REQUISITOS NO FUNCIONALES

ASTRID CAROLINA DÍAZ GÓMEZ

ARQUITECTURA DE SOFTWARE

INGENIERÍA DE SISTEMAS

UDEA

2020-1

**Access Control**

Hace referencia al mecanismo que en función de la identificación ya autenticada permite acceder a datos o recursos, se enfoca en restringir o permitir el acceso a las diferentes funcionalidades de la aplicación de software que se está diseñando.

Como ejemplo en una aplicación bancaria en la que los usuarios que trabajan con ella tienen diferentes roles y permisos, cada uno de estos de acuerdo con el cargo que desempeñan; un usuario puede ingresar solo a las funciones con las que tiene permisos y si requiere realizar más tareas deberá solicitarlas de forma que se pueda tener los accesos correspondientes controlados.

Debe realizarse un control de acceso a los usuarios para evitar accesos no

autorizados al sistema.

Debe realizarse un control de acceso a los usuarios para evitar accesos no

autorizados a las bases de datos.

Ante accesos no autorizados (o intentos reiterados) el sistema debe responder

con acciones como alertas visuales o sonoras, bloqueos del terminal, avisos a los

administradores

Dependiendo de los privilegios de acceso del usuario el sistema limita su

actividad, restringiendo su acceso sólo a los lugares o a las acciones autorizadas

**Agility**

La agilidad es la capacidad de diseñar los pasos a seguir en función del aprendizaje que surge de los resultados producidos por los pasos anteriores y de la habilidad de adaptarse a los cambios del contexto, con el fin de alcanzar los objetivos propuestos de manera efectiva.

Avanzar de a pasos, donde cada pequeño avance nos permita medir los resultados, verificar supuestos, sacar conclusiones y aprender de ello. Implica además trabajar de modo tal de obtener un producto funcional en cada paso, un producto cuya utilidad pueda ser comprobable.

Un ejemplo de esto es como se distribuyen los trabajos a realizar en el desarrollo de un proyecto de software de forma que se puedan generar componentes de valor los cuales puedan ser reforzados y mejorados en conjunto, para dar como resultado una aplicación estable y que cumpla con los requerimientos del cliente.

**Analyzability**

Analizabilidad es la capacidad de un producto software de ser diagnosticado por deficiencias o causas de fallos en este, o para las partes a ser modificadas o identificadas. [ISO 9126]

Un ejemplo de esto se da en el análisis, teniendo en cuenta que el proyecto debe tener un tiempo de vida útil y que este puede tener algunos errores, se busca la forma de poder informar de estos errores para darle el correcto soporte, tanto en los informes como en la estructura del código.

Los requerimientos del software deben estar reflejados e identificados en la

documentación del análisis

Para que el sistema sea completo, el análisis tiene que satisfacer todos los

requerimientos descritos en los requisitos software. Hay que tener en cuenta las

sucesivas modificaciones que suelen sufrir las especificaciones de requisitos software,

lo cual implica modificaciones a realizar en el análisis.

El análisis debe reflejar los requerimientos de almacenamiento, que se tendrán en

cuenta al evaluar la eficiencia de almacenamiento

**Simplicity**

Simplicidad es la ausencia de complejidad o dificultades. En el desarrollo de software puede resultar de interés diferenciar entre complejidades esenciales y accidentales. Determinar si una dificultad en un diseño o programa es esencial o accidental, nos permite atacar las dificultades accidentales, buscando soluciones más simples.

En el análisis se ve casos en los que se puede tener una propuesta para un proyecto que se puede hacer de diferentes formas, pero con excepción de que unas pueden tener un mayor costo de esfuerzo tiempo y presupuesto que otras, gracias a la simplicidad es que se puede establecer la mejor opción para empezar a trabajar.

**Confidentiality**

Es el grado en el que los datos tienen atributos que aseguran que los datos son sólo accedidos e interpretados por usuarios autorizados en un contexto de uso específico.

La confidencialidad, requiere que la información sea accesible de forma única a las personas que se encuentran autorizadas. Es necesario acceder a la información mediante autorización y control.

La confidencialidad hace referencia a la necesidad de ocultar o mantener secreto sobre determinada información o recursos. Su objetivo es, prevenir la divulgación no autorizada de la información sobre las organización.

**Distributivity**

Es un diseño de sistema de marketing vertical para cumplir con los requisitos tanto de un fabricante como de sus distribuidores o entre una red de computadoras. Es el programa de software que se ejecuta en computadoras en red para coordinar y comunicar acciones basadas en procesos que ya son familiares para las computadoras.

El software está distribuido si sus partes se encuentran localizadas en distintos dispositivos de proceso o almacenamiento

La distribución de clases en ficheros favorece la expansión del código al poderse localizar fácilmente cada uno de los módulos del sistema.

**Modularity**

La modularidad se refiere a la medida en que un software o una aplicación web puede dividirse en módulos más pequeños. La modularidad del software indica que el número de módulos de aplicación son capaces de servir a un dominio empresarial específico.

La modularidad tiene éxito porque los desarrolladores utilizan código preescrito, lo que ahorra recursos. En general, la modularidad proporciona una mayor capacidad de gestión del desarrollo de software.

Se puede lograr un diseño modular eficaz si los módulos divididos se pueden resolver, modificar y compilar por separado.

Para construir un software con un diseño modular efectivo hay un factor de “Independencia Funcional” que entra en juego. El significado de Independencia funcional es que una función es de naturaleza atómica, de modo que realiza solo una tarea del software sin o con la menor interacción con otros módulos.

La arquitectura de software en capas es un ejemplo de modularidad.

* Los métodos deben tener un tamaño reducido
* Los métodos no deben ser excesivamente pequeños, puesto que, en ese caso, se estaría alcanzando una modularidad exagerada, es decir, se podría llegar a una atomización de los componentes
* Las clases deben tener el número apropiado de métodos
* Las clases deben tener el número apropiado de atributos.
* Las clases deben tener un único objetivo. El concepto de modularidad se basa en que cada módulo represente una única funcionalidad
* Dentro de una jerarquía de clases, cuanto mayor sea el número de relaciones de herencia, más disminuirá la modularidad debido a las interrelaciones que se crean
* Las jerarquías de clases deben tener el número apropiado de clases, atendiendo a la finalidad de la jerarquía
* Las clases sólo deben permitir el acceso a sus atributos a través de sus métodos.
* Las clases deben ofrecer constructores, destructor y funciones operador
* Las clases deben ocultar todos sus atributos. La presencia de atributos protegidos y, en mayor medida, de atributos públicos disminuye la modularidad de la clase ya que se podría acceder a esta información sin utilizar la interfaz prevista.
* Las clases deben disponer de comentarios que sigan una estructura estándar
* Las clases deben llevar una documentación técnica completa, que describa su estructura, uso y utilidad, y explicando su modo de empleo y su funcionamiento.
* Un programa modular no permitiría a un método utilizar atributos externos de otras clases, dado que a esta información se debe acceder a través de los servicios proporcionados por la clase.
* Las clases deben ser autosuficientes y necesitar el menor número posible de servicios de otras clases
* Las clases abstractas favorecen la modularidad debido a que permiten definir esquemas de comportamiento comunes a otras clases

**Webgrafía**

Technopedia. (2012). Modularity. *www.techopedia.com.* Recuperado de

<https://www.techopedia.com/definition/24772/modularity#:~:text=In%20software%20engineering%2C%20modularity%20refers,serving%20a%20specified%20business%20domain>.

TECNOSeguro. (). ¿Qué es un Sistema de Control de Acceso? *www.tecnoseguro.com.* Recuperado de [https://www.tecnoseguro.com/faqs/control-de-acceso/que-es-un-control-de-acceso#:~:text=La%20definición%20más%20generalizada%20de,formas%20y%20para%20diversas%20aplicaciones](https://www.tecnoseguro.com/faqs/control-de-acceso/que-es-un-control-de-acceso#:~:text=La%20definici%C3%B3n%20m%C3%A1s%20generalizada%20de,formas%20y%20para%20diversas%20aplicaciones).

Mishra R. (2019). Effective Modular Design in Software Engineering. *www.geeksforgeeks.org.* Recuperado de <https://www.geeksforgeeks.org/effective-modular-design-in-software-engineering/>

Buonamico D. (2018). ¿Qué es -en esencia- la Agilidad? *medium.com* Recuperado de [https://medium.com/kleer/qué-es-en-esencia-la-agilidad-5e2cee81a92b](https://medium.com/kleer/qu%C3%A9-es-en-esencia-la-agilidad-5e2cee81a92b)

Lay D. (2017). ¿Que es la agilidad? *es.slideshare.net.* Recuperado de <https://es.slideshare.net/davidlaym/que-es-la-agilidad-73391658>

Globe. (2018). Analizabilidad. *www.globetesting.com.* Recuperado de <https://www.globetesting.com/glosario/analizabilidad/>

Fuertes J. (2002). Modelo de calidad para el software orientado a objetos. *oa.upm.es* Recuperado de <http://oa.upm.es/34988/1/TD_Fuertes_JOSE_LUIS.pdf>

Iso 2500. (2019). Confidencialidad. *iso25000.com.* Recuperado de <https://iso25000.com/index.php/normas-iso-25000/iso-25012/109-confidencialidad#:~:text=Grado%20en%20el%20que%20los,IEC%2013335-1%3A2004>.

SGSI. (2018). Los tres pilares de la seguridad de la información: confidencialidad, integridad y disponibilidad. *www.pmg-ssi.com.* Recuperado de <https://www.pmg-ssi.com/2018/02/confidencialidad-integridad-y-disponibilidad/>

Uqbar Foundation. (2020). Atributos de calidad. *wiki.uqbar.org.* Recuperado de <http://wiki.uqbar.org/wiki/articles/atributos-de-calidad.html>