



Universidad de Oviedo



Técnicas arquitectura del software



Curso 2020/2021

Jose Emilio Labra Gayo

Arquitecto del software

La disciplina evoluciona

Arquitecto debe conocer:

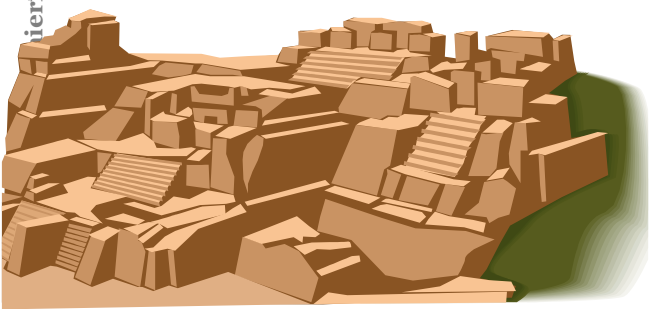
Avances en técnicas de construcción

Estilos y patrones

Mejor herramienta = experiencia (*no silver bullet*)

Experiencia propia

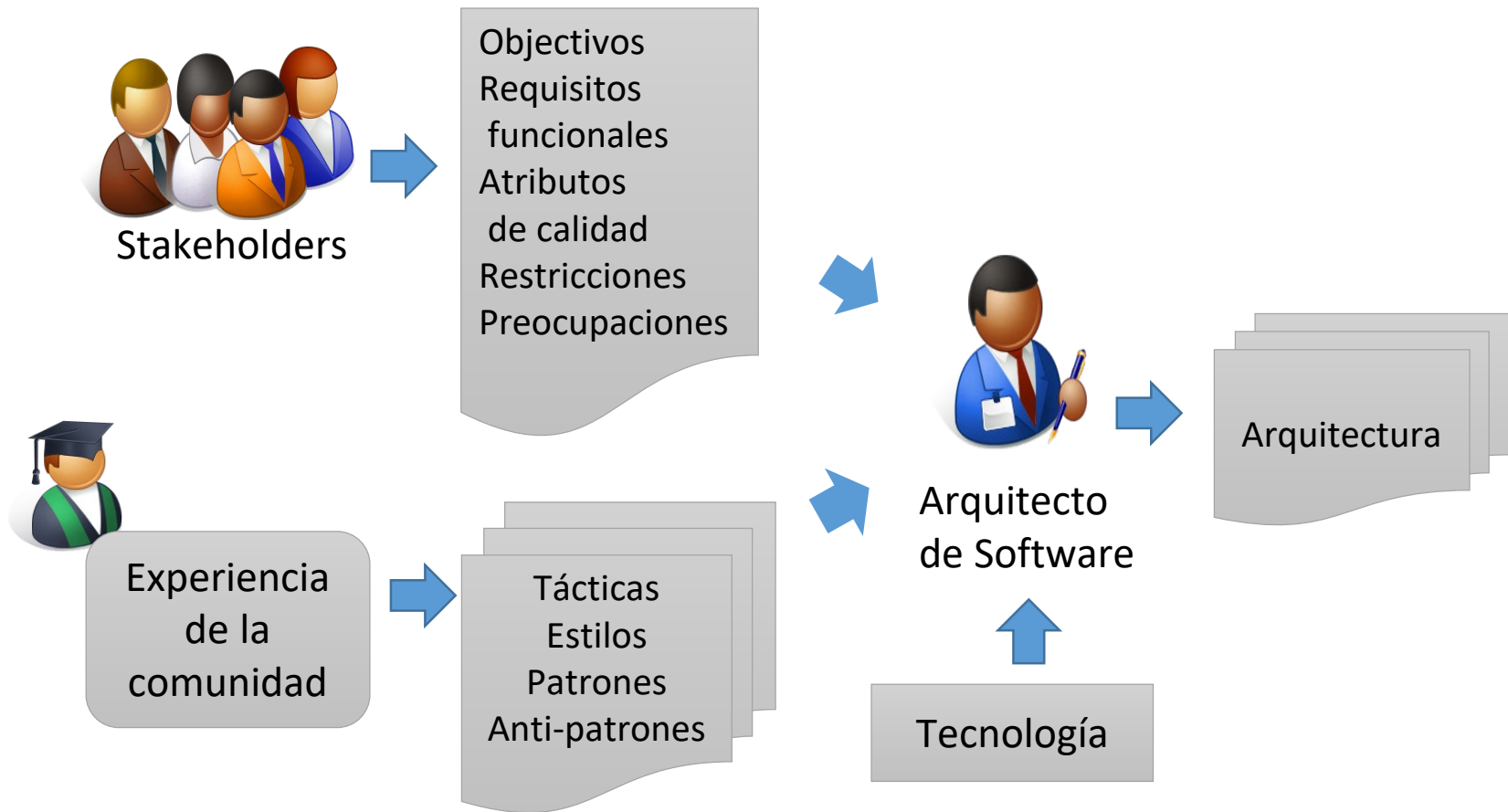
Experiencia de la comunidad



Arquitecto



Papel del arquitecto de software



Tácticas

Técnicas de diseño para alcanzar una respuesta a algunos atributos de calidad

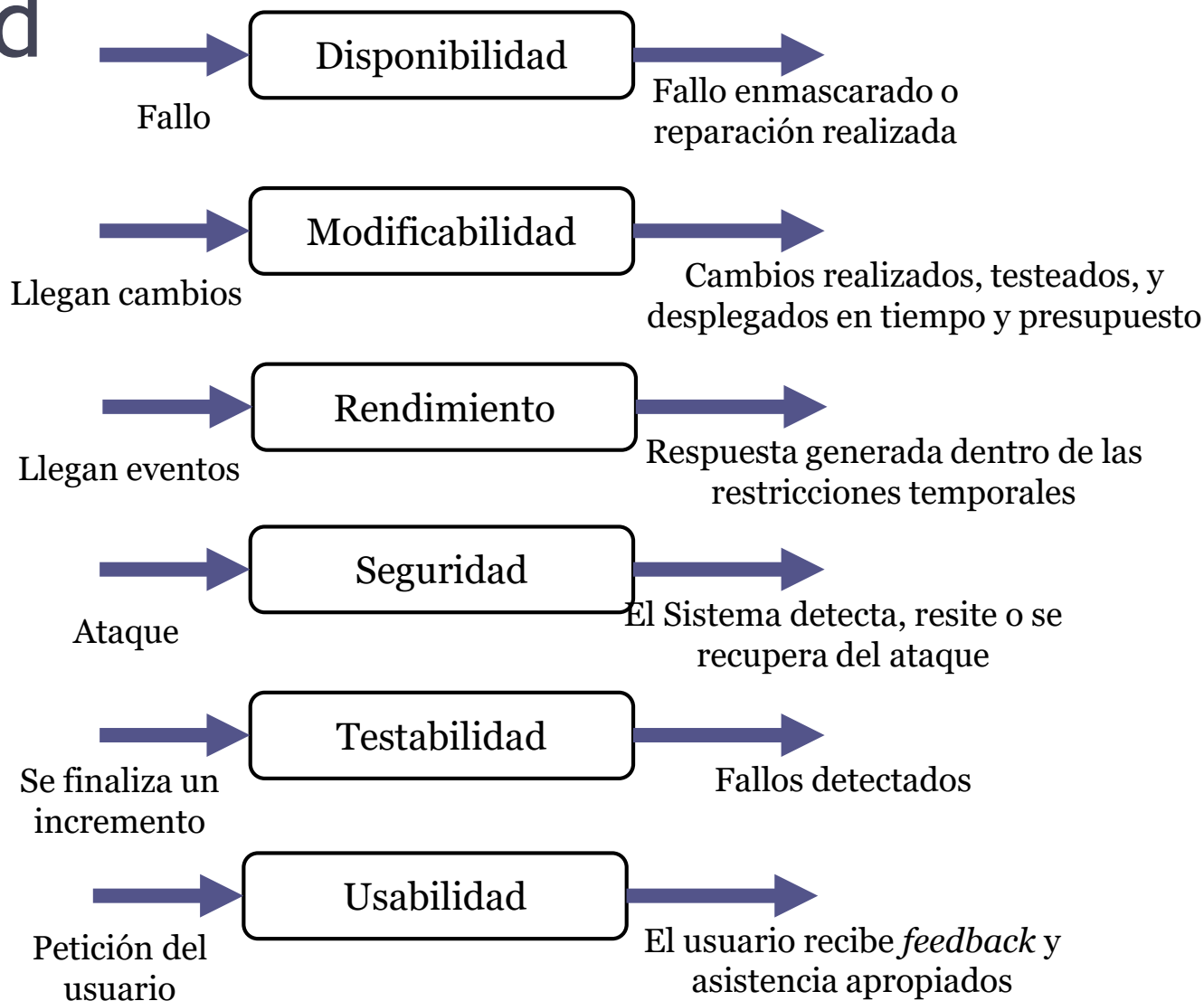
Las tácticas se enfocan en la respuesta a un Atributo de calidad

Pueden chocar con otros atributos de calidad

Las tácticas intentan controlar respuestas a estímulos



Tácticas dependen del atributo de calidad



¿Dónde podemos encontrar tácticas?

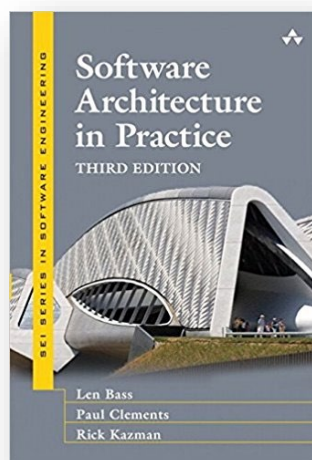
Propia experiencia del arquitecto

Experiencia documentada de la comunidad

Libros, conferencias, blogs,...

Las tácticas evolucionan con tiempo y tendencias

Libro "Software architecture in practice" contiene una lista de varias tácticas



<http://www.ece.ubc.ca/~matei/EECE417/BASS/ch05lev1sec1.html>
<https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm>

Estilos arquitectónicos

Definen la forma general del sistema

Contienen:

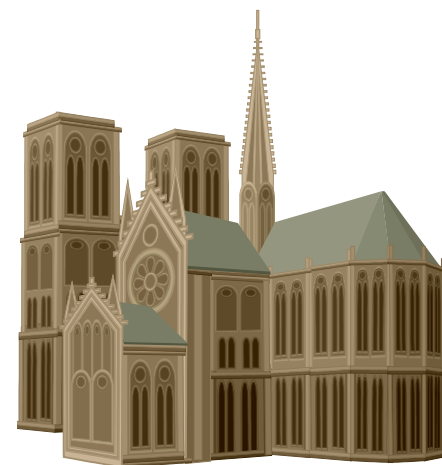
Elementos: Componentes que contienen funcionalidad

Relaciones: Relaciones entre los elementos

Restricciones: Limitan la integración entre elementos

Lista de atributos:

Ventajas/desventajas de un estilo



¿Existen estilos puros?

Estilos puros = idealización

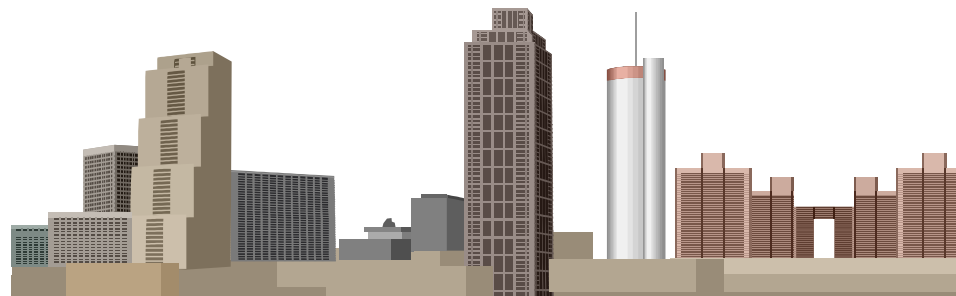
En práctica, los estilos puros se dan pocas veces
Normalmente, los sistemas se desvían de estilos puros...

...o combinan varios estilos arquitectónicos

Es importante comprender los estilos puros para:

Comprender pros/cons de un estilo

Valorar las consecuencias de desviarse del estilo



Patrón arquitectónico

Solución general y reutilizable a algún problema recurrente que aparece en un contexto

Parámetro importante: **problema**

3 tipos:

Estructurales: Tiempo de construcción

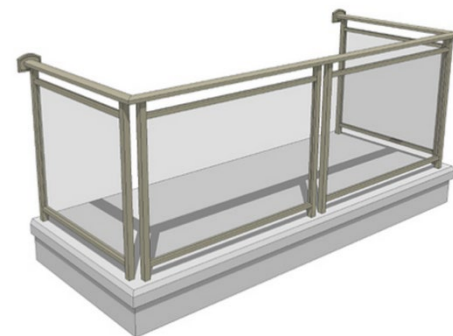
Ejemplo: Layers

Runtime (comportamiento)

Ejemplo: Pipes & filters

Despliegue

Ejemplo: Cluster de balanceo de carga



Patrón vs Estilo

Patrón = solución a un problema

Estilo = genérico

No tiene que estar asociado a un problema

Estilo define la arquitectura general de una aplicación

Normalmente, una aplicación tiene un estilo

...pero puede tener varios patrones

Los patrones aparecen en escalas diferentes

Alto nivel (patrones arquitectónicos)

Diseño (patrones de diseño)

Implementación (idiomas)

. . .

Patrón vs Estilo

Los estilos, en general, son independientes entre sí

Un patrón puede relacionarse con otros patrones

Un patrón puede estar compuesto de varios patrones

Pueden crearse interacciones entre patrones

Lenguajes y catálogos de patrones

Catálogo de patrones

Un conjunto de patrones sobre un asunto

No tiene porqué ser exhaustivo

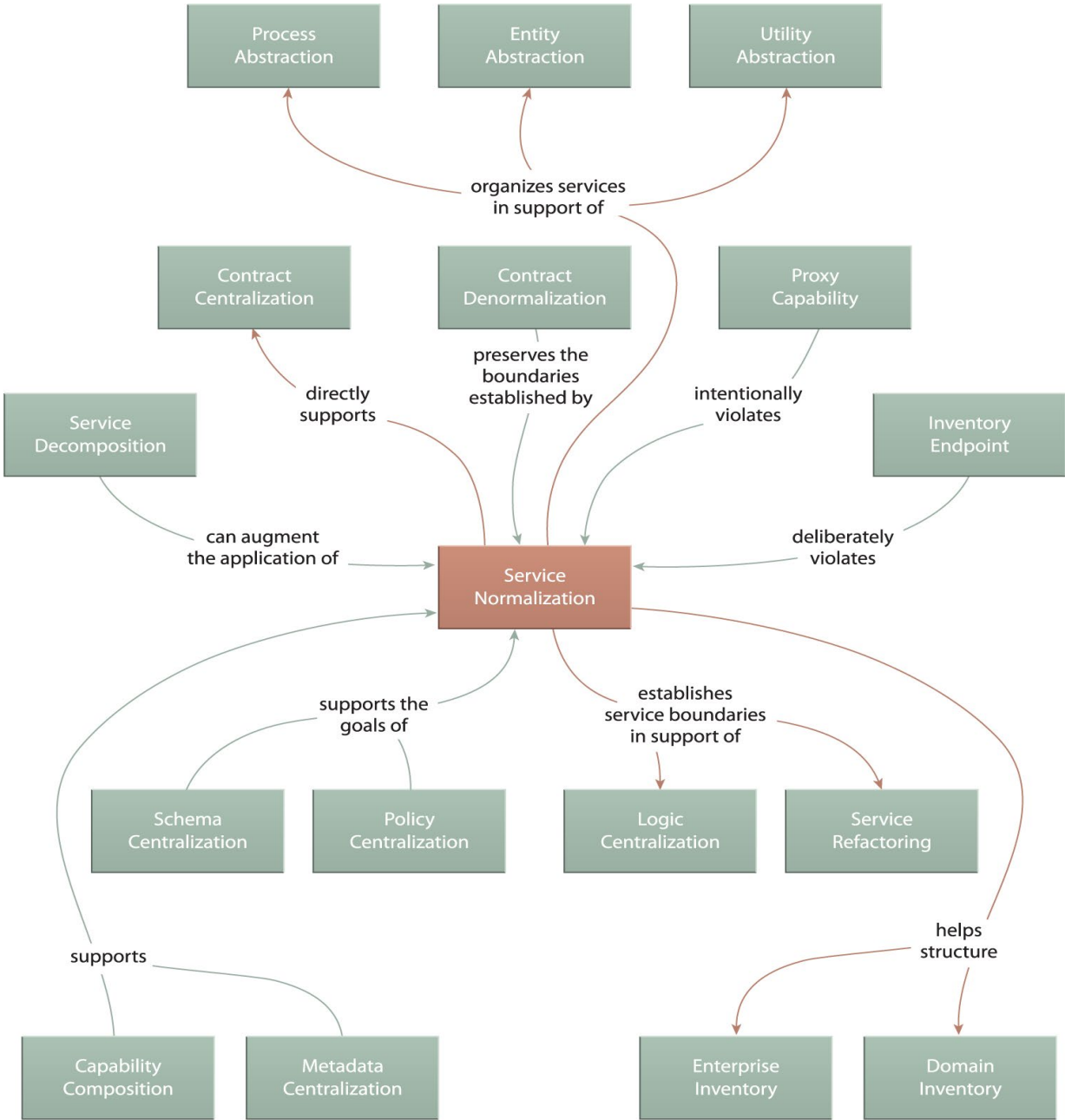
Lenguaje de patrones

Un catálogo de patrones completo sobre un tema

Objetivo: documentar todas las posibilidades

Normalmente incluyen relaciones entre patrones

Mapa gráfico



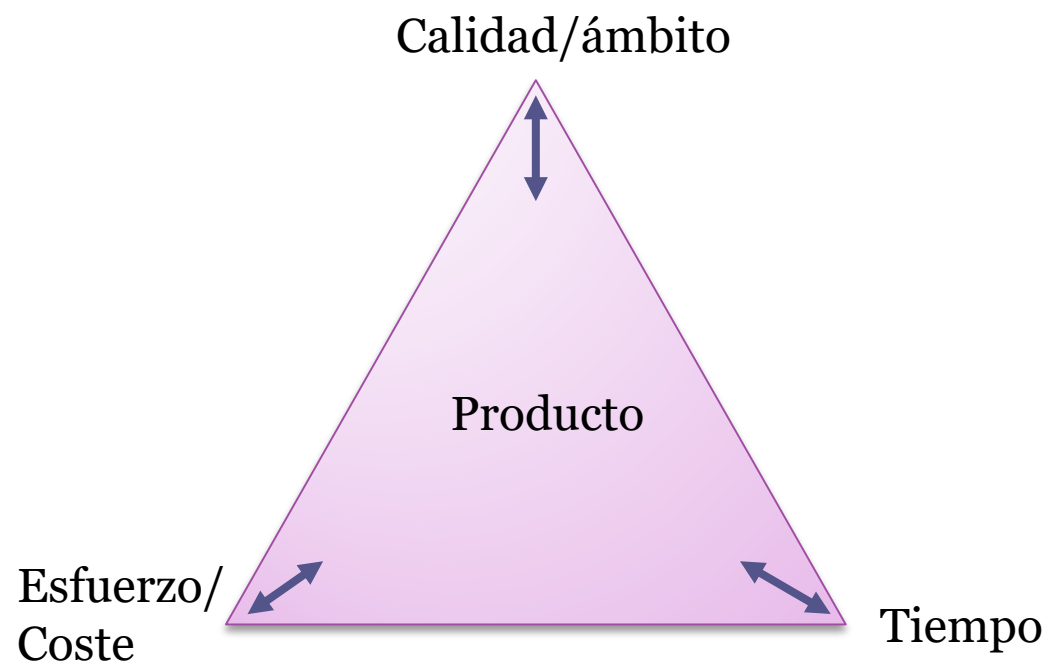
Ejemplo de lenguaje de patrones
Source: "SOA with REST" book

Construir vs reutilizar

En algunos dominios, reutilizar arquitecturas existentes puede ser más eficiente

Arquitecturas de referencia

Componentes desarrollados externamente

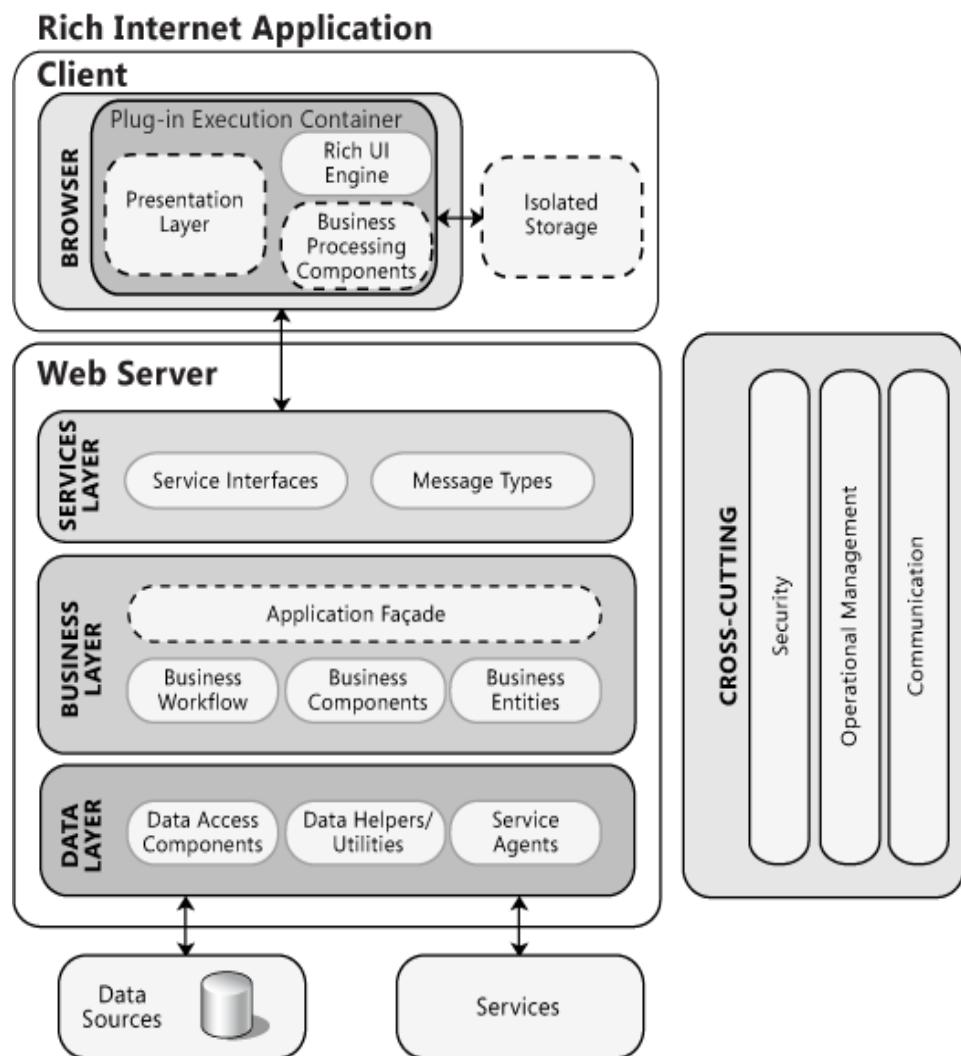


Arquitecturas de referencia

Planos que proporcionan una estructura general para ciertos tipos de aplicaciones

Pueden contener varios patrones

Pueden ser un estándar *de-facto* en algunos dominios



Arquitecturas de software de dominios específicos

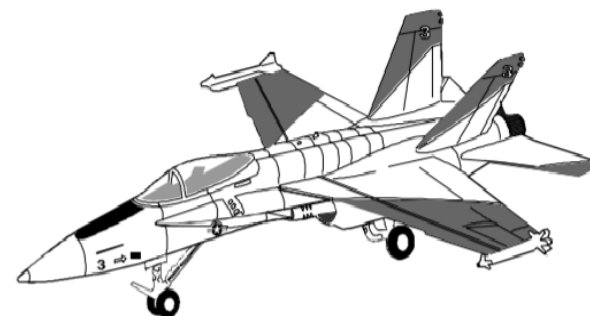
Combinación de:

- Arquitectura de referencia para un dominio
- Librería de componentes para dicha arquitectura
- Método para elegir y configurar components para trabajar dentro de una instancia de dicha arquitectura de referencia

Especializados para un dominio concreto

Ejemplos:

ADAGE, MetaH



Componentes desarrollados externamente

Pilas tecnológicas o familias

MEAN (Mongo, Express, Angular, Node), **LAMP** (Linux, Apache, MySQL, PHP), ...

Productos

COTS: Commercial Off The Shelf

FOSS: Free Open Source Software

¡Cuidado con las licencias!

Marcos de aplicación

Componentes de software reutilizables

Plataformas

Proporcionan infraestructura completa para construir y ejecutar aplicaciones

Example: JEE, Google Cloud

Fin