Procesos de Ingeniería de Software

Actividad N°2 [Modelos de desarrollo de software]

Iván David Valderrama Corredor

Pontificia Universidad Javeriana Cali

Facultad de Ingería

Ingeniería de Sistemas y Ciencias de la Computación

Santiago de Cali

2018

Cualidades del Software



* -Correctitud: Un sistema es correcto cuando se comporta de acuerdo a los requerimientos del cliente. En otras palabras, un sistema es correcto si resuelve el problema real que causó su desarrollo. La correctitud es una propiedad matemática que establece la igualdad entre el software y su especificación, por lo que cuanto más específica haya sido en las instrucciones, más precisa y sistemática podrá ser su evaluación. Posteriormente se verá que la correctitud puede ser valorada mediante diversas técnicas, algunos de enfoque experimental como las pruebas, otros de enfoque analítico como verificación formal de la correctitud.
* Confiabilidad: El software es confiable si se comporta de acuerdo a los requerimientos del usuario. En otros términos, es la calidad que se puede esperar de una aplicación que lleve a cabo las operaciones establecidas y con la claridad requerida. Pueden tenerse aplicaciones correctas diseñadas para requerimientos “incorrectos”, por lo que la correctitud del software puede no ser suficiente para garantizar al usuario que el software se comporta como “es esperado”.
* Robustez: Un programa es robusto si se comporta en forma razonable aún en situaciones que no fueron pronosticadas en la descripción de los requerimientos. La robustez es igual a la distancia del caos, es decir, mientras menor es la distancia al caos, mayor solidez posee el sistema.
* Performance: Se puede medir la eficiencia del sistema de acuerdo a dos dimensiones: Los recursos necesarios que abarcan la construcción y desarrollo del software, y los recursos necesarios que comprende la ejecución de la aplicación. Cabe destacar, que un sistema es eficiente cuando se utilizan los equipos computacionales en forma económica. Por ejemplo, si es muy lento reduce el entendimiento de los usuarios, si usa demasiado espacio de disco puede ser muy costoso de ejecutar, si utiliza demasiada memoria puede afectar al resto de las aplicaciones que se están ejecutando mientras el sistema operativo intenta balancear el uso de la memoria.
* Amigabilidad: Un sistema es amigable cuando el usuario encuentra la interfaz fácil de manejar. La amigabilidad está dada por la habilidad con que el sistema puede configurarse y ajustarse al ambiente de hardware. Un sistema que produce respuestas erróneas no es amigable sin importar lo “atractiva” que sea la interfaz de usuario, del mismo modo que un sistema que produce respuestas más tardías de lo que requiere el usuario no es amigable aunque estas respuestas sean desplegadas en colores.
* Verificabilidad: Un software es verificable si sus propiedades pueden ser verificadas sencillamente. Ejemplo, la correctitud o la performance de un sistema son propiedades que concierne comprobar. La verificabilidad es en general una cualidad interna pero a veces puede ser externa, por ejemplo, en muchas aplicaciones de seguridad crítica, el cliente solicita la verificación de algunas propiedades.
* Mantenimiento: Es la forma fácil de corregir y remediar fallas que pueda tener algún software. El mantenimiento también puede aplicarse a la reparación los problemas que surgen en la aplicación después de la liberación o agregarle al producto que no estaban en las especificaciones originales. El mantenimiento abarca un grupo amplio de actividades que tiene que ver con las modificaciones de un software existente la lograr una mejora. Existen tres tipos de mantenimiento:
* El mantenimiento correctivo: El mantenimiento correctivo es la eliminación de errores excedentes presentes en el producto al ser liberado, así como errores implantados al software durante su mantenimiento.
* Mantenimiento adaptativo: Involucra el ajuste de la aplicación a ajustes en el entorno, por ejemplo, la creación de hardware o del sistema operativo. En este asunto la necesidad de cambios al software no puede ser atribuida a una particularidad del software en sí mismo como la inhabilidad de prestar cierta funcionalidad demandada por el usuario o errores residuales, sino que se producen debido a cambios en su entorno.
* El mantenimiento perfectivo: Implica cambios en el software para perfeccionar sus cualidades, los cuales se deben a la necesidad de cambiar las funciones brindadas por el software, añadir nuevas funciones, renovar la performance, facilitar su manejo, entre otras. Estos cambios pueden ser producidos tanto por el ingeniero de software para perfeccionar el estatus del producto en el mercado, como por el cliente debido a nuevos requerimientos.
* La reusabilidad: Se refiere a que una aplicación puede utilizarse en otras aplicaciones. Es complicado lograr reusabilidad, se debe examinar el instante de desarrollar los componentes del software; un modo para conseguir reusabilidad es el manejo de diseño orientado a objetos.  Diferentes metodologías de software pueden verse como tentativas de reutilizar el mismo proceso para la construcción de productos diferentes, y los modelos de ciclo de vida también son intentos de reutilizar procesos de alto nivel.
* Portabilidad: Se refiere a la manera en que los clientes pueden acceder a los productos ya que un software portable es mucho más fácil de obtener por los clientes dado que pueden acceder a dicho software. El software es portable si puede ser desarrollado en distintos ambientes, refiriéndose este último tanto a plataformas de hardware como a ambientes de software como puede ser determinado sistema operativo.
* Comprensibilidad: Algunos sistemas de software son más cómodos de comprender que otros, ciertas tareas son sustancialmente más complicadas que otras. La comprensibilidad es una cualidad interna del producto y ayuda a lograr muchas de las otras cualidades como evolucionabilidad y verificabilidad. La comprensibilidad no es más que ejecute su función de acuerdo a lo que el usuario predice.
* Interoperabilidad: Es la destreza que posee un sistema para coexistir e interactuar con otros, por ejemplo, la habilidad de un procesador de texto de incluir gráficas producidas por un paquete de gráficos.
* La productividad: Es una cualidad del proceso de producción de software, calcula la eficiencia del proceso. Un proceso eficiente resulta en una entrega más rápida del producto. Los ingenieros originan software individualmente a cierta tasa, la cual puede alterarse considerablemente entre individuos con habilidad distinta. Cuando los individuos forman un equipo, la productividad de éste es alguna función de las productividades individuales, y en general esta productividad combinada es menor que la suma de las partes.
* Oportunidad: La oportunidad es una cualidad del proceso que se refiere a la habilidad de liberar el software a tiempo. Esto puede ser un arma de doble filo ya que muchos procesos fracasan en alcanzar los resultados a tiempo. La oportunidad en sí misma no es una cualidad útil, aunque llegar tarde puede llevar a perder oportunidades en el mercado, entregar un producto a tiempo que carece de otras cualidades como confiabilidad o performance, no tiene sentido. Entregar un producto a tiempo requiere una agenda planeada cuidadosamente, con un trabajo de estimación acertado y puntos de revisión especificados claramente y verificables.