Metodologías de desarrollo de Proyectos

El desarrollo de una aplicación Web para Internet o Intranet es a menudo un proceso complejo que implica muchos desarrolladores de diferentes equipos desempeñando varios roles. Para organizar el proceso de desarrollo y garantizar que todas las personas en el proyecto trabajen en equipo, podemos utilizar alguna Metodología de Desarrollo de la amplia gama de metodologías existentes. Esas metodologías de desarrollo describen las fases del proyecto de desarrollo, los roles que las personas toman, los entregables que concluyen cada fase y otros aspectos del proyecto. Debemos elegir una metodología de desarrollo en una etapa temprana en el proyecto. Muchas organizaciones tienen una metodología estándar que utilizan siempre para el desarrollo del proyecto.

Algunas metodologías de desarrollo de proyectos incluyen el Modelo de Cascada, el Modelo de Desarrollo Iterativo, el Modelo de Prototipo, el Modelo de Desarrollo Ágil de Software, Programación Extrema y el Desarrollo Orientado a Pruebas.

Las metodologías de desarrollo en Cascada, Iterativa y de Prototipos son modelos viejos y en la actualidad menos populares. En la actualidad deberíamos inclinarnos por las metodologías Ágil, Programación Extrema y Orientada a Pruebas que son los modelos más actualizados.

Modelo en Cascada (Waterfall Model)

El modelo de cascada es una metodología que define las siguientes fases del proyecto:

- **Análisis de Viabilidad**. En esta fase, los planificadores y desarrolladores estudian y determinan los enfoques y las tecnologías que pueden ser utilizadas para construir la aplicación.
- **Análisis de requerimientos**. En esta fase, los planificadores y analistas, entrevistan a los usuarios, directores, administradores y a otros involucrados en el desarrollo del software para determinar sus necesidades.
- **Diseño de la aplicación.** En esta fase, los planificadores, analistas y desarrolladores registran una solución propuesta.
- **Codificación y pruebas unitarias**. En esta fase, los desarrolladores crean el código y prueban de forma individual los componentes que conforman el sistema.
- **Integración y pruebas del sistema**. En esta fase, los desarrolladores integran los componentes que han desarrollado y prueban el sistema completo.
- **Publicación y mantenimiento**. En esta fase, los desarrolladores y los administradores entregan la solución de tal forma que los usuarios pueden empezar a utilizar la aplicación.

El modelo de Cascada clasifica el proyecto de desarrollo en distintas fases, con una definición clara de los entregables en cada fase. El modelo también hace hincapié en la importancia de las pruebas. Sin embargo, el cliente no recibe ningún software funcional para su revisión hasta el final del proyecto.

Esto dificulta trabajar con cambios al diseño como respuesta a una retroalimentación o al manejo de circunstancias alteradas.

Modelo de Desarrollo Iterativo (Iterative Development Model)

Cuando se utiliza un modelo de desarrollo iterativo, el proyecto se divide en partes pequeñas. Para cada parte, realizamos las actividades relacionadas con todas las etapas del modelo en cascada. El proyecto se construye etapa por etapa, con pruebas exhaustivas en cada etapa para asegurar la calidad.

En un proyecto iterativo, podemos realizar las acciones correctivas al final de cada iteración. Esas correcciones podrían reflejar un mejor entendimiento de los problemas de negocios, retroalimentación importante de los usuarios o un mejor entendimiento de las tecnologías utilizadas para construir la solución. Dado que los requerimientos son agregados al final de cada iteración, los proyectos iterativos requieren una gran cantidad de esfuerzo de administración del proyecto y con frecuencia presentan una saturación de esfuerzos planificados.

Modelo de Prototipos (Prototyping Model)

El modelo de Prototipos es adecuado para un proyecto en el que se empieza con pocos o escasos requerimientos de negocio definidos. Esta situación ocurre cuando los clientes o los interesados en el desarrollo de la solución tienen una idea vaga de sus necesidades y la forma de resolverlas.

En este enfoque, los desarrolladores crean una versión simplificada de la aplicación (Prototipo) y luego buscan la retroalimentación de los usuarios del sistema. Esta retroalimentación sobre el prototipo es utilizada para definir los requerimientos detallados que los desarrolladores utilizan en la siguiente iteración para construir una solución que cumpla con las necesidades de las partes interesadas para ayudarlos a realizar su trabajo.

Después de dos o más iteraciones, cuando tanto los interesados en el desarrollo como los desarrolladores llegan a un consenso sobre los requerimientos, una solución completa es construida y probada. El modelo de prototipos, sin embargo, puede llevar a una aplicación mal diseñada ya que en ninguna etapa de proyecto hay un claro enfoque sobre la arquitectura general.

Modelo de Desarrollo Ágil de Software (Agile Software Development Model)

El modelo de Cascada, el modelo de Desarrollo Iterativo y el modelo de Prototipos están basados sobre la premisa de que los requerimientos de negocio y otros factores no cambian desde el principio hasta el final del proyecto. En realidad, esta suposición es a menudo no válida. El Desarrollo Ágil de

Software es una metodología diseñada para integrar las circunstancias y requerimientos cambiantes a lo largo de todo el proceso de desarrollo. Los proyectos ágiles se caracterizan por:

- **Un desarrollo incremental**. El software es desarrollado en ciclos rápidos que se construyen sobre ciclos anteriores. Cada iteración se prueba exhaustivamente.
- **Énfasis en las personas e interacciones**. Los desarrolladores escriben código basados en lo que la gente hace en sus roles en lugar de escribir código basados en lo que pueden hacer con las herramientas de desarrollo.
- **Énfasis en el software funcionando**. En lugar de escribir documentos de diseño detallados para las personas interesadas, los desarrolladores escriben soluciones que las personas interesadas puedan evaluar en cada iteración para validar si se resuelve un requerimiento.
- **Colaboración estrecha con los clientes**. Los desarrolladores trabajan con los clientes y las partes interesadas día con día para verificar los requerimientos.

Programación Extrema (Extreme Programming)

La Programación Extrema evolucionó del desarrollo Ágil de software. En la Programación Extrema, la fase de diseño preliminar es reducida a un mínimo y los desarrolladores se enfocan en resolver unas cuantas tareas críticas. Tan pronto como esas tareas críticas sean finalizadas, los desarrolladores prueban la solución simplificada y obtienen una retroalimentación de las partes interesadas. Esta retroalimentación ayuda a los desarrolladores a identificar los requerimientos detallados lo cual evoluciona con el ciclo de vida del proyecto.

La Programación Extrema define una Historia de Usuario por cada rol de usuario. Una Historia de Usuario describe todas las interacciones que un usuario con un rol específico podría llevar a cabo con la aplicación terminada. La colección de todas las Historias de Usuario de todos los roles de usuario describen la aplicación entera.

En la Programación Extrema, los desarrolladores frecuentemente trabajan en parejas. Un desarrollador escribe el código y el otro desarrollador revisa el código para asegurar que utiliza soluciones simples y se adhiere a las mejores prácticas. El desarrollo orientado a pruebas es una práctica fundamental en la programación extrema.

Para obtener más información sobre el modelo de Programación Extrema, puedes consultar el siguiente enlace:

Extreme Programming: A gentle introduction
http://www.extremeprogramming.org/

Desarrollo Orientado a Pruebas (Test Driven Development)

En el Desarrollo Orientado a Pruebas, los desarrolladores escriben código de prueba como su primera tarea en una iteración dada. Por ejemplo, si deseamos escribir un componente que almacena detalles de una tarjeta de crédito, comenzamos escribiendo las pruebas que el componente debe pasar. Esto puede ser si cumple con un formato de número correctamente, si escribe cadenas en una tabla de base de datos correctamente o si invoca a los servicios bancarios correctamente.

Después de definir las pruebas, escribimos el componente que debe pasar esas pruebas. En iteraciones posteriores, las pruebas de la tarjeta de crédito permanecen en su lugar. Esto asegura que si se daña la funcionalidad de la tarjeta de crédito, tal vez por una refactorización de código o al añadir un nuevo constructor, se detecta esto rápidamente porque las pruebas fallan.

En Microsoft Visual Studio, podemos definir un Proyecto de Pruebas dentro de la misma solución del Proyecto principal para almacenar y ejecutar pruebas unitarias. Después de escribir las pruebas, se pueden ejecutar por separado o en grupos después de cada cambio de código. Dado que los proyectos MVC tienen el código del Modelo, la Vista y el Controlador en archivos separados, es fácil crear pruebas unitarias para todos los aspectos del comportamiento de la aplicación. Esta es la ventaja principal de MVC sobre las Web Pages y Web Forms.

Lenguaje Unificado de Modelado (Unified Modeling Language)

El Lenguaje Unificado de Modelado (UML) es una notación estándar de la industria para elaborar el diseño de cualquier aplicación que utiliza la tecnología orientada a objetos.

UML no es un modelo de desarrollo. Más bien, los diagramas UML se utilizan a menudo para planificar y documentar la arquitectura de la aplicación y sus componentes a través de todas las metodologías de desarrollo de proyectos. Al utilizar UML para diseñar y documentar una aplicación, creamos una serie de diagramas con formas y conectores estándar. Estos diagramas se pueden clasificar en tres clases:

- **Diagramas de Comportamiento**. Estos diagramas representan el comportamiento de los usuarios, las aplicaciones y los componentes de la aplicación.
- **Diagramas de interacción**. Estos diagramas son un subconjunto de los diagramas de comportamiento que se enfocan en las interacciones entre los objetos.
- **Diagramas de estructura**. Estos diagramas representan los elementos de una aplicación que son independientes del tiempo. Esto quiere decir que no cambian a través del ciclo de vida de la aplicación.