REPUBLICA BOLIVARIANA DE VENEZUELA

MINISTERIO DEL PODER POPULAR PARA LA EDUCACION

UNIVERSIDAD POLITECNICA TERRITORIAL DEL ESTADO BOLIVAR

PROGRAMA NACIONAL DE FORMACION EN INFORMATICA

INGENIERIA DEL SOFTWARE

UNIDAD III: PROCESO DE DESARROLLO DE SOFTWARE

PROFESOR:
HERNÁN RODRIGUEZ

ESTUDIANTES:
OLIVER CASTILLO
C.I: V-28.030.110

CIUDAD BOLIVAR, MES DEL 20XX

INDICE

Pagina

INDICE 2
INTRODUCCION 3
PLANIFICACION 4
Fundamentos De La Planificación 4
DESARROLLO 6
Paradigmas en el Desarrollo de Software 6
Paradigma Tradicional 6
Paradigma Orientado a Objetos 6
Paradigma de Desarrollo Ágil 9
Lenguaje Unificado De Modelado (UML) 10
Estándares en el Desarrollo de Software 12
Metodologías 13
Proceso Unificado de Desarrollo (UP) 13
PRUEBAS 14
Documentación y Artefactos 14
CONCLUSION 15
REFERENCIAS BIBLIOGRAFICAS 16

INTRODUCCION

El Proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software, es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. Algunos autores consideran un modelo de ciclo de vida un término más general que un determinado proceso para el desarrollo de software. Por ejemplo, hay varios procesos de desarrollo de software específicos que se ajustan a un modelo de ciclo de vida de espiral.

PLANIFICACION

Es el paso previo al inicio de cualquier proyecto de desarrollo y sin dudas el más importante. En este se definen los requerimientos y funcionalidades que debe tener el software, mediante el trabajo en conjunto entre los desarrolladores, el departamento de ventas, los estudios de mercado y, fundamentalmente, el contacto con el cliente.

Fundamentos De La Planificación

1. **Definición de requisitos**: Se Identifican y documentan las necesidades y expectativas de los clientes, usuarios finales y cualquier otra parte interesada, para luego clasificar y priorizar los requisitos y poder entender cuáles son esenciales y cuáles son deseables.
2. **Establecimiento del Alcance del Proyecto:** S determinan qué funcionalidades y características se incluirán en el proyecto y cuáles quedarán fuera para luego crear un documento que describa claramente el alcance del proyecto, para así evitar malentendidos futuros.
3. **Planificación del Cronograma:** Se evalúa cuánto tiempo tomará completar cada tarea o fase del proyecto para poder desarrollar un cronograma que incluya hitos importantes y fechas límite.
4. **Asignación de Recursos:** Se determinan qué recursos (humanos, tecnológicos, financieros) serán necesarios para llevar a cabo el proyecto y se asignan roles definir quién será responsable de cada tarea o componente del proyecto.
5. **Evaluación de riesgos:** Se analizan posibles obstáculos o problemas que podrían surgir durante el desarrollo. De esta manera se pueden planificar protocolos para abordar estos problemas en caso de que se presenten.
6. **Presupuesto:** Se calculan los costos asociados al desarrollo del software, incluyendo mano de obra, herramientas y otros gastos. Tener un presupuesto detallado que sirva como guía financiera para el proyecto es vital en caso de que surjan gastos no previstos o se deba recortar el presupuesto.

7. **Selección de Metodología:** Se decide qué metodología se utilizará (ágil, cascada, espiral, etc.) según las necesidades del proyecto y la organización.
8. **Documentación inicial:** Se elaboran documentos iniciales como el plan del proyecto, especificaciones funcionales y técnicas, entre otros.
9. **Revisión y Aprobación:** Se revisan los planes con las partes interesadas clave para obtener su aprobación antes de avanzar a la fase de implementación. También se realizan modificaciones al plan según la retroalimentación recibida.

DESARROLLO

La fase de desarrollo engloba todo lo relacionado a la creación del software. Involucra la implementación de la metodología, la fase de pruebas para la corrección y ajustes, y la documentación que registra todos los cambios referentes al desarrollo del software.

La implementación es parte del proceso en el que los ingenieros de software programan el código para el proyecto de trabajo que está en relación de las demandas del software, en esta etapa se realizan las pruebas de “caja blanca” y “caja negra”.

Las pruebas de software son parte esencial del proceso de desarrollo del software. Esta parte del proceso tiene la función de detectar los errores de software lo antes posible.

Paradigmas en el Desarrollo de Software

Paradigma Tradicional

Es uno de los paradigmas más antiguo, se inventó durante la creación del método estructurado. Si se aplica este paradigma, unos de los principales problemas, es que las etapas realizadas no son autónomas de las siguientes, creando una dependencia estructural y en el caso de un error atrasaría todo el proyecto. Se tiene que tener pautas bien definidas, y que no se incurra a modificación porque implicaría en que el software no cumpla con su ciclo de vida, Así como también tener en cuenta que el cliente no se vea afectado por la impaciencia.

Paradigma Orientado a Objetos

Estos modelos se basan en la **Programación orientada a objetos (POO)**; por lo tanto, se refiere al concepto de clase, el análisis de requisitos y el diseño. El modelo o paradigma orientado a objetos permite la re-utilización de software y facilita el desarrollo de herramientas informáticas de apoyo al desarrollo, el cual es simple al implementarla en una notación orientado a objetos llamado **UML**.

FUNDAMENTOS DE LA PROGRAMACION ORIENTADA A OBJETOS

Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos.

Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.

Modularidad: Se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase.

Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

Herencia: Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación.

COMPONENTES DE LA PROGRAMACION ORIENTADA A OBJETOS

Clase: Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Herencia: (Por ejemplo, herencia de la clase C a la clase D) Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D.

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos) los mismos que consecuentemente reaccionan a eventos. Se

corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

Método: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un «mensaje». Desde el punto de vista del comportamiento, es lo que el objeto puede hacer.

Evento: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente.

Mensaje: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

Propiedad o atributo: Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Estado interno: Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos).

Componentes de un objeto: Atributos, identidad, relaciones y métodos. Identificación de un objeto: Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

Paradigma de Desarrollo Ágil

Las prácticas ágiles (a veces denominadas "Agile")[1] consisten en la mejora de requisitos, investigación y soluciones mediante el esfuerzo colaborativo de equipos autoorganizados y multifuncionales junto con sus clientes/usuarios finales.[2]estos valores y principios se derivaron de, y sustentan, una amplia gama de modelos de desarrollo de software, incluidos Scrum y Kanban. Se basa en el Manifiesto Ágil, que establece principios fundamentales para guiar el proceso de desarrollo. Aquí te presento sus características clave:

1. **Iteraciones Cortas**: El trabajo se divide en ciclos cortos llamados iteraciones o sprints, generalmente de 1 a 4 semanas, lo que permite realizar entregas frecuentes y obtener retroalimentación rápida.

2. **Colaboración Continua**: Fomenta la comunicación constante entre los miembros del equipo y con los interesados (stakeholders), incluyendo al cliente, para asegurar que el producto final cumpla con las expectativas.

3. **Adaptabilidad**: Permite cambios en los requisitos incluso en etapas avanzadas del desarrollo, lo que ayuda a responder mejor a las necesidades cambiantes del cliente o del mercado.

4. **Enfoque en el Cliente**: Prioriza la satisfacción del cliente mediante la entrega continua de funcionalidades útiles y valiosas.

5. **Equipos Autogestionados**: Los equipos son responsables de su propio trabajo y toman decisiones sobre cómo abordar las tareas, lo que fomenta la creatividad y la responsabilidad.

6. **Pruebas Continuas**: Se realizan pruebas a lo largo del proceso de desarrollo para identificar y corregir errores rápidamente, asegurando una mayor calidad del producto final.

Lenguaje Unificado De Modelado (UML)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

Es una especificación basada en Booch, OMT y OSEE, de allí sus principios. Divide cada proyecto en un numero de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo. Permite describir un sistema en diferentes niveles de abstracción. se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del desarrollo de una aplicación sin definir un modelo de desarrollo.

Diagramas

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones.

En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. incluye los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de colaboración.
- Diagrama de estados.
- Diagrama de actividades.
- Diagrama de componentes.
- Diagrama de despliegue.

Notación

La notación es la parte gráfica que se ve en los modelos y representa la sintaxis del lenguaje de modelado. Por ejemplo, la notación del diagrama de clases define como se representan los elementos y conceptos como son: una clase, una asociación y una multiplicidad. ¿Y qué significa

exactamente una asociación o multiplicidad en una clase?. Un metamodelo es la manera de definir esto (un diagrama, usualmente de clases, que define la notación).

Herramientas CASE

Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

De acuerdo con Kendall y Kendall la ingeniería de sistemas asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o mas fases del ciclo de vida del desarrollo de sistemas.

Entre algunos ejemplos de herramientas CASE, se encuentran:

Erwin: PLATINUM Erwin es una herramienta de diseño de base de datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada.

EasyCASE: Esta herramienta permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, para poder crear las aplicaciones eficazmente desde procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real.

Oracle Designer: Es un juego de herramientas para guardar las definiciones que necesita el usuario y automatizar la construcción rápida de aplicaciones cliente/servidor flexibles y gráficas.Integrado con Oracle Developer

Power Designer: Es una suite de aplicaciones de Powersoft para la construcción, diseño y modelado de datos a través de diversas aplicaciones. Es la herramienta para el análisis, diseño inteligente y construcción sólida de una base de datos y un desarrollo orientado a modelos de datos a nivel físico y conceptual, que dan a los desarrolladores Cliente/Servidor la más firme base para aplicaciones de alto rendimiento.

System Architect: Esta herramienta posee un repositorio único que integra todas las herramientas, y metodologías usadas. En la elaboración de los diagramas, el System Architect conecta directamente al diccionario de datos, los elementos asociados, comentarios, reglas de validaciones, normalización, etc.

Posee control automático de diagramas y datos, normalizaciones y balance entre diagramas «Padre e Hijo», además de balance horizontal, que trabaja integrado con el diccionario de datos, asegurando la compatibilidad entre el Modelo de Datos y el Modelo Funcional.

Rational Rose: Es una herramienta de producción y comercialización establecidas por Rational Software Corporation (actualmente parte de IBM). Rose es un instrumento operativo conjunto que utiliza el Lenguaje Unificado (UML) como medio para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Este software tiene la capacidad de: Crear, Ver, Modificar y Manipular los componentes de un modelo.

Estándares en el Desarrollo de Software

Para promover un nivel mayor en el área de ingeniería de software tenemos un conjunto de reglas que ya han usado otros como normas a seguir para que la programación sea más fácil y eficiente para todos.

El estándar internacional que regula el método de selección, implementación y monitoreo del ciclo de vida del software es ISO 15504. Durante décadas se ha perseguido la meta de encontrar procesos reproducibles y predecibles que mejoren la productividad y la calidad. Algunas de estas soluciones intentan sistematizar o formalizar la aparentemente desorganizada tarea de desarrollar software. Otros aplican técnicas de gestión de proyectos para la creación del software. Sin una gestión del proyecto, los proyectos de software corren el riesgo de demorarse o consumir un presupuesto

mayor que el planeado. Dada la cantidad de proyectos de software que no cumplen sus metas en términos de funcionalidad, costes o tiempo de entrega, una gestión de proyectos efectiva es algo que a menudo falta.

Metodologías

Un proceso de software detallado y completo suele denominarse "Metodología". Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc.

Proceso Unificado de Desarrollo (UP)

Es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

Se compone de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un *incremento* del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

PRUEBAS

Documentación y Artefactos

La documentación no es más que la debilidad más frecuente en productos e instalaciones informáticos. Cabe mencionar que los actores que intervienen en el ciclo de vida del software desempeñan diversos roles. Arquitectos, diseñadores, analistas, programadores, implementadores, administradores o auditores son quienes explicitan distintos aspectos de los productos y procesos.

Un artefacto es una pieza de información que es producida o utilizada por procesos. Los artefactos son los elementos son los elementos tangibles de un proyecto, elementos que el proyecto produce o usa mientras se trabaja en busca del producto final. Éstos, pueden tomar varias formas y formatos, como, por ejemplo:

- Un documento, tal como la visión o la lista de riesgos.
- Un modelo, por ejemplo un diagrama de casos de uso o el modelo de diseño.
- Un elemento dentro de un modelo, tal como una clase, un caso de uso o un subsistema.
- Ejecutables, por ejemplo el ejecutable del prototipo.
- Código fuente.

Las actividades tienen artefactos como entrada y salida. Los roles usan artefactos para ejecutar actividades y producen artefactos durante la ejecución de sus actividades. Los artefactos son la responsabilidad sencilla del rol, creando responsabilidades fáciles de identificar y entender, promoviendo la idea de que cada pieza de información producida en un proceso de desarrollo requiere un conjunto apropiado de habilidades.

CONCLUSION

El desarrollo de software no es solo una cuestión técnica; es un esfuerzo colaborativo que combina habilidades técnicas con una comprensión profunda de las necesidades del cliente. Al adoptar enfoques flexibles y centrados en el usuario, los equipos pueden crear soluciones innovadoras que no solo resuelven problemas actuales, sino que también anticipan las necesidades futuras. Este enfoque proactivo es fundamental para navegar por los desafíos del panorama digital contemporáneo y asegurar el éxito a largo plazo en el campo del desarrollo de software.

REFERENCIAS BIBLIOGRAFICAS

Proceso Unificado, Wikipedia, La Enciclopedia Libre, (16 de julio de 2024) Recuperado el 01 de diciembre de 2024 de https://es.wikipedia.org/wiki/Proceso_unificado

Alvarado, F. *Documentación y Artefactos*, Ingeniería del Software, Blogger, (12 de julio de 2012), Recuperado el 01 de diciembre de 2024 de http://brfranciscoosunaiuty.blogspot.com/2012/07/documentacion-y-artefactos.html

Garcia, J. *Fundamentos del Enfoque Orientado a Objetos*, WordPress, (22 de abril de 2015) Recuperado el 01 de diciembre de 2024 de https://ingdelsoftwareseccion2.wordpress.com/

Proceso para el Desarrollo de Software, Wikipedia, La Enciclopedia libre, (06 de noviembre de 2024) Recuperado el 01 de diciembre de 2024 de https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software