Florencia Molina 21.004

Actividad 1

a) Agregue nuevas cuestiones inherentes a los conceptos de abstracción y ocultamiento de información que a su criterio no están presentes en este documento.

- ¿Cómo funciona la abstracción en cuanto al desarrollo de software y cómo se relaciona con el ocultamiento de información?
- ¿La abstracción y el ocultamiento de información se puede aplicar a cualquier paradigma de programación?
- ¿Se aplican estos métodos a las bases de batos?
- ¿Con qué otra técnica se puede implementar estos dos conceptos en la POO?

b) Teniendo en cuenta que uno de los objetivos de la orientación a objetos es la calidad del software, defina calidad de software e indique factores para obtener dicha calidad.

La calidad del software puede definirse como el nivel en el que un sistema, componente o proceso satisface los requisitos y las necesidades del cliente o usuario, así como también se ajusta a los estándares y prácticas recomendadas de la industria. Hace referencia a la calidad de un producto digital que cumple tanto las especificaciones del cliente como las expectativas de quienes lo realizan. La calidad de software en programación hace referencia a un conjunto de características y propiedades que hacen que un programa cumpla con los requisitos y expectativas del usuario, al mismo tiempo que se ajusta a los estándares y prácticas recomendadas de la industria.

Factores: Ideas claras del cliente, capacidad de pensamiento lógico para interpretar las especificaciones, etc.

c) Cuando se habla de productividad del software, se habla de reducir los costos del software y el tiempo de desarrollo. Investigue sobre cualidades relacionadas a la productividad del software. (Se encuentra en la página 7)

Cualidades relacionadas a la productividad del software:

- Correctitud: La corrección funcional de un programa se determina por su capacidad para comportarse de acuerdo a la especificación de requerimientos funcionales. La correctitud es una propiedad matemática que establece la equivalencia entre el software y su especificación, y la precisión de la evaluación depende de la rigurosidad de la especificación. La correctitud puede evaluarse utilizando métodos experimentales, como las pruebas, o métodos analíticos, como la verificación formal de la correctitud.
- Confiabilidad: En términos informales, se considera que el software es confiable si el usuario puede confiar en él. Sin embargo, la confiabilidad se define formalmente en términos del comportamiento estadístico, como la probabilidad de que el software opere como se espera durante un intervalo de tiempo determinado. A diferencia de la correctitud, que es una cualidad absoluta, la confiabilidad es relativa. Si bien cualquier desviación de los requerimientos hace que el sistema sea incorrecto, un software incorrecto aún puede ser confiable si las consecuencias de un error no son graves.

- Robustez: La robustez de un programa se refiere a su capacidad para comportarse adecuadamente incluso en situaciones no previstas en la especificación de requerimientos. Es una cualidad difícil de definir con precisión, ya que, si se pudiera establecer lo que se debe hacer para obtener una aplicación robusta, sería equivalente a la correctitud o confiabilidad en el caso ideal mencionado anteriormente.
- Performance: En el campo de la Ingeniería de Software, el término "performance" se refiere a la eficiencia del sistema de software. Se considera que un sistema es eficiente si utiliza los recursos computacionales de manera económica. La eficiencia es un factor importante porque afecta la usabilidad del sistema. Si el sistema es demasiado lento, puede reducir la productividad de los usuarios, si ocupa demasiado espacio en disco, puede resultar costoso de ejecutar y si utiliza demasiada memoria, puede afectar al rendimiento de otras aplicaciones que se estén ejecutando en el mismo momento o ejecutarse lentamente mientras el sistema operativo intenta equilibrar el uso de memoria entre las distintas aplicaciones.
- Amigabilidad: La amigabilidad del software se refiere a la facilidad con que los usuarios pueden interactuar con él y alcanzar sus objetivos de manera efectiva, eficiente y satisfactoria. Esto implica que el software debe ser fácil de aprender y de recordar cómo usarlo, que la interfaz de usuario debe ser clara y fácil de entender, que el software debe proporcionar retroalimentación útil y en tiempo real, y que el software debe permitir a los usuarios personalizar la forma en que interactúan con él. Además, el software debe ser consistente en su comportamiento y en su diseño para que los usuarios puedan predecir cómo funcionará en diferentes situaciones y evitar errores. La amigabilidad es una cualidad importante porque puede influir en la aceptación y adopción del software por parte de los usuarios y, por lo tanto, en su éxito en el mercado.
- Verificabilidad: La verificabilidad también puede estar relacionada con la documentación y la trazabilidad de los requisitos y decisiones de diseño. Un sistema de software con una documentación clara y detallada es más fácil de verificar que uno con documentación escasa o inexistente. Además, la trazabilidad permite seguir el proceso de desarrollo y verificar que se han cumplido los requisitos establecidos.
- Mantenibilidad Es importante tener en cuenta que la evolución del software es un proceso continuo y que puede ser necesario para mantener la relevancia y competitividad del software en el mercado.
- Reparabilidad: La reparabilidad de un sistema de software se refiere a la capacidad de corregir defectos en el software con una carga limitada de trabajo. A diferencia de otros campos de ingeniería, en los que a veces es más económico reemplazar partes o componentes en lugar de repararlos, en el software la reparabilidad se basa en la capacidad de identificar y corregir errores con una cantidad razonable de trabajo.
- Evolucionabilidad: La evolucionabilidad es una cualidad importante tanto del producto como del proceso. En el proceso de desarrollo de software, éste debe ser capaz de adaptarse a nuevas técnicas de gestión y organización, cambios en la educación en ingeniería, etc. Además, la evolucionabilidad implica otros conceptos como familias de programas cuyo propósito es fomentar la evolucionabilidad.
- Reusabilidad: La reusabilidad es similar a la evolucionabilidad en el sentido de que en la evolucionabilidad se modifica un producto para construir una nueva versión del mismo producto, mientras que en la reusabilidad se utiliza un producto, posiblemente con modificaciones menores, para construir otro producto.

- Portabilidad: El software es portable si puede ser ejecutado en diferentes entornos, tanto en plataformas de hardware como en ambientes de software, como determinado sistema operativo.
- Comprensibilidad: La comprensibilidad de un sistema de software se refiere a su capacidad de ser comprendido y entendido. Algunas tareas son inherentemente más complejas que otras, por lo que algunos sistemas pueden ser más fáciles de comprender que otros.
- Interoperabilidad: La interoperabilidad se refiere a la capacidad de un sistema de coexistir y colaborar con otros sistemas.
- Productividad: La productividad en el contexto del proceso de producción de software, se refiere a la eficiencia del proceso y se puede medir mediante la cantidad de trabajo realizado en un tiempo determinado.
- Oportunidad: La puntualidad es una característica del proceso que se refiere a la capacidad de entregar un producto en el plazo establecido. Históricamente, los procesos de producción de software han carecido de esta característica, lo que llevó a la "crisis del software" y, en consecuencia, al surgimiento de la ingeniería de software. Incluso en la actualidad, muchos procesos fracasan en lograr resultados en el tiempo previsto.
- Visibilidad: La visibilidad en un proceso de desarrollo de software es esencial para garantizar el éxito del proyecto. Al mantener una documentación clara y mantener el estado del proyecto en forma adecuada, se pueden reducir los riesgos y mejorar la eficiencia del equipo.