



Universidad de Oviedo



Alcanzando la arquitectura del software



Curso 2022/2023

Jose Emilio Labra Gayo

Alcanzando arquitectura del software

Metodologías

ADD

Basado en riesgos

Toma de decisiones

Incidencias arquitectónicas

Riesgos, desconocidos, problemas, etc.

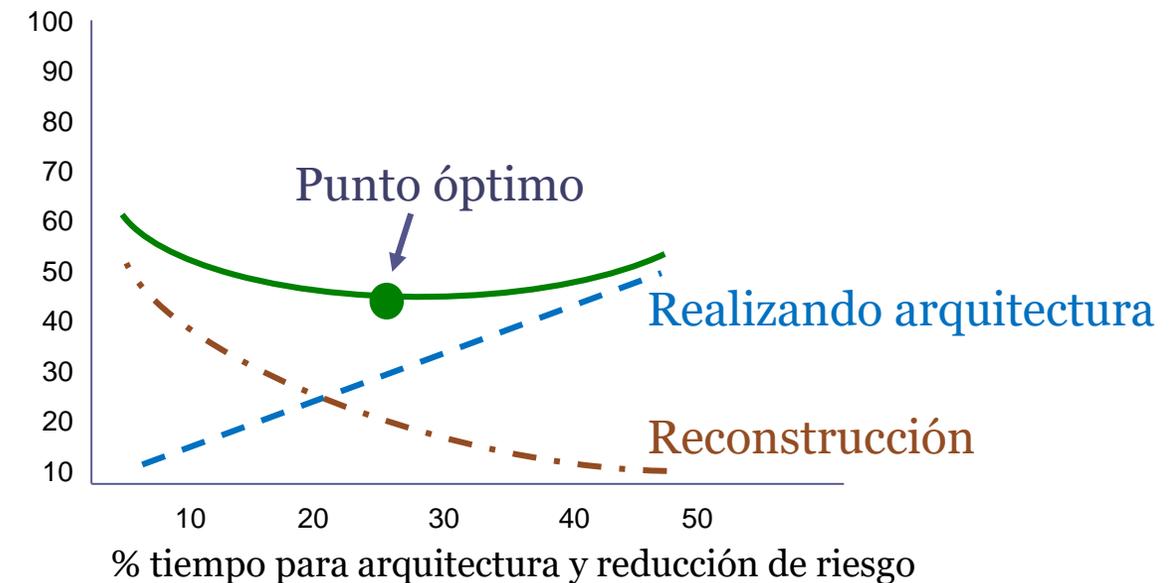
Evaluación de arquitecturas

¿Cuánta arquitectura?

Punto óptimo entre demasiada arquitectura y demasiada reconstrucción

$$\text{Tiempo Total proyecto} = \begin{cases} \text{Tiempo desarrollo} + \\ \text{Tiempo arquitectura} + \\ \text{Tiempo reconstrucción} \end{cases}$$

% tiempo añadido
al plan inicial



ADD - Attribute Driven Design

ADD: Attribute-driven design

Define una arquitectura del software basada en ACs

Proceso de descomposición recursivo

En cada etapa, se eligen patrones y tácticas para satisfacer un conjunto de ACs

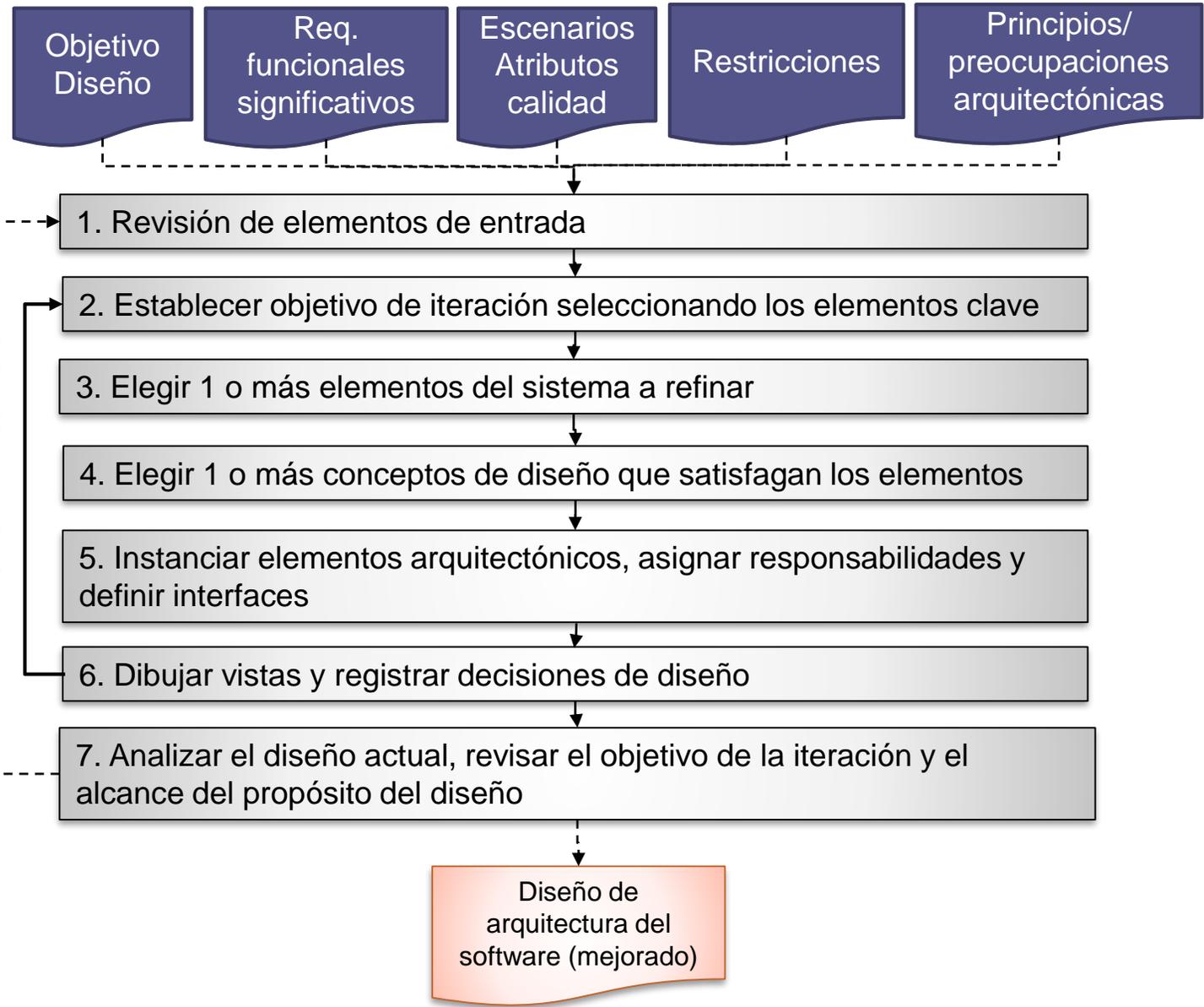
Entrada

- Requisitos AC
- Restricciones
- Req. funcionales significativos para la arquitectura

Salida

- Primeros niveles de descomposición modular
- Varias vistas del Sistema según se consideren apropiadas
- Conjunto de elementos con funcionalidad asignada e interacciones entre los elementos

ADD 3.0



Clave

- Elemento de entrada
- Diseño de arquitectura
- Paso del método
- Precedencia
- Flujo de artefactos

Desde la ronda de iteraciones anterior o de Sistema existente (desarrollos *brownfield*)

Iterar si es necesario

Decisiones arquitectónicas

Decisiones significativas desde el punto de vista de la arquitectura del software

Normalmente, decisiones que afectan a:

Atributos de calidad

Estructura

Dependencias

Interfaces

Técnicas de construcción

Anti-patrones decisiones arquitectónicas

Parálisis por análisis

No tomar decisiones

Miedo a tomar decisiones equivocadas

Consejo: Evaluar decisión con equipo (esperar cambios)

Esperar hasta último momento responsable (pero no más)

Atrapado en el tiempo

La gente no sabe porqué se hizo una decisión

Se discute una y otra vez

Consejo: siempre incluir justificación

Decisiones basadas en E-mail

La gente pierde, olvida o ni siquiera saben de una decisión

Consejo: Registro de decisiones arquitectónicas

Registro decisiones arquitectónicas

Toda decisión de diseño es *suficientemente buena* pero pocas veces óptima

Es necesario registrar la justificación y los riesgos

Cosas a registrar:

¿Cuál es la evidencia que justifica la decisión?

¿Quién toma la decisión?

¿Porqué se han tomado ciertos atajos?

¿Porqué se realizaron ciertos compromisos?

¿Qué suposiciones se han realizado?

| Clave | Decisión de diseño | Justificación y suposiciones |
|-------|---|--|
| DA-1 | Introducir concurrencia (táctica) en TimeServerConnector y FaultDetectionService | La concurrencia debería ser introducida para ser capaz de recibir y procesar varios eventos simultáneamente |
| DA-2 | Utilizar un patron de mensajería mediante una cola de mensajes en la capa de comunicaciones | Aunque el uso de una cola de mensajes puede ir en contra del rendimiento impuesto por el escenario, será útil para dar soporte al escenario QA-3 |
| ... | ... | ... |

Registros decisiones arquitectónicas

Plantillas: <https://adr.github.io/>

Estructura básica:

Título

Breve título descriptivo

Estado

Propuesto, aceptado, reemplazado

Contexto

Qué es lo que fuerza a tomar la decisión

Incluir alternativas

Decisión

Decisión y justificación correspondiente

Consecuencias

Impacto esperado de la decisión



Los registros deberían ser inmutables: si aparece un cambio, crear un nuevo registro que reemplaza el anterior.

Para borradores, puede ser útil utilizar RFCs (*Request for comments*)

Incidencias arquitectónicas

Incidencias arquitectónicas

Riesgos

Desconocidos

Problemas

Deuda técnica

Diferencias de comprensión

Erosión/Ir a la deriva

Evolución del contexto

Riesgos

Riesgo = algo malo que podría ocurrir pero que todavía no ha ocurrido

Riesgos deberían ser identificados y registrados

Riesgos pueden aparecer como parte de escenarios de AC

Riesgos pueden ser mitigados o aceptados

Si es posible, identificar tareas de mitigación

Tabla valoración riesgo

Valorar riesgos en 2 dimensiones:

Impacto del riesgo

Probabilidad de ocurrencia del riesgo

Matriz riesgo 3x3

Simplificar valores: bajo (1), medio (2), alto (3)

| | | Probabilidad | | |
|---------|-----------|--------------|-----------|----------|
| | | Bajo (1) | Medio (2) | Alto (3) |
| Impacto | Bajo (1) | 1 | 2 | 3 |
| | Medio (2) | 2 | 4 | 6 |
| | Alto (3) | 3 | 6 | 9 |

Matriz riesgo 3x3

| Criterio riesgo \ Area | Registro cliente | Realización Petición |
|------------------------|------------------|----------------------|
| Escalabilidad | 2 | 1 |
| Disponibilidad | 3 | 2 |
| Rendimiento | 4 | 3 |
| Seguridad | 6 | 1 |
| Integridad datos | 9 | 1 |

Tabla de valoración de riesgo

Risk storming: ejercicio recomendado para evaluar riesgos colaborativamente

Desconocidos (unknowns)

Algunas veces no tenemos suficiente información sobre cómo una arquitectura puede satisfacer los requisitos

Requisitos poco especificados

Presuposiciones implícitas

Requisitos cambiantes

...

Las evaluaciones arquitectónicas pueden ayudar a convertir los desconocidos en riesgos



Problemas

Problemas = cosas malas que ya han pasado

Tomar decisiones que no funcionan como se espera

Cambios en el contexto

Una decisión que era buena puede dejar de serlo

Los problemas pueden arreglarse o ser aceptados

Los problemas que no se arreglan pueden generar una **deuda técnica**
(*technical debt*)



Deuda técnica

Deuda adquirida cuando consciente o inconscientemente se toman decisiones de diseño equivocadas

Si se pagan los plazos la deuda es devuelta y no se crean más problemas

En caso contrario, se incurre en una penalización (interés)

Si no se puede pagar durante mucho tiempo, la deuda puede ser tan grande que se declara bancarrota

En términos software, significaría producto abandonado

Varios tipos:

Deuda de código: Estilo de código malo o inconsistente

Deuda diseño: *Malos olores* de diseño

Deuda de prueba: Falta de pruebas, poca cobertura,...

Deuda documentación: Aspectos importantes sin documentación, documentación no actualizada,...



Diferencias de comprensión

Aparecen cuando lo que piensan los stakeholders sobre la arquitectura no encaja con el diseño

Las arquitecturas evolucionan rápidamente y las diferencias aparecen rápidamente y sin previo aviso

Diferencias pueden afrontarse con formación

Presentando la arquitectura a los *stakeholders*

Realizando preguntas a los *stakeholders*



Deterioro arquitectónico

Diferencia entre la arquitectura diseñada y la arquitectura del sistema construido

El sistema implementado casi nunca se parece al sistema que el arquitecto se imagina

Sin vigilancia, la arquitectura puede ir derivando y alejándose del diseño planificado hasta que un día apenas se parezcan

Código arquitectónicamente evidente puede mitigar esta diferencia

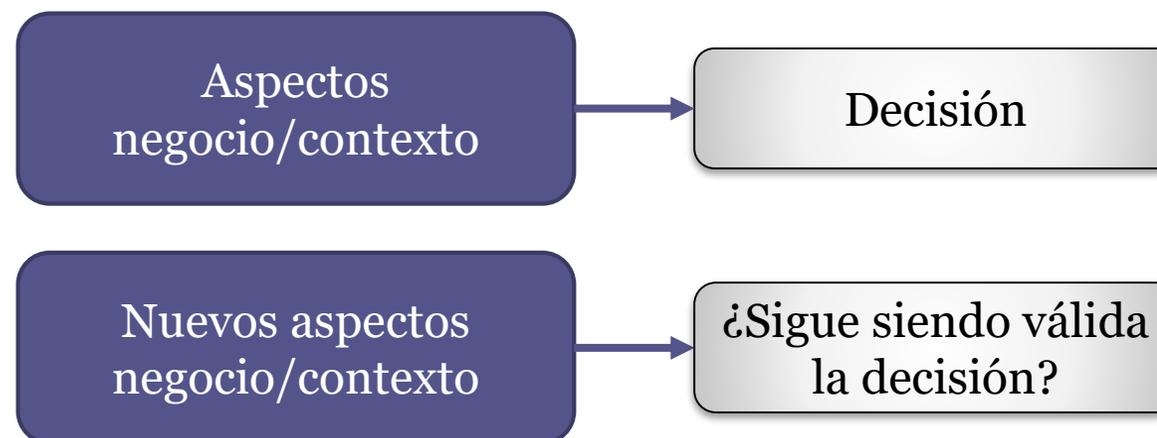


Evolución del contexto

Ocurre cuando aspectos claves del contexto cambian después de haber tomado una decisión de diseño

Es necesario revisar continuamente los requisitos

Arquitecturas evolutivas



Evaluación de arquitecturas

Evaluación de la arquitectura

ATAM (Architecture Trade-off Analysis Method)

Método para evaluar arquitecturas del software

Versión simplificada:

- Presenta aspectos clave de negocio
- Presentar arquitectura
- Identificar enfoques de la arquitectura
- Generar árbol de utilidad de Atributos de Calidad
- Analizar enfoques arquitectónicos
- Presenta resultados

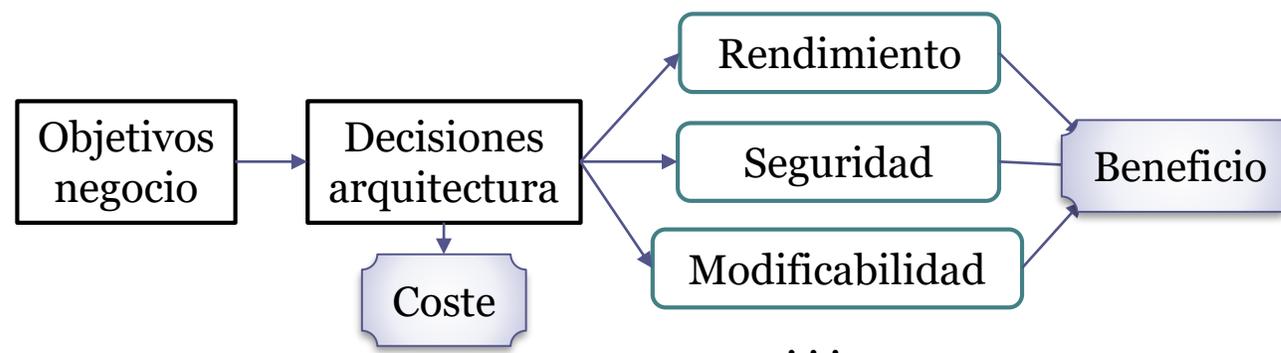


Cost Benefit Analysis Method (CBAM)

1. Elegir escenarios y estrategias arquitectónicas
2. Valorar beneficios para atributos de calidad
3. Cuantificar beneficios de estrategias arquitectónicas
4. Cuantificar costes e implicaciones de las estrategias
5. Calcular la deseabilidad de cada opción

$$VFC \text{ (Valor para coste)} = \frac{\text{Beneficio}}{\text{Coste}}$$

6. Tomar decisiones de diseño arquitectónico



Fin