Boujon Ignacio

Boujon Sebastián

Lavallen Lucas

Pereyra C. David

Scheffler Agustin

Facultad de Ciencia y Tecnología

Universidad Autónoma de Entre Ríos

Programación Orientada a Objetos

Tournour Alberto Adrián
Pérez Alejandro

22 /marzo/ 2024

1) ¿Cuál es el factor determinante en la aparición de técnicas de programación?

El factor determinante en la aparición de técnicas de programación es la necesidad de resolver problemas de manera eficiente y efectiva. Las demandas de la industria, los avances tecnológicos, impulsan constantemente el desarrollo de nuevas técnicas.

2) ¿Cuáles son los factores que derivan en la complejidad del software?

Existen 4 factores que derivan en la dificultad del software:

•La complejidad del dominio del problema. Un proyecto software siempre está salpicado por la complejidad propia del problema que pretende resolver. Por ejemplo, el desarrollo de un software de contabilidad tiene la complejidad propia del desarrollo, más la complejidad de las normas y del proceso de contabilidad.

• La dificultad de gestionar el proceso de desarrollo. Cuando el desarrollo software que se realiza tiene miles de líneas de código, cientos de ficheros, muchos

desarrolladores. El proceso que gestiona todos estos elementos no es trivial.

• La flexibilidad de las herramientas de software. Esta flexibilidad es un obstáculo, ya que permite a los desarrolladores usar elementos muy básicos para construir el software desde cero en vez de usar elementos más elaborados y probados construidos por otros. Este hecho se deriva de la poca confianza que existe en los desarrollos de otras personas, y de los problemas de comunicación relativos al traspaso de software.

• Comportamiento impredecible del software. El software puede verse como un sistema discreto, con multitud de variables que definen su estado en cada momento. Un pequeño cambio en una de esas variables puede llevar al sistema a un estado totalmente diferente. Esto puede verse como una alta sensibilidad al ruido, es decir, que un pequeño error (ruido) puede provocar un comportamiento totalmente erróneo.

3) ¿Cuáles son las herramientas para tratar la complejidad?

Se suelen utilizar técnicas de análisis como: descomponer, abstraer y jerarquizar.

Descomponer

Descomponer consiste en dividir sucesivamente un problema en problemas menores e independientes, cuando se crean sistemas grandes es esencial la descomposición para poder abordar el problema.

Descomposición algorítmica

El problema se descompone en tareas más simples. Luego, cada tarea se descompone a su vez otras más simples y así sucesivamente.

Descomposición orientada a objetos

El problema se descompone en objetos de cuya interacción surge la solución. Cada objeto a su vez se descompone en más objetos. La descomposición orientada a objetos tiene varias ventajas cuando se aplica a proyectos grandes

Abstraer

Para comprender un sistema muy complejo las personas solemos ignorar los detalles que nos parecen poco significativos y solemos concentrarnos en otros que consideramos esenciales, construyendo un modelo simplificado del sistema que se conoce como abstracción.

Jerarquizar

Esta técnica de análisis nos permite ordenar los elementos presentes en un sistema complejo. Ordenar los elementos en grupos nos permite descubrir semejanzas y deferencias que nos guían para comprender la complejidad del sistema

4) ¿Qué aspectos mejoro la Programación Modular?

Unas de las principales mejoras que hizo la programación modular es la idea de dividir un programa en un conjunto de módulos o subprogramas autónomos que son programados, verificados y modificados individualmente.

Estos módulos pueden ser desarrollados por distintos programadores y testeados y mantenidos de forma independiente. En otra forma

5) ¿Qué beneficios obtengo al utilizar componentes desarrollados por un tercero?

Utilizar componentes desarrollados por terceros puede ser una estrategia eficaz para acelerar el desarrollo, mejorar la calidad y reducir los costos en proyectos de software. Sin embargo, es importante evaluar cuidadosamente la calidad, la seguridad y la idoneidad de los componentes antes de integrarlos en tu sistema.

6) ¿Qué es un Tipo de Datos Abstracto (TDA)?

Un Tipo de Datos Abstractos (TDA) es un modelo matemático para tipos de datos. Se define por su comportamiento desde el punto de vista del usuario, en términos de operaciones permitidas y sus respuestas, no por su implementación.

Los TDA son fundamentales en la programación orientada a objetos, ya que permiten encapsular datos y operaciones en una sola entidad, ocultando los detalles internos y proporcionando una interfaz consistente para interactuar con los datos.

7) Ventajas de usar Tipos de Datos Abstractos (TDA)

•MODULARIDAD: encapsulamiento de especificación de datos y operaciones en un solo lugar promueve el modularidad, siendo conocido por otras unidades de programa solo por su interfaz, bajando así la carga cognitiva (variables, código, conflictos de nombres, etc.)
•MODIFICABILIDAD: es reforzada al proveer interfaces que son independientes de la implementación, dado que se puede modificar la implementación del módulo sin afecta el resto del programa. Promueve compilación separada.
•REUSABILIDAD: interfaces estándares del módulo permite que su codificación sea reusada por diferentes programas.
•SEGURIDAD: permite proteger el acceso a detalles de implementación a otras partes del programa.

8) ¿Por qué surge la necesidad de ocultar la implementación?

La necesidad de ocultar la implementación surge a partir de que el programador se preocupa por el comportamiento del objeto que exhibe y no de los detalles de cómo ese comportamiento es logrado por medio de la implementación. Al usuario le interesa conocer qué hacen esas operaciones sobre el dato, pero no cómo lo hacen.

9) ¿Por qué cree que las aplicaciones crecen en complejidad?

Requerimientos cambiantes, a medida que las necesidades de los usuarios evolucionan, los requisitos de las aplicaciones también cambian. Esto puede llevar a la incorporación de nuevas características y funcionalidades, lo que aumenta la complejidad del código.

Escalabilidad, a medida que una aplicación gana popularidad o se utiliza en entornos más grandes, debe poder manejar una mejor carga de trabajo y datos. Esto puede requerir la implementación de estructuras más complejas y técnicas de optimización.

Interconexión, las aplicaciones modernas suelen depender de una variedad de servicios externos, y bases de datos. La gestión de estas conexiones y la integración de diferentes componentes puede aumentar significativamente la complejidad del desarrollo.

Mantenimiento a largo plazo, con el tiempo, las aplicaciones necesitan actualizaciones, correcciones de errores y mejoras continuas. A medida que el código base crece, el mantenimiento se vuelve más difícil y puede requerir una gestión más sofisticada.

Evolución tecnológica, las nuevas tecnologías y herramientas pueden ofrecer beneficios significativos, pero también pueden introducir nuevas complejidades. Adoptar tecnologías emergentes puede requerir aprender nuevas API (interfaz de programación de aplicaciones, patrones de diseño y paradigmas de programación.

Requisitos de seguridad y cumplimientos, A medida que aumentan las preocupaciones sobre la seguridad de los datos y el cumplimiento de regulaciones, las aplicaciones deben incorporar medidas de seguridad más avanzadas y seguir estándares estrictos, lo que puede aumentar la complejidad del desarrollo.

10) ¿Cómo influyen las nuevas tecnologías en el desarrollo de software?

Las nuevas tecnologías tienen un impacto significativo en el desarrollo de software de varias maneras, nuevas oportunidades y soluciones, las nuevas tecnologías a menudo introduces oportunidades y soluciones para problemas existentes. Por ejemplo, el desarrollo de la inteligencia artificial y el aprendizaje automático ha permitido crear sistemas más inteligentes y automatizados en una variedad de dominios, desde reconocimientos de voz hasta la detección de fraudes.

Mayor eficiencia y productividad, las nuevas herramientas y frameworks de desarrollo suelen ofrecer características que aumentan la eficiencia y la productividad de los desarrolladores. Por ejemplo, los entornos de desarrollo integrado con funcionalidades avanzadas de autocompletado, depuración y refactorización pueden acelerar el proceso de desarrollo.

Mejoras en el rendimiento, las nuevas tecnologías a menudo introducen mejoras en el rendimiento del software. Esto puede incluir avances en la optimización de algoritmos, la velocidad de procesamiento y la gestión de recursos, lo que permite crear aplicaciones más rápidas y eficientes.

Facilitación de la colaboración, las tecnologías de colaboración en línea, como los sistemas de control de versiones distribuidos y las plataformas de desarrollo colaborativo, permiten a los equipos de desarrollo trabajar juntos de manera más eficaz, independientemente de su ubicación geográfica.

adopción de paradigmas de desarrollo modernos, las nuevas tecnologías a menudo fomentan la adopción de paradigmas de desarrollo modernos, como la programación funcional o el desarrollo basado en microservicios. Estos enfoques pueden ofrecer beneficios en términos de escalabilidad, mantenibilidad y rendimiento.

Mejoras en la experiencia del usuario, las nuevas tecnologías pueden mejorar la experiencia de usuario al permitir la creación de interfaces más intuitivas y atractivas. Por ejemplo, el uso de tecnologías de realidad aumentada o virtual puede ofrecer experiencias más inmersivas y envolventes para los usuarios.

11) Enumere y describa brevemente los factores que tienen influencia en la Calidad del Software.
La calidad del software esta influenciada tanto por factores internos, que son aquellos afectan al informático y externos que perjudican al usuario.

Confiabilidad: este es el factor de la calidad por excelencia. Se basa a su vez en dos necesidades: corrección (que el sistema cumpla con las especificaciones) y robustez (capacidad de reaccionar bien antes situaciones excepcionales).

Extensibilidad: es la facilidad de adaptarse a cambios de especificación.

Reutilización: es la capacidad de las partes de un sistema para servir en la construcción de aplicaciones distintas.

Facilidad de uso o usabilidad: Significa que el sistema pueda ser utilizado por personas de diferentes formaciones y capacidades.

Portabilidad: Facilidad de correr el sistema en otras plataformas.

Eficiencia: Usar la menor cantidad posible de recursos de hardware.

Funcionalidad: El sistema no debe pecar ni por exceso ni por defecto respecto de los requerimientos.

Oportunidad: Que el sistema este en el mercado en el momento en que los usuarios lo precisan.

12) Qué objetivos impulso la Programación Orientada a Objetos?

La metodología de orientación a objetos se impulso con una serie de objetivos que veremos a continuación:

Reducción de la entre el mundo de los problemas y el mundo de los modelos.

Aumento del nivel de complejidad de los sistemas.

Fomentar la reutilización y extensión del condigo.

Uso de prototipos.

Programación en ambientes de interfaz de usuario grafica.

Programación por eventos.