de tareas, que se adaptan a las características del proyecto que va a emprenderse. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo. Para proyectos mayores y más críticos cada región de tareas contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad. En todos los casos, se aplican las actividades de protección.

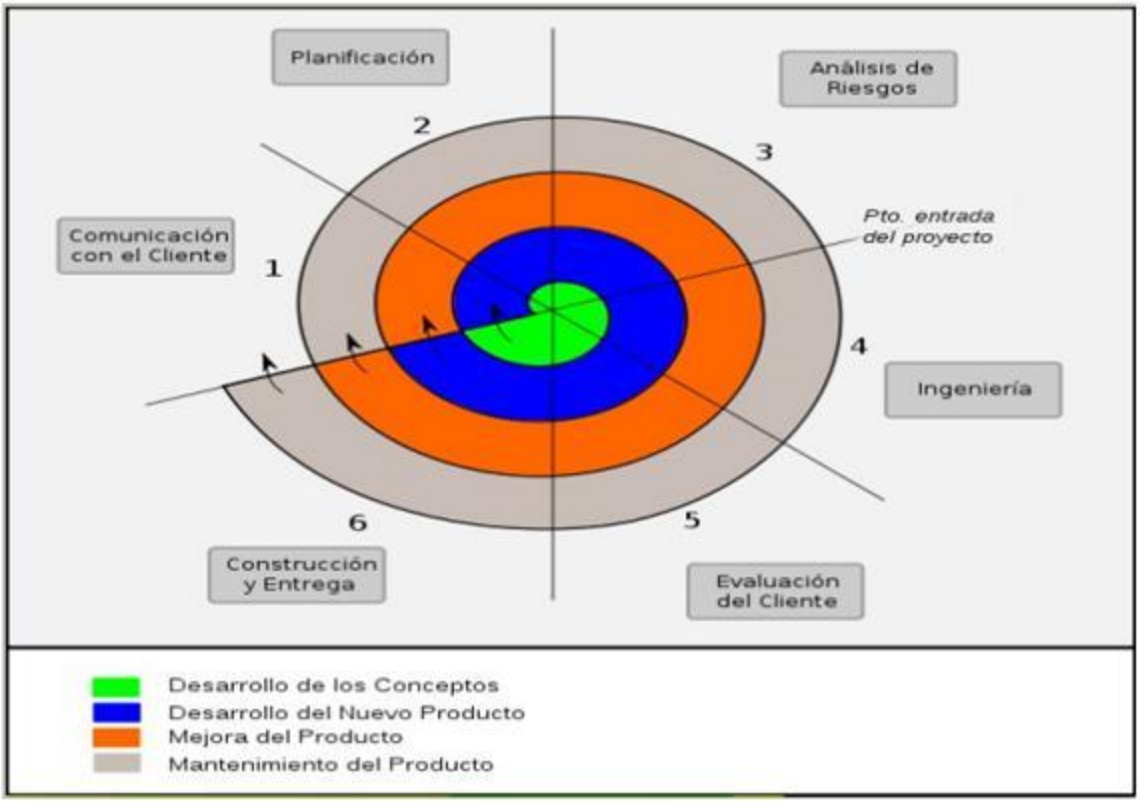
3.2.3 Ventajas del Modelo:

- Puede adaptarse y aplicarse a lo largo de la vida del software de computadora.
- Es un enfoque realista del desarrollo de sistemas y de software a gran escala.
- Como el software evoluciona, a medida que progresa el proceso el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.
- Utiliza la construcción de prototipos como mecanismo de reducción de riesgos.
- Permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real.

- Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto, y si se aplica adecuadamente, debe reducir los riesgos antes de que se conviertan en problemáticos.

3.2.4 Desventajas del Modelo:

- Puede resultar difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable.
- Requiere una considerable habilidad para la evaluación del riesgo.
- No se ha utilizado tanto como los paradigmas lineales secuenciales o de construcción de prototipos.



3.3 Modelo Prototipos:

Se inicia con la definición de los objetivos globales para el software, luego se identifican los requisitos conocidos y las áreas del esquema en donde es necesaria más definición. Estos modelos se utilizan para dar al usuario una vista preliminar de parte del software. Este modelo es básicamente prueba y error ya que si al usuario no le gusta una parte del prototipo significa que la prueba fallo por lo cual se debe corregir el error que se tenga hasta que el usuario quede satisfecho. Además, el prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar mucho dinero pues a partir de que este sea aprobado nosotros podemos iniciar el verdadero desarrollo del software. Pero eso si al construir el prototipo nos asegura que nuestro software sea de mejor calidad, además de que su interfaz sea de agrado para el usuario. Un prototipo podrá ser construido solo si con el software es posible experimentar.

3.3.1 Etapas:

- Recolección y refinamiento de requisitos.
- Modelado, diseño rápido.
- Construcción del Prototipo.
- Desarrollo, evaluación del prototipo por el cliente.
- Refinamiento del prototipo.
- Producto de Ingeniería.

3.3.2 Como se lleva acabo:

Se comienza elaborando un prototipo del producto final: qué aspecto tendrá, cómo funcionará. Para muchas interfaces de usuario, este modelo puede resultar tan simple como unos dibujos con lápiz y papel o tan complejo como el propio código operativo final. Para interfaces de hardware o estaciones de trabajo, el modelo puede consistir en maquetas de espuma, caucho, cartón o cartulina. Cuanto más próximo se encuentre el prototipo al

producto real, mejor será la evaluación, si bien se pueden obtener magníficos resultados con prototipos de baja fidelidad.

3.3.3 Ventajas:

- No modifica el flujo del ciclo de vida.
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.
- Reduce costo y aumenta la probabilidad de éxito.
- Exige disponer de las herramientas adecuadas.
- Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
- También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema. operativo o de la forma que debería tomar la interacción humano-máquina.

3.3.4 Desventajas:

- Debido a que el usuario ve que el prototipo funciona piensa que este es el producto terminado y no entienden que recién se va a desarrollar el software.
- El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y mantenimiento que tiene con el cliente.

3.3.4 Para que sea efectivo:

- Debe ser un sistema con el que se pueda experimentar.
- Debe ser comparativamente barato (menor que el 10%).
- Debe desarrollarse rápidamente.

- Énfasis en la interfaz de usuario.
- Equipo de desarrollo reducido.
- Herramientas y lenguajes adecuadas.

3.3.5 Tipo de modelos de prototipo:

3.3.5.1 Modelo de Prototipos rápido: Metodología de diseño que desarrolla rápidamente nuevos diseños, los evalúa y prescinde del prototipo cuando el próximo diseño es desarrollado mediante un nuevo prototipo.

3.3.5.2 Modelo de Prototipos reutilizable: También conocido como "Evolutionary Prototyping"; no se pierde el esfuerzo efectuado en la construcción del prototipo pues sus partes o el conjunto pueden ser utilizados para construir el producto real. Mayormente es utilizado en el desarrollo de software, si bien determinados productos de hardware pueden hacer uso del prototipo como la base del diseño de moldes en la fabricación con plásticos o en el diseño de carrocerías de automóviles.

3.3.5.3 Modelo de Prototipos Modular: También conocido como Prototipado Incremental (Incremental prototyping); se añaden nuevos elementos sobre el prototipo a medida que el ciclo de diseño progresa.

3.3.5.4 Modelo de Prototipos Horizontal: El prototipo cubre un amplio número de aspectos y funciones, pero la mayoría no son operativas. Resulta muy útil para evaluar el alcance del producto, pero no su uso real.

3.3.5.5 Modelo de Prototipos Vertical: El prototipo cubre sólo un pequeño número de funciones operativas. Resulta muy útil para evaluar el uso real sobre una pequeña parte del producto.

3.3.5.6 Modelo de Prototipos de Baja-fidelidad: El prototipo se implementa con papel y lápiz, emulando la función del producto real sin mostrar el aspecto real del mismo. Resulta muy útil para realizar test baratos.

3.3.5.7 Modelo de Prototipos de Alta-fidelidad: El prototipo se implementa de la forma más cercana posible al diseño real en términos de aspecto, impresiones, interacción y tiempo.

3.3.6 Tipos de prototipos: Hay dos clases de prototipos el desechable y el evolucionario.

3.3.6.1 El desechable: nos sirve para eliminar dudas sobre lo que realmente quiere el cliente además para desarrollar la interfaz que más le convenga al cliente.

3.3.6.2 El evolucionario: es un modelo parcialmente construido que puede pasar de ser prototipo a ser software, pero no tiene una buena documentación y calidad.

3.4 Desarrollo Rápido de Aplicaciones (RAD):

Modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. Es una adaptación a "Alta velocidad" en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de periodos cortos de tiempo.

3.4.1 Fases de modelo RAD: Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las siguientes fases:

3.4.1.1 Modelado de gestión: El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la proceso?

3.4.1.2 Modelado de datos: El flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.

3.4.1.3 Modelado de proceso: Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.

3.4.1.4 Generación de aplicaciones: El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación

de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

3.4.1.4 Pruebas y entrega: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfases a fondo.

3.4.2 Ventajas:

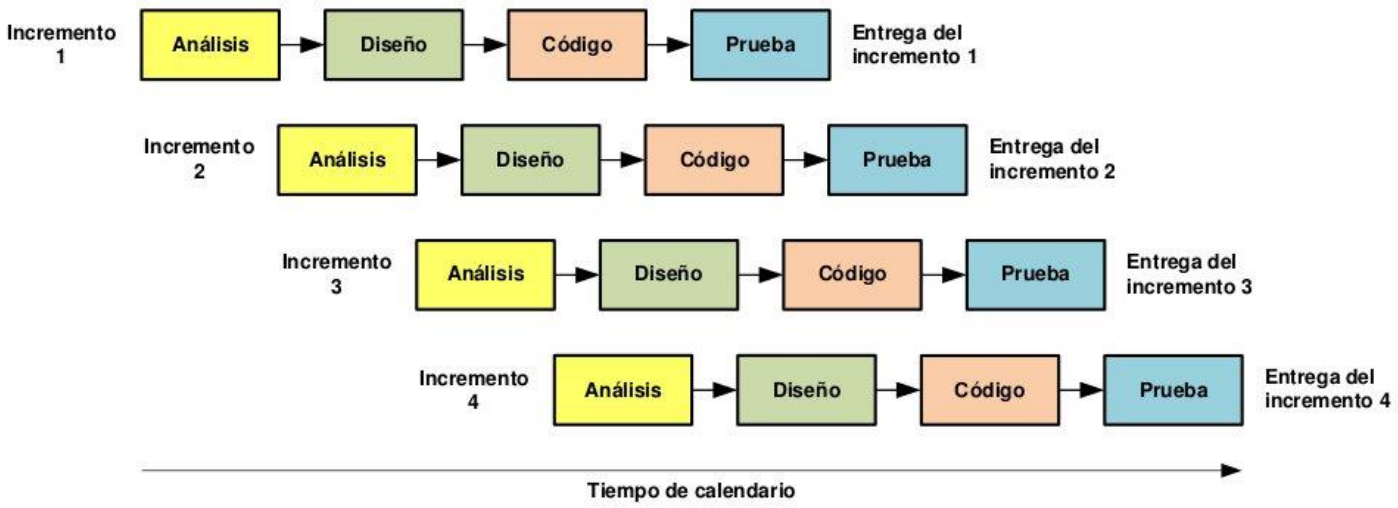
- Es muy rápido.
- Permite trabajar en él a varias personas a la vez.

3.4.3 Desventajas:

- Para proyectos grandes, aunque por escalas, el DRA requiere recursos humanos suficientes como para crear el número correcto de equipos DRA.
- Requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso, por ninguna de las partes constituyentes, los proyectos DRA fracasaran.
- No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modulizar adecuadamente. La construcción de los componentes necesarios para DRA será problemático. Si está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistema, el enfoque DRA puede que no funcione.
- No es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando

el nuevo software requiere un alto grado de interoperabilidad con programas de computadora ya existentes.

- Enfatiza el desarrollo de componentes de programas reutilizables. La reutilización es la piedra angular de las tecnologías de objetos, y se encuentra en el modelo de proceso de ensamblaje.



3.5 Metodología de Programación Extrema (XP):

Conocida comúnmente por sus siglas en inglés XP. Es una metodología de desarrollo del software, enmarcada dentro de las metodologías ágiles de desarrollo. Constituye la metodología más utilizada dentro del grupo de las ágiles. Su objetivo principal es asegurar la producción de software con buena calidad y cubriendo las necesidades y requerimientos del usuario.

3.5.1 Características generales:

Surge como posible solución a los problemas derivados del cambio en los requerimientos; esta metodología ofrece la posibilidad de cambiar los requisitos

en cualquier momento de la vida de un proyecto, ya que es adaptable a estos cambios. Se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Este método es típicamente atribuido a Kent Beck, Ron Jeffries y Ward Cinningham. El objetivo de Xp son grupos pequeños y medianos de construcción de software en donde los requisitos aún son muy ambiguos, cambian rápidamente o son de alto riesgo. Xp busca la satisfacción del cliente tratando de mantener durante todo el tiempo su confianza en el producto. Además, sugiere que el lugar de trabajo sea una sala amplia, si es posible sin divisiones (en el centro los programadores, en la periferia los equipos individuales). Una ventaja del espacio abierto es el incremento en la comunicación y el proporcionar una agenda dinámica en el entorno de cada proyecto. XP es un proceso muy orientado a la implementación, en el que se genera poca documentación y en que la funcionalidad exacta del sistema final no se define nunca formal y contractualmente. Es por eso que este método es más aplicable para desarrollos internos.

3.5.1.1 Ciclo de vida del XP:

- Exploración.
- Planificación de la Entrega (Release)
- Iteraciones.
- Producción.
- Mantenimiento.
- Muerte del Proyecto.

3.5.1.2 Proceso de XP:

- El cliente define el valor de negocio a implementar.
- El programador estima el esfuerzo necesario para su implementación.

- El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
- El programador construye ese valor de negocio.
- Vuelve al paso 1.

3.5.2 Las Historias de Usuario en XP:

Las historias de usuario son utilizadas por la metodología XP como una técnica para especificar los requisitos del mismo, tanto requisitos no funcionales como funcionales. Se trata de tarjetas en las cuales el usuario describe las características que el sistema debe poseer, de forma muy breve. Las características fundamentales del método son:

3.5.2.1 Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.

3.5.2.2 Pruebas unitarias continuas: frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión: Se aconseja escribir el código de la prueba antes de la codificación.

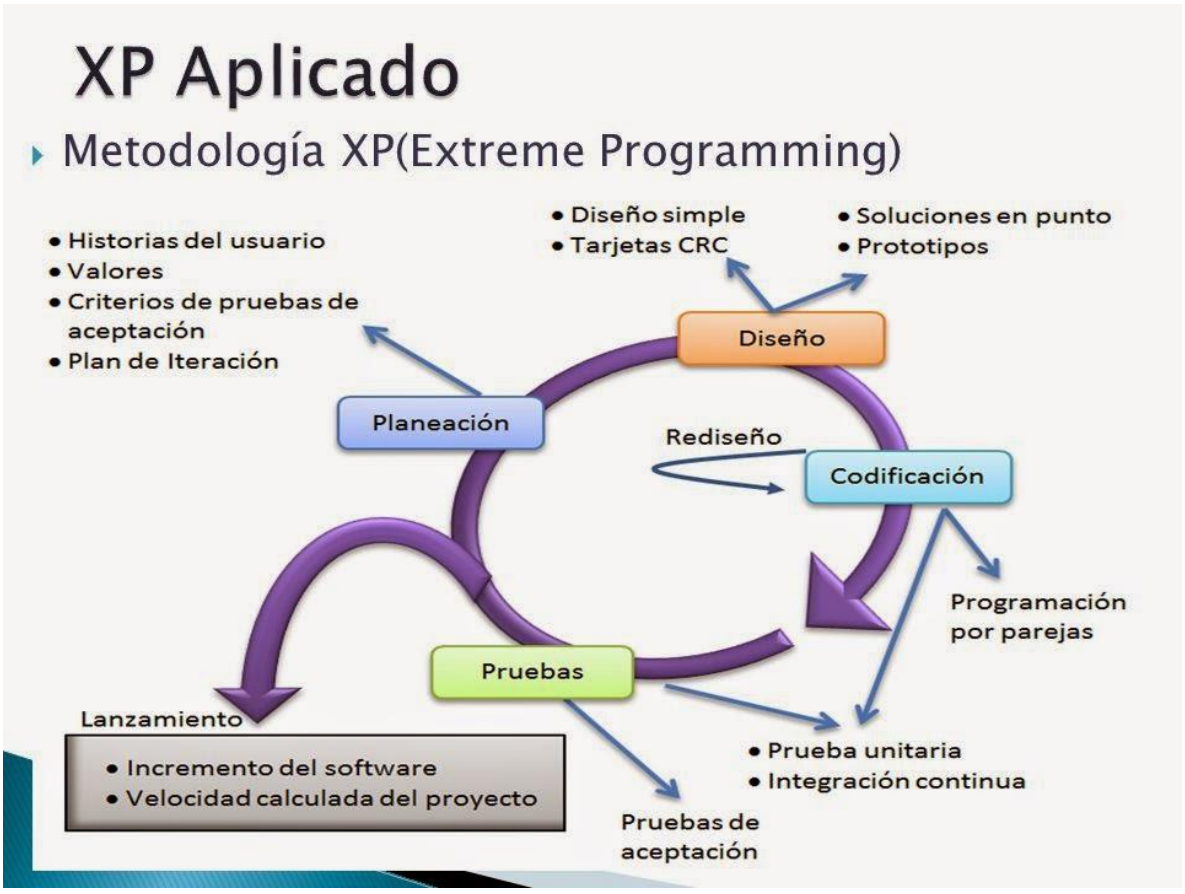
3.5.2.3 Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.

3.5.2.4 Frecuente integración del equipo de programación 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.

3.5.2.5 Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario.

3.5.3 Roles XP:

- Programador.
- Cliente.
- Encargado de pruebas (Tester).
- Encargado de seguimiento (Tracker).
- Entrenador (Coach).
- Consultor.
- Gestor (Big boss).



4. ¿Qué es MyISAM y InnoDB?

El motor de almacenamiento (storage-engine) se encarga de almacenar, manejar y recuperar información de una tabla. Los motores más conocidos son MyISAM e InnoDB. La elección de uno u otro dependerá mucho del escenario donde se aplique, pero Arsys quiere ayudarnos a conocer mejor estos conocidos motores de almacenamiento. En la elección se pretende conseguir la mejor relación de calidad acorde con nuestra aplicación. Si necesitamos transacciones, claves foráneas y bloqueos, tendremos que escoger InnoDB. Por el contrario, escogeremos MyISAM en aquellos casos en los que predominen las consultas SELECT a la base de datos.

4.1 InnoDB: dota a MySQL de un motor de almacenamiento transaccional (conforme a ACID) con capacidades de commit (confirmación), rollback (cancelación) y recuperación de fallos. InnoDB realiza bloqueos a nivel de fila y también proporciona funciones de lectura consistente sin bloqueo al estilo Oracle en sentencias SELECT. Estas características incrementan el rendimiento y la capacidad de gestionar múltiples usuarios simultáneos. No se necesita un bloqueo escalado en InnoDB porque los bloqueos a nivel de fila ocupan muy poco espacio. InnoDB también soporta restricciones FOREIGN KEY. En consultas SQL, aún dentro de la misma consulta, pueden incluirse libremente tablas del tipo InnoDB con tablas de otros tipos.

4.1.1 Ventajas:

- Soporte de transacciones
- Bloqueo de registros
- Nos permite tener las características ACID (Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español), garantizando la integridad de nuestras tablas.
- Es probable que si nuestra aplicación hace un uso elevado de INSERT y UPDATE notemos un aumento de rendimiento con respecto a MyISAM.

4.1.2 Desventajas:

- El diseño de su estructura es mucho más complejo que el diseño de una tabla MyISAM.
- No permite las búsquedas denominadas full-text, que para conjuntos de datos grandes son mucho más eficientes.
- Bajo rendimiento con operaciones sencilla.

4.2 MyISAM

El tipo MyISAM es el motor de almacenamiento por defecto que se le asigna a las tablas que se crean, si no se le indica que utilice otro tipo de motor. Se basa en el código ISAM lo que hace que sean tablas muy fiables, pero añadiendo nuevas características. Cada tabla MyISAM se almacena en disco en tres ficheros. Los ficheros tienen nombres que comienzan con el nombre de tabla y tienen una extensión para indicar el tipo de fichero. Un fichero .FRM almacena la definición de tabla. El fichero de datos tiene una extensión .MYD mientras que el fichero índice tiene una extensión .MYI.

4.2.1 Ventajas:

- Mayor velocidad en general a la hora de recuperar datos.
- Recomendable para aplicaciones en las que dominan las sentencias SELECT ante los INSERT /UPDATE.
- Ausencia de características de atomicidad ya que no tiene que hacer comprobaciones de la integridad referencial, ni bloquear las tablas para realizar las operaciones, esto nos lleva como los anteriores puntos a una mayor velocidad.

4.2.2 Desventajas:

- Tablas más simples, lo que hace que sean la mejor opción para aquellos que están empezando.
- Bloqueo de tablas. - Nos permite tener características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), garantizando de esta forma la integridad de nuestra tabla.
- Aumento del rendimiento si nuestra aplicación realiza un elevado número de consultas “Select”.

4.3 Tabla Comparativa MyISAM y InnoDB

MyISAM	InnoDB
Bloqueo a nivel de tabla. Si hay actualizaciones (UPDATE) de registros, bloquea la tabla entera, lo cual es más ineficiente.	Bloqueo a nivel de registro. Lo cual es más eficiente que a nivel de tabla, ya que se permite que otras transacciones trabajen con el resto de registros.
Si se realiza una gran cantidad de consultas (SELECT), el rendimiento es más elevado. Por lo tanto, se puede decir que tiene una mayor velocidad al recuperar datos.	Soporte de transacciones.
En caso de errores, las tablas pueden dar problemas en la recuperación de datos. Una de las claves para ganar rendimiento viene dado por esta peor gestión de la recuperación ante errores.	Mayor rendimiento para operaciones de inserción y actualización (INSERT/UPDATE).
Permite hacer búsquedas full-text, las cuales son más precisas que las búsquedas con like.	Mejor respuesta a la concurrencia.
Menos consumo de recursos (memoria RAM).	Más confiabilidad, al tener características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad. Esto también repercute en una mejor integridad de los datos.
Ausencia de características de atomicidad. No tiene comprobaciones de integridad.	partir de mySQL 5.6, también permite búsquedas full-text.
	Mejor recuperación de errores a partir de sus logs.

5. Conclusión:

Al finalizar la investigación nos descubrimos diferentes modos para realizar un proyecto, descubrimos sus ventajas y desventajas y de acuerdo a ella podemos establecer cuál es la más adecuada para nuestro proyecto dependiendo a las necesidades del cliente, así como a la cantidad de colaboradores del mismo. También descubrimos unos significados básicos de gestión de bases de datos y sus diferencias, así nos ayudara durante este curso y también para nuestro futuro profesional.

Bibliografía:

Gómez, K. (2017 Julio). Top 5 Metodologías de Desarrollo de Software. Recuperado de https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software

Grijalva, N (2017 febrero). Modelo Espiral. Recuperado de http://sofware1nathalygrijalva.blogspot.com/2012/10/modelo-espiral.html

García, O. (2013 septiembre). Modelo de prototipos. Recuperado de http://www.proyectum.lat/2013/09/02/modelo-de-prototipos/

Gutiérrez, D. (2011 Julio). Modelo en Cascada Concepto. Recuperado de https://sites.google.com/site/proyectoadpmodelosdedesarrollo/home/modelo-en-cascada

Roger, S. METODOLOGÍA DE ROGER PRESSMAN. Recuperado de https://sisteminformacii.wikispaces.com/METODOLOG%C3%8DA+DE+ROGER+PRESSMAN

Pressman, R. Editorial: MacGraw Hill - 5ta Edición Título: Ingeniería del Software

Calero, W. (2010 octubre). Modelo Incremental. Recuperado de http://ingenieraupoliana.blogspot.mx/2010/10/modelo-incremental.html

Hiebaum, J. (2015 mayo). El origen de la programación extrema. Recuperado de http://manualdelgamedesigner.blogspot.com/2015/05/programacion-extrema-xp.html

Arsys. (2012 abril). ¿MyISAM o InnoDB? Elige tu motor de almacenamiento MySQL. Recuperado de https://www.arsys.es/blog/programacion/myisam-o-innodb-elige-tu-motor-de-almacenamiento-mysql/

Gardok, C. Conoce los tipos de Bases de Datos MYSQL y sus diferencias. Recuperado de https://www.hostalia.com/news/mayo12/Conoce-los-tipos-de-Bases-de-Datos-MYSQL-y-sus-diferencias.pdf