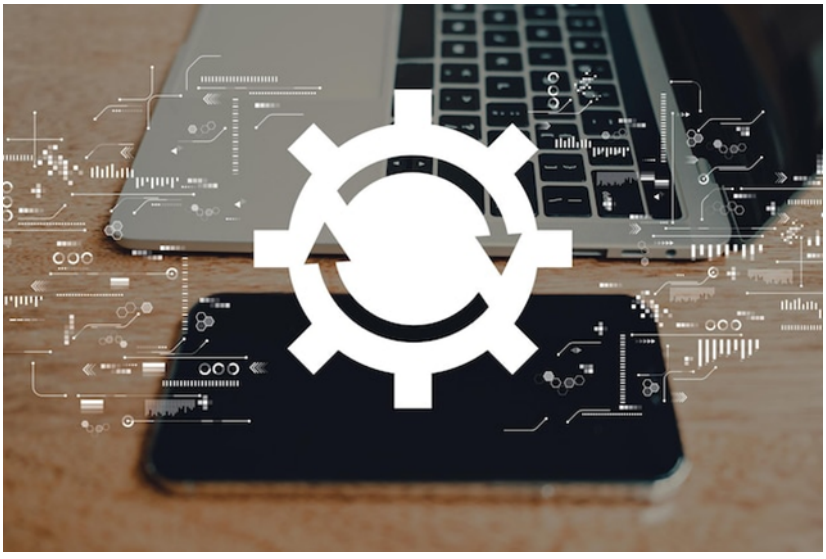
TÉCNICAS ORIENTADAS A OBJETOS PARA EL ANÁLISIS DE REQUERIMIENTOS



INTRODUCCIÓN

El desarrollo basado en componentes (CBD) es un área nueva y poco explorada. Se le suele asociar e incluso confundir con el desarrollo orientado a objetos (OOD); a pesar de que ambos enfoques están relacionados, los mismos son aplicables a sistemas de distinto porte. Generalmente OOD es asociado con Programming-in-the-Small, mientras que CBD es más aplicable a Programming-in the-Large.
Los sistemas de hoy en día son cada vez más complejos, deben ser construidos en tiempo récord y deben cumplir con los estándares más altos de calidad. Para hacer frente a esto, se concibió y perfeccionó lo que hoy conocemos como Ingeniería de Software Basada en Componentes (ISBC)

Actualmente el desarrollo de software basado en componentes se ha transformado en el componente más seguro tanto para la construcción de grandiosos sistemas como para aplicaciones de software.

COMPONENTE

Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

CARACTERÍSTICAS DE UN COMPONENTE

Identificable: Debe tener una identificación que pueda acceder fácilmente a sus servicios que permita su clasificación.

Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiarán a lo largo de su implementación.

Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.

Puede ser remplazado por otro componente: Se puede reemplazar por nuevas versiones u otro componente que lo reemplace y mejore.

Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

Bien documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.

Es genérico: Sus servicios deben servir para varias aplicaciones.

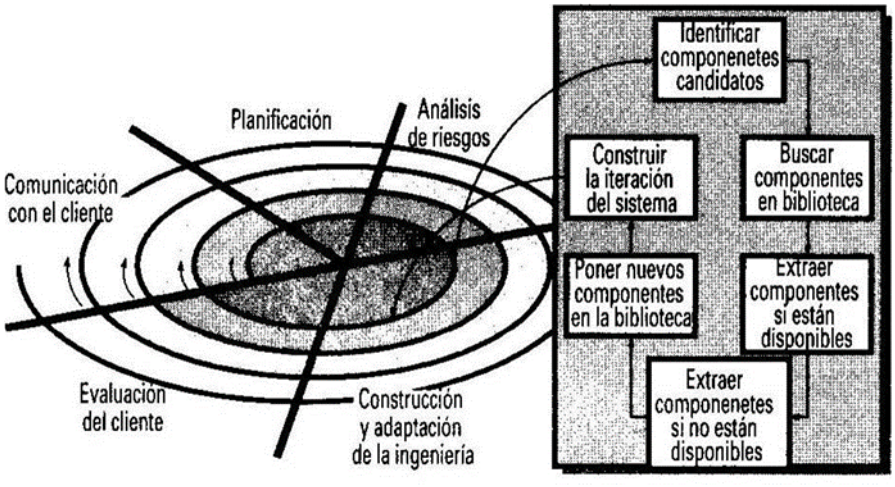
Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.

Independiente de la plataforma: Hardware, Software, S.O.

¿QUE ES ISBC?

La ingeniería del software basada en componentes (ISBC) es un proceso que se centra en el diseño y construcción de sistemas basados en computadora que utilizan «componentes» de software reutilizables.

El modelo de desarrollo basado en componentes incorpora muchas de las características del modelo en espiral. Es evolutivo por naturaleza y exige un enfoque iterativo para la creación del software.



OBJETIVOS

* Los desarrollos tradicionales de aplicaciones incurren en altos costos y en una inversión de tiempo extensa.

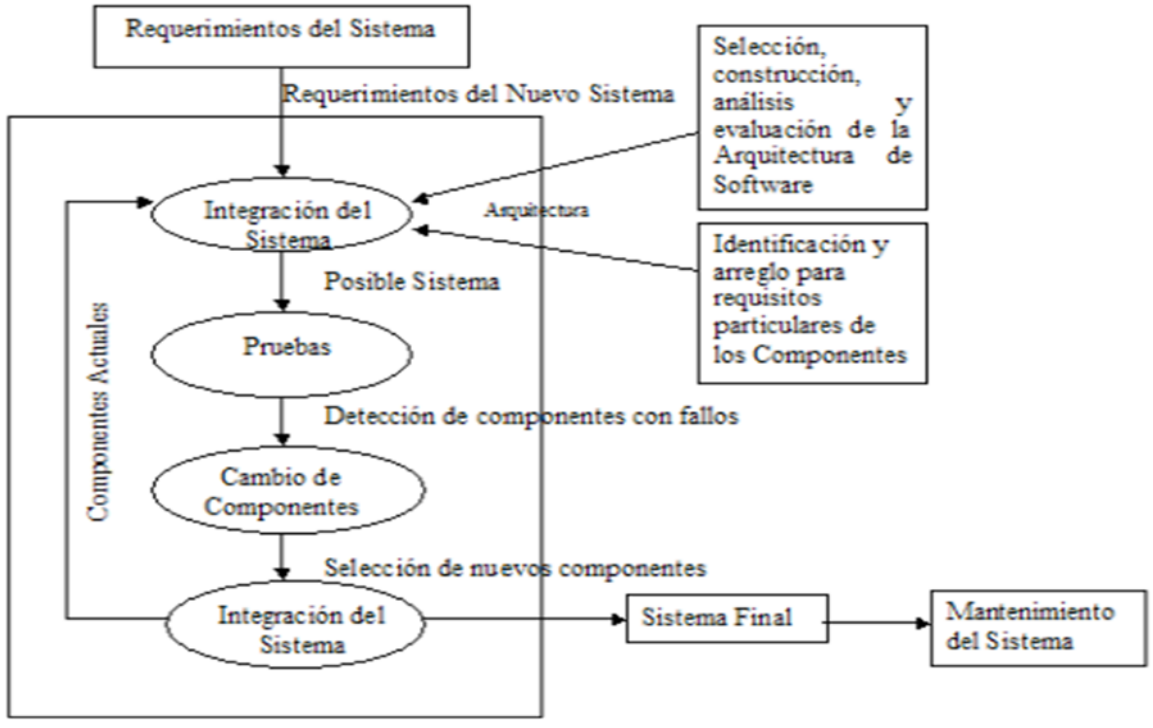
El DSBC busca, dentro de otros objetivos:

* Reducir el tiempo de trabajo.
* El esfuerzo que requiere implementar una aplicación y los costos del proyecto, y, de esta forma,
* Incrementar el nivel de productividad de los grupos desarrolladores.

* Minimizar los riesgos globales.

DESVENTAJAS

* Genera mucho tiempo en el desarrollo del sistema
* Modelo costoso
* Requiere experiencia en la identificación de riesgos
* Genera mucho trabajo adicional
* Cuando un sistema falla se pierde tiempo y coste dentro de la empresa
* Exige una cierta habilidad en los analistas (es bastante difícil)



DESCRIPCIÓN DE LAS ETAPAS DEL MODELO

Análisis de Requerimientos: En esta etapa del ciclo de vida, los procesos y las necesidades del negocio se descubren y se expresan en los casos de uso.

Selección, construcción, análisis y evaluación de la Arquitectura de Software: "La arquitectura del software define un sistema en términos de componentes computacionales y la interacción entre ellos". Estas tareas podrán ser realizadas con éxito si se exponen las propiedades externas de los componentes que harán parte de la aplicación y las relaciones entre ellos.

Identificación y arreglo para requisitos particulares del Componente: En esta actividad, los componentes deben ser seleccionados por los requerimientos funcionales y de calidad que satisfaga cada componente. Luego de haber sido identificados los componentes que serán integrados al sistema, se debe evaluar si el componente necesita ser sujeto a alguna modificación.

Integración del Sistema: En esta actividad se debe examinar, evaluar y determinar cómo va a ser la comunicación y la coordinación entre los componentes que harán parte del sistema. Luego debe ensamblarse el sistema y proseguir con una serie de pruebas que determinarán si los componentes seleccionados son los adecuados.

Pruebas: Posterior a la integración de los componentes se debe proceder con las pruebas, esto implica evaluar el funcionamiento de los componentes que fueron integrados en el sistema, si algún componente demuestra no estar funcionando de forma correcta se debe pensar en la posibilidad de reemplazarlo o modificarlo para luego proceder con la reintegración.

Mantenimiento: En el período del mantenimiento, se lleva a cabo un proceso similar al desarrollado en la POO, esto es vigilar el correcto funcionamiento del sistema, corregir fallas en el comportamiento, etc.

El disponer de componentes software no es suficiente para desarrollar aplicaciones, ya provengan éstos de un mercado global o sean desarrollados a medida para la aplicación. Un aspecto crítico a la hora de construir sistemas complejos es el diseño de la estructura del sistema.

MARCOS DE TRABAJO

ARQUITECTURA SOFTWARE

La reutilización de arquitecturas software se define dentro de un marco de trabajo (framework, o abreviadamente MT). En general, un MT se suele definir de la siguiente forma: "Un MT es el esqueleto de una aplicación que debe ser adaptado a necesidades concretas por el programador de la aplicación".

La arquitectura de software nace como una herramienta de alto nivel para cubrir distintos objetivos:

* Comprender y manejar la estructura de las aplicaciones complejas.
* Reutilizar dicha estructura (o partes de ella) para resolver problemas similares.
* Planificar la evolución de la aplicación, identificando sus partes mutables e inmutables, así como los costes de los posibles cambios.
* Analizar la corrección de la aplicación, y su grado de cumplimiento respecto a los requisitos iniciales.
* Permitir el estudio de alguna propiedad específica del dominio.

PROGRAMACIÓN ORIENTADA A COMPONENTES (POC)

La POC nace con el objetivo de construir un mercado global de componentes software, cuyos usuarios son los propios desarrolladores de aplicaciones que necesitan reutilizar componentes ya hechos y probados para construir sus aplicaciones de forma más rápida y robusta.

TENDENCIAS ACTUALES DE LA POC

TECNOLOGÍAS DE COMPONENTES ESTANDARIZADAS

- La adecuación de los lenguajes de programación.
- Diseño de nuevos lenguajes y modelos de componentes.
- La construcción de herramientas de desarrollo y marcos de trabajo para componentes.
- La aplicación de técnicas formales para razonar sobre las aplicaciones desarrolladas a base de componentes.

En cuanto a los lenguajes de programación, sólo hay unos pocos que realmente incorporan conceptos suficientes para realizar una programación orientada a componentes: Oberón, Java, Ada95, Modula-3 y Component Pascal.

TECNOLOGÍAS DE COMPONENTES ESTANDARIZADAS

CORBA (CommonObjectRequestBrokerArchitecture — arquitectura común de intermediarios en peticiones a objetos): Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

DCOM (DistributedComponentObjectModel-Modelo de Objetos de Componentes Distribuidos): Es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.

NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Enterprise JavaBeans: Modelo de componentes basado en arquitectura cliente servidor: Esta plataforma ofrece una solución multiplataforma, de fácil reutilización, integración universal con otros componentes además de la máquina virtual de java. La tecnología java, ha estado a la vanguardia del DSBC y constituye una referencia clave en este tema.

La programación de aplicaciones distribuidas se basa en un conjunto de servicios que proporcionan a los componentes el acceso a los recursos compartidos de una forma segura y eficiente. Estos servicios suelen englobarse en las siguientes categorías básicas:

* Comunicaciones Remotas
* Servicios de Directorio
* Seguridad
* Transacciones
* Gestión

CONCLUSION

La ingeniería del software basada en componentes proporciona unos beneficios inherentes en lo concerniente a calidad del software, productividad del desarrollador y costo general del sistema. Sin embargo, es preciso vencer muchas dificultades antes de que el modelo de proceso CBSE se utilice ampliamente en la industria. Además de los componentes del software, un ingeniero del software puede adquirir toda una gama de elementos reutilizables. Entre estos se cuentan las representaciones técnicas del software (por ejemplo, especificación, modelos de arquitectura, diseños y códigos), documentos, datos de prueba e incluso tareas relacionadas con los

procesos (por ejemplo, técnicas de inspección). Un desarrollo basado en componentes no garantiza el éxito de un sistema, depende de cómo es definido y aplicadas las metodologías, procesos y técnicas de la CBSE.

* La idea principal del desarrollo por componentes es construir sistemas desde componentes existentes.

* Lo ideal es codificar lo menos posible para implementar funcionalidades.

Referencias:

DESARROLLO BASADO EN COMPONENTES. (s.f.-c). prezi.com. https://prezi.com/p1x_2efoedrc/desarrollo-basado-en-componentes/

https://img.freepik.com/fotos-premium/actualizacion-software-concepto-smartphone-portatil-sobre-mesa_102583-6287.jpg?w=740