NIVELES DE ABSTRACCIÓN
-Clases, estructuras
-Módulos, paquetes, librerías
- Servicios (procesos, grupos de procesos)

CARACTERÍSTICAS DESEABLES DE UN PRODUCTO DE SOFTWARE
Mantenibilidad ¡ Confiabilidad ¡ Eficiencia ¡ Usabilidad

TIPOS DE REQUERIMIENTOS y también arquitectura de drivers
REQUERIMIENTOS FUNCIONALES: Describen el comportamiento del sistema
REQUERIMIENTOS no funcionales: Propiedades que el sistema debe tener limite y frontera:restricciones del sistema

RESTRICCIONES
Técnicas: El uso de un lenguaje de programación en particular
De negocio: Un presupuesto limitado o un fecha de terminación estricta
Legales o regulatorias: En la unión europea hay limitaciones en la información que se puede recolectar

DESARROLLO BASADO EN COMPONENTES
Los componentes brindan una funcionalidad que se persigue con interfaces bien definidas que permiten que se integren en el software que se va a construir

PROGRAMACIÓN ORIENTADA A COMPONENTES
Permite construir programas a partir de componentes de software, que son bloques de código reusables e independientes
La COP es una programación basada en la interfaz, los clientes no necesitan un conocimiento de cómo el componente implementa su interfaz
CARACTERISTICAS DE UN COMPONENTE
Autocontenido ¡ Un componente no debe requerir la reutilización de otros componentes para cumplir su función ¡ Si requiere la interacción con otros componentes, el componente principal debe verse como el componente de mas alto nivel ¡ Accesible solo por medio de su interfaz ¡ Servicios inmutables ¡ La implementación puede variar ¡ Tiene que estar documentado

PRINCIPIOS SOLID
SOLID es una serie de principios y buenas prácticas que deben tenerse como base antes de proponer una arquitectura de software

ACOPLAMIENTO SE DEFINE COMO EL NIVEL DE INTERDEPENDENCIA ENTRE VARIOS COMPONENTES DE SOFTWARE

CAMBIOS
Los cambios en el software pueden introducir una serie de errores ¡ Mas tiempo y esfuerzo implica dinero ¡ Los principios SOLID no son elementos sofisticados ¡ Se traducen en costos de mantenimiento a largo plazo

PROPIEDADES DE UNA ARQUITECTURA
Estructurales, define los componentes de un sistema (módulos, objetos, filtros, etc.) y la manera en que estos están agrupados e interactúan unos con otros ¡ Extrafuncionales, la forma en que satisface los requerimiento no funcionales ¡ Familias de sistemas relacionados, debe basarse en patrones repetibles que es común encontrar en sistemas similares, reutilización de bloques de construcción arquitectónica

ATRIBUTOS DE CALIDAD
Los atributos de calidad son requerimientos no funcionales ¡ Describen: ¡ Las cualidades de los requerimientos funcionales ¡ Las propiedades en general del sistema ¡ Proporciona una medida cualitativa de que tan bién se comporta nuestro sistema respecto a una dimensión en particula

EJEMPLO
“Cuando un usuario presione el botón de búsqueda despues de haber tecleado algunas palabras claves, el sistema debe proporcionar una lista de productos que coicidan con las palabras de búsqueda en un tiempo máximo de 100 milisegundos”

CONSIDERACIONES IMPORTANTES
Los atributos de calidad deben ser: ¡ Medibles ¡ Poderse probar

CARACTERISTICAS DE UN COMPONENTE ¡ Autocontenido ¡ Un componente no debe requerir la reutilización de otros componentes para cumplir su función ¡ Si requiere la interacción con otros componentes, el componente principal debe verse como el componente de mas alto nivel ¡ Accesible solo por medio de su interfaz ¡ Servicios inmutables ¡ La implementación puede variar ¡ Tiene que estar documentado **interfaz** de un componente ¡ Determina las operaciones que implementa un componente. Usualmente son los atributos y métodos publicos que el componente define mas los eventos que emite ¡ Eventos ¡ Esepecifican la forma que el componente notifica al exterior una respuesta a un estímulo externo o un cambio en alguna condición externa	**Single responsibility** -que una interfaz o metodos solo aga una accion **open closed** -abierto a extenciones y cerrado a modificar el codigo original **liskob sustitucion** -los objetos deben ser remplazados por subtipos conservando el programa correcto **interfaz segregation** -es preferible tener muchas interfases a tener unica sola **dependencias de inversion** -los modelos de alto nivel no deben depender de los modelos de bajo nivel ambos deben depender de su abstraccion

VISIÓN ORIENTADA A OBJETOS ¡ En el contexto de la ingeniería de software orientada a objetos, un componente contiene un conjunto de clases que colaboran ¡ Cada clase dentro del componente incluye todos los atributos y operaciones para su implantación ¡ Se deben definir las interfaces que permiten que las clases se comuniquen entre si ¡ Para lograr esto, es importante el modelo de requerimientos y la elaboración de análisis de clases e infraestructura **ACOPLAMIENTO** SE DEFINE COMO EL NIVEL DE INTERDEPENDENCIA ENTRE VARIOS **Cohesion** se define como el grado en que están relacionadas las distintas partes de un componente de software	