



Universidad de Oviedo



Arquitectura del Software

Definiciones

2025-26



Jose Emilio Labra Gayo

Esquema

Definiciones de arquitectura software

¿Qué es arquitectura del software?

Stakeholders

Atributos de calidad

Restricciones

¿Qué es la arquitectura?

Etimológicamente, viene del griego:

Arquitectura = ἀρχιτέκτων

ἀρχι- "jefe"

τέκτων "creador"

Architectura = Proceso y producto de planificar, diseñar y construir edificios u **otras estructuras**



Vitruvio, "De arquitectura"

Escrito entre 30 y 15 antes de Cristo

3 pilares de una buena arquitectura

Utilitas (utilidad):

Ser útil y funcionar bien para las personas que lo van a usar

Firmitas (durabilidad):

Mantenerse de forma robusta y en buena condición

Venustas (elegancia):

Ser agradable a las personas

Lo mismo puede aplicarse a los sistemas de software



¿Qué es arquitectura del software? (1)

Arquitectura [ISO/IEC/IEEE 42010:2011, 3.2]

Conceptos fundamentales o propiedades de un Sistema en su entorno representados por sus elementos, relaciones y los principios de su diseño y evolución

Descripción de arquitectura

Producto de trabajo explícito que expresa una arquitectura de un sistema, normalmente a través de modelos, texto o gráficos

Diseñar una arquitectura (*architecting*)

Proceso de crear una arquitectura

¿Qué es arquitectura del software? (2)

Estructuras fundamentales de un sistema...

...que comprenden:

Elementos de software

Relaciones entre ellos

Propiedades de ambos

Arquitectura vs diseño

Arquitectura se enfoca en estructura de alto nivel de un sistema de software

Decisiones de diseño principales del sistema

Si hay que cambiarlas ⇒ Coste elevado

*"Toda arquitectura conlleva diseño pero
no todo el diseño es arquitectura"*

G. Booch

Arquitectura de edificios vs del software

Algunas similitudes

Sistemas complejos

Desarrollados por equipos/organizaciones

Utilizados por personas

Ambos emplean estilos, patrones, tácticas, ...

Y están afectados por tendencias

Arquitectura de edificios vs del software

Algunas diferencias

Arquitectura de edificios

Entornos y contexto más estables

Servicio/producto físico

Límites físicos

Gran tradición e historia

Sirve para inspiración



Arquitectura del Software

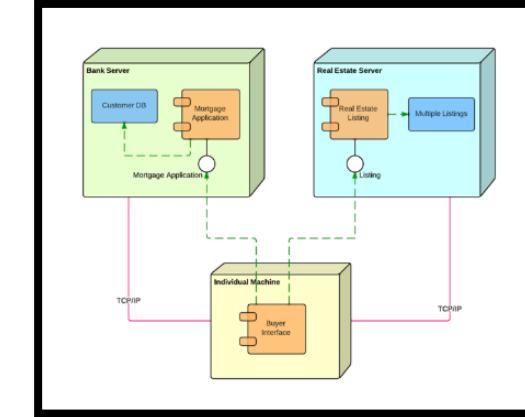
Múltiples cambios en contexto

Servicio/producto virtual

No suele haber límites físicos

Disciplina relativamente nueva

Podemos aprender mucho de otras



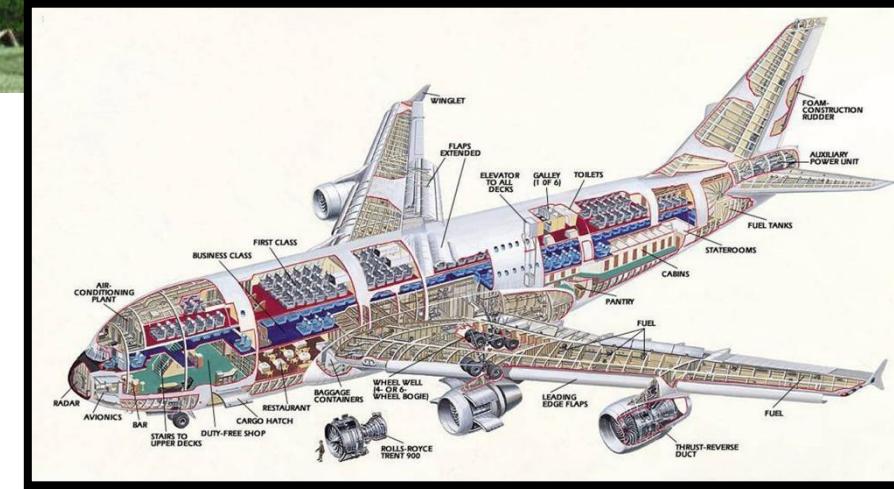
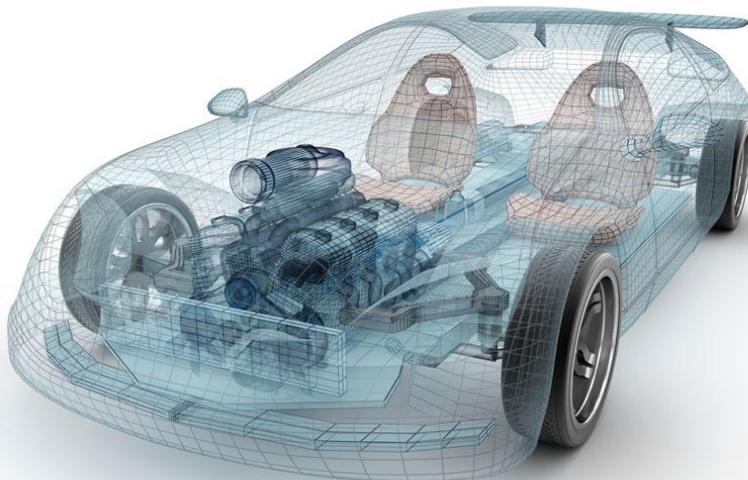
Otras disciplinas similares

Ingeniería Civil

Ingeniería mecánica

Aeronáutica

...



Otras arquitecturas

Arquitectura de negocios

Arquitectura empresarial

Arquitectura de sistemas

Arquitectura de la información

Arquitectura de datos

...

Cosas comunes a todas: Estructura y visión

Beneficios arquitectura del software

Proporcionar visión y mapa a seguir por equipo

Liderazgo y coordinación técnica

Responder cuestiones sobre decisiones significativas

Atributos de calidad y otras preocupaciones

Marco para **identificar y mitigar riesgo**

Gestión consistente de estándares y código

Fundamentos firmes del producto a desarrollar

Estructura para **comunicar** la solución

Diferentes niveles de abstracción y audiencias

Retos arquitectura del software

Arquitectos en la torre de marfil

Falta de comunicación

Centralización de todas las decisiones

Arquitecto = cuello de botella

Tomar demasiadas decisiones

Retrasar ciertas decisiones puede ser mejor que deshacerlas

Big Design Up-Front

Demasiados diagramas y documentos innecesarios

Retardos debidos al proceso de arquitectura

Arquitectura de software ágil

Arquitectura que puede reaccionar a cambios en entorno

Adaptación a cambios continuos

También conocidas como arquitecturas evolutivas

Buena arquitectura permite agilidad

Mejor comprensión de decisiones y soluciones de compromiso

Anti-patrón común:

Adoptar técnicas desarrollo de software ágil que crean arquitecturas que no son ágiles

Causado por demasiado enfoque en funcionalidad

Leyes de la arquitectura del software

1er ley:

Todo en arquitectura del software es una solución de compromiso



Corolario 1:

Si un arquitecto piensa que no ha encontrado algo que no sea una solución de compromiso, lo más seguro es que no ha identificado la solución de compromiso todavía

Corolario 2:

Todas las decisiones significativas tienen desventajas

2^a ley:

El Porqué es más importante que el cómo

Documentar decisiones tomadas

Cuestionar todo

Proceso de diseñar una arquitectura

Dominio del problema

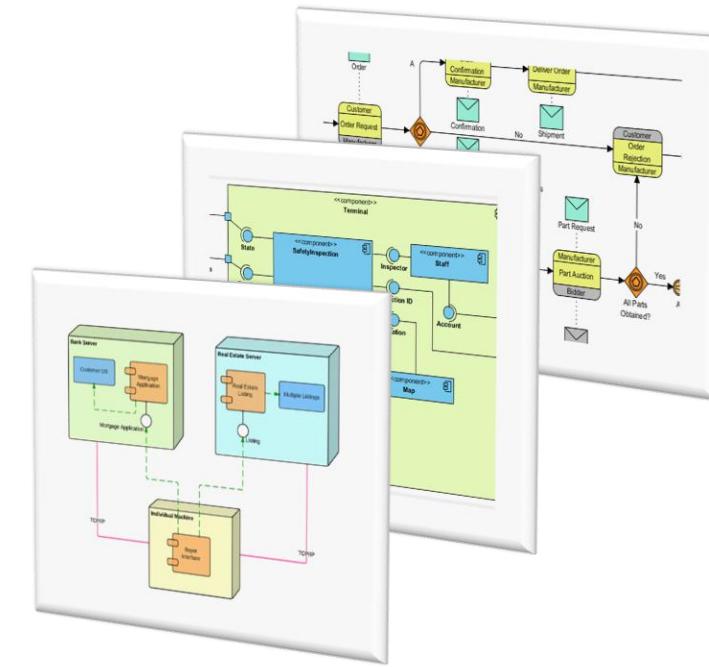
- Objetivos de Diseño
- Requisitos Funcionales
- Atributos de Calidad
- Restricciones
- Preocupaciones (Concerns)



Arquitecto

Actividad de diseño

Dominio de la solución



Diseño de La arquitectura (salida)

Entradas

Motivaciones proceso de arquitectura

Entradas

Objetivos de diseño

Requisitos funcionales

Atributos de calidad

Restricciones

Preocupaciones

Objetivos de diseño

Aclarar **porqué** se diseña un sistema

Ejemplos:

Propuesta pre-venta: diseño rápido de una solución inicial para obtener una estimación

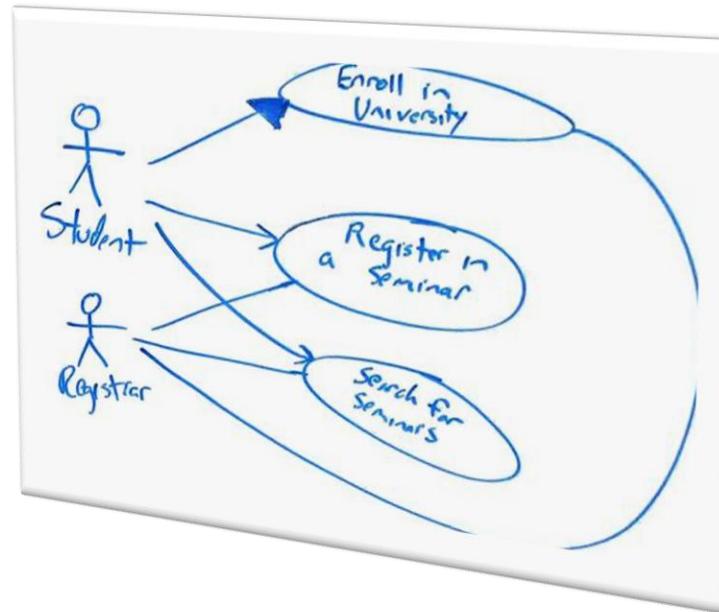
Sistema a medida con un tiempo y coste establecido que no puede variar mucho una vez enviado

Incremento Nuevo ó versión de un sistema que está continuamente evolucionando

Requisitos funcionales

Funcionalidad que debe soportar los objetivos de negocio
Lista de requisitos como casos de uso o historias de usuario

Casos de uso



Historias de usuario



Atributos de calidad

Características medibles de interés para usuarios o desarrolladores

También conocidos como requisitos no-funcionales

Rendimiento, disponibilidad, modificabilidad, testabilidad,...

También conocidos como -idades (-ities en inglés)

Pueden especificarse mediante escenarios

Técnica estímulo-respuesta

ISO 25010: lista de algunos requisitos no funcionales

Lista: https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Atributos de calidad

Los atributos de calidad determinan la mayoría de las decisiones de diseño en arquitectura

Si la única preocupación fuese la funcionalidad, un sistema monolítico sería suficiente

Sin embargo, es habitual ver:

Estructuras redundantes para fiabilidad

Estructuras concurrentes para rendimiento

Capas para modificabilidad

...

Priorizados por:

El cliente de acuerdo al éxito esperado del sistema

El arquitecto de acuerdo al riesgo técnico

Restricciones

Restricciones del sistema que vienen impuestas

Muy poco software tiene libertad total

Pueden ser técnicas u organizativas

Pueden surgir del cliente o también de la organización de desarrollo

Limitan las alternativas a considerar para decisiones de diseño particulares

Ejemplos:

Marcos de aplicaciones (*frameworks*)

Lenguajes de programación, ...

Normalmente son “*amigos*” del arquitecto

Preocupaciones

Decisiones de diseño que deben tomarse aunque no estén enunciadas explícitamente en los objetivos o requisitos

Ejemplos:

Crear una estructura física o lógica consistente

Validar campos de entrada

Gestión de excepciones y logging

Migración de datos y backup

Organización del código fuente

...

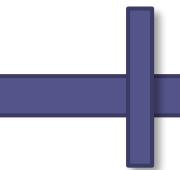
Creatividad vs método

Creatividad

Divertido
Arriesgado
Puede ofrecer soluciones nuevas
Puede ser innecesario

Método

Eficiente en terrenos familiares
Resultado predecible
No siempre es lo mejor
Técnicas de calidad contrastada



Arquitecto



Tipos de sistemas

Sistemas *greenfield* en dominios nuevos

Ejemplo Google, WhatsApp

Dominios innovadores poco conocidos



Sistemas *greenfield* en dominios maduros

Ejemplo: aplicaciones empresariales tradicionales, aplicaciones móviles, ...

Dominios conocidos, menos innovadores



Dominios *brownfield*

Cambios a sistemas existentes

Greenfield: no urbanizado

Brownfield: antigua zona industrial



Arquitecto del software

La disciplina evoluciona

Arquitecto debe conocer:

Avances en técnicas de construcción

Estilos y patrones

Mejor herramienta = experiencia (*no silver bullet*)

Experiencia propia

Experiencia de la comunidad



Arquitecto



Papel del arquitecto de software

