

Arquitectura de Software

Decidir, documentar y evaluar

Vistas, C4 y arc42 · ADR · Diseño guiado por atributos · ATAM

Gerson Garrido

Semana 7

UNIVERSIDAD NACIONAL DE INGENIERÍA

Agenda

Sesión A: decidir y documentar

1. Para quién se documenta
2. Vistas
3. C4: cuatro niveles de zoom
4. arc42: el documento completo
5. ADR: una decisión, un documento

Sesión B: diseñar y evaluar

6. Diseñar con un método: ADD
7. Evaluar antes de construir
8. ATAM
9. Una versión liviana · Risk storming

Lectura

A: *Software Architecture in Practice*, cap. 22; Nygard, *Documenting Architecture Decisions* (2011); Brown, *C4: diagramas*.
B: *Software Architecture in Practice*, cap. 20 y 21; Kazman, Klein y Clements, *ATAM: Method for Architecture Evaluation* (SEI, 2000), secciones 3 y 7.

Seis semanas de decisiones, ¿y dónde están?

- Semanas 3 y 4: límites entre módulos y reglas de dependencia.
- Semana 5: un estilo, comparado con escenarios.
- Semana 6: tácticas, medidas antes y después.

Cada una tuvo alternativas, una razón y un costo. Hoy casi todo eso está en la memoria de quienes estuvieron.

Esta semana: tres cosas que se hacen con las decisiones

- **Documentarlas**: vistas, C4, arc42, ADR.
- **Tomarlas con un método**: ADD.
- **Evaluarlas**: ATAM.

Las tres dependen de los escenarios de la semana 2.

SECCIÓN 01

Para quién se documenta

01

Cuatro lectores, cuatro preguntas

Lector	Qué quiere saber	Qué necesita ver
Quien construye	Qué me toca y qué no puedo tocar	Módulos, interfaces y reglas de dependencia
Quien llega en un año	Por qué está así	Las decisiones y lo que se descartó
Quien evalúa	Si cumple sus escenarios	Qué corre y qué pasa si falla
Quien opera	Qué corre y dónde	Procesos, máquinas y dependencias

Error común: el diagrama único

Carpetas, procesos, tablas y máquinas en una figura: no responde ninguna de las cuatro preguntas.

Vistas

Vista *(view)*

La representación de una estructura, hecha para un lector. La estructura está en el sistema; la vista, en el documento.

Tipo de vista	Responde	Lector típico
De módulos	¿Qué toca este cambio?	Quien construye
De componentes y conectores	¿Cuánto tarda? ¿Qué pasa si se cae?	Quien evalúa
De asignación	¿Qué corre en qué máquina?	Quien opera

Qué vistas elegir depende de qué stakeholders hay y qué pregunta trae cada uno.

Lo que lleva una vista

1. El **diagrama principal**: elementos y relaciones.
2. El **catálogo de elementos**: qué es cada caja, su interfaz y su comportamiento.
3. La **leyenda**: qué significa cada forma y cada flecha.
4. La **justificación**: por qué es así.

Sin leyenda no es documentación

Son cajas y flechas con un significado privado.

En el formato del curso

La leyenda está en las columnas: *Tipo*, *Qué viaja* y *Protocolo*.

SECCIÓN 02

El sistema de la semana: Yupana

02

Yupana: el juez de tareas de programación de una facultad

- Compila y ejecuta cada solución contra casos ocultos, en un **sandbox**.
- Devuelve un **veredicto**.
- 6 cursos, 2 personas, una máquina de 4 núcleos.

Atributo	Escenario	Notas
Rendimiento	E1. 600 envíos en los 30 min antes de un cierre: confirmación en 1 s, veredicto en 2 min (p95)	(A, A)
Seguridad	E2. Un programa intenta leer fuera de su carpeta y usar la red: 0 bytes, 0 conexiones	(A, A)
Disponibilidad	E3. Un worker muere a mitad de una calificación: veredicto en 5 min, 0 envíos perdidos	(A, A)
Modificabilidad	E4. Agregar un lenguaje: 3 días-persona, 1 módulo	(M, B)
Disponibilidad	E5. El proveedor de identidad no responde 15 min	(A, M)

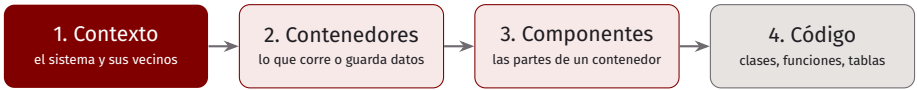
El laboratorio evalúa el Tinkuy de cada pareja; por eso los ejemplos de hoy son de otro sistema.

SECCIÓN 03

C4: cuatro niveles de zoom

03

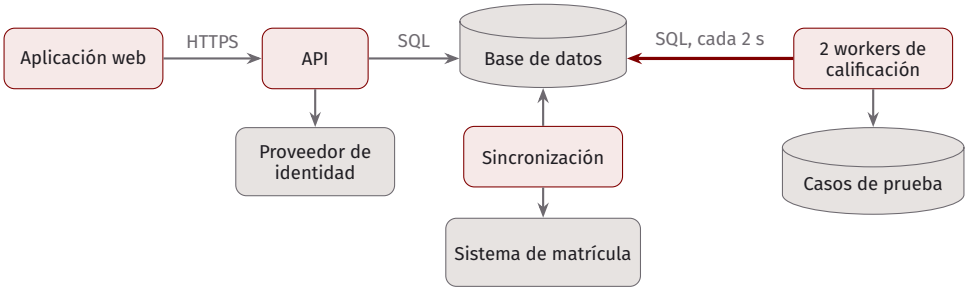
C4: el mismo sistema, cuatro niveles de zoom



Nivel	Abre	Quién lo lee	¿Se dibuja?
1. Contexto	Nada	Todos	Siempre
2. Contenedores	El sistema	Quien construye, evalúa y opera	Siempre
3. Componentes	Un contenedor	Quien construye	Si aporta
4. Código	Un componente	Quien construye	Casi nunca; se genera

La última columna es la recomendación del propio modelo (c4model.com/diagrams).

Yupana por dentro: los contenedores



Lo que el diagrama deja ver

La API y los workers no se hablan: los une la base de datos. Es una decisión, y tiene su ADR.

El tercer nivel: el diagrama de componentes

Componente

Un grupo de funcionalidad detrás de una interfaz, dentro de un contenedor. **No se despliega por separado.** En el curso, casi siempre un módulo de `src/`.

- Abre un solo contenedor.
- Alrededor, solo los vecinos directos.
- Entre componentes: llamadas dentro de un proceso. Hacia afuera: la red.

docs/c4/components.md

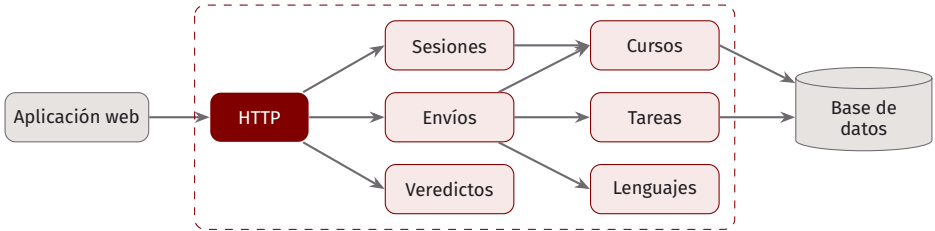
```
# Yupana: diagrama de componentes de la API

- Nivel: componentes
- Sistema: Yupana
- Contenedor: API
```

Error común

Un inventario de archivos o de clases. Con cuarenta cajas ya es el cuarto nivel.

Los componentes de la API de Yupana



- E4 pide agregar un lenguaje tocando un módulo: *Lenguajes* no depende de nadie, y solo lo usa *Envíos*.
- Si ya tienen reglas de `depcruise`, este diagrama y las reglas dicen lo mismo. Compárenlos.

Simplificado. El completo, con sus 17 relaciones: `example/yupana/docs/c4/components.md`.

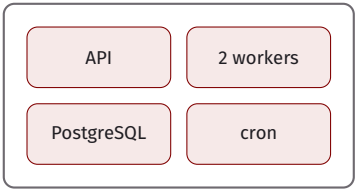
Complementario: el diagrama dinámico

Una interacción, paso a paso. Como texto, es una lista numerada:

1. La aplicación web manda el archivo a la API.
2. *HTTP* pregunta a *Sesiones* quién es, y entrega el envío a *Envíos*.
3. *Envíos* comprueba matrícula, cierre y lenguaje.
4. *Envíos* guarda envío, código y trabajo en una transacción. La API responde 202.
5. Un worker toma el trabajo y lo marca “en calificación”.
6. Lo califica en un sandbox y anota el veredicto.
7. La aplicación web, que pregunta cada 3 s, lo muestra.

> **Una regla útil**
>
> Uno por cada escenario (A, A): es el recorrido que una evaluación va a pedir.

Complementario: el diagrama de despliegue



una máquina virtual, 4 núcleos

- Qué corre dónde: una vista de asignación.
- El diagrama de contenedores no lo dice a propósito: el despliegue cambia de un entorno a otro.
- Aquí hay información que ningún otro diagrama tiene: todo comparte cuatro núcleos.

Volveremos a esta figura en la evaluación.

Las reglas de notación, y por qué como texto

Toda caja dice

Qué es, qué hace y con qué tecnología.

Toda flecha tiene

Una dirección, un verbo y un protocolo.
“Usa” no es un verbo que diga algo.

El formato y el revisor son los de la semana 1; ya aceptaban `Nivel: componentes`.

Diagramas como texto, en el repositorio

- Se **versionan** con el código.
- Se **revisan** en el mismo cambio que los vuelve falsos.
- Se **comparan** entre versiones con `git diff`.

Un PNG no se puede comparar

Un diagrama que nadie revisa envejece sin que nadie lo note.

SECCIÓN 04

arc42: el documento completo

04

arc42: doce secciones

	Sección	Ya la conocen
1	Introducción y objetivos	Stakeholders
2	Restricciones	Semana 1
3	Contexto y alcance	C4, nivel 1
4	Estrategia de solución	nueva
5	Vista de bloques	C4, niveles 2 y 3
6	Vista de ejecución	Dinámico

	Sección	Ya la conocen
7	Vista de despliegue	Despliegue
8	Conceptos transversales	nueva
9	Decisiones de arquitectura	Los ADR
10	Requisitos de calidad	Utility tree
11	Riesgos y deuda técnica	Una evaluación
12	Glosario	

Es un índice, no un formulario

Se llena lo que alguien va a leer; en las demás, una línea que diga por qué todavía no. Se enlaza en vez de copiar.

La sección 4 de Yupana, completa

example/yupana/docs/arc42.md

```
## 4. Estrategia de solución

- Recibir y calificar son procesos distintos, unidos por una tabla de
  trabajos en PostgreSQL (ADR 0003): atiende E1 sin agregar infraestructura.
- Todo código de estudiantes corre dentro de un sandbox sin red y con
  límites, en los workers y nunca en la API: atiende E2.
- El envío y su código se guardan en una transacción antes de confirmar
  (ADR 0002): atiende E3.
```

- Tres líneas, y cada una nombra el escenario que atiende.
- **Trade-off**: un template da un índice conocido, e invita a llenar por llenar. La alternativa es un `README` de arquitectura con la forma que el equipo quiera.
- El proyecto del curso pide los diagramas y los ADR, no un documento arc42.

SECCIÓN 05

ADR: una decisión, un documento

05

Lo que más rápido se pierde es el porqué

Alguien llega y encuentra una decisión que no entiende. Tiene dos opciones *(Nygard, 2011)*:

Aceptarla a ciegas

Mal, si el contexto cambió. El equipo termina con miedo de tocar cualquier cosa.

Cambiarla a ciegas

Mal, si sostenía algo que nadie había probado todavía.

ADR *(architecture decision record)*

Un documento corto por cada decisión significativa para la arquitectura, numerado y guardado en el repositorio.

El formato de Nygard

Título	Una frase corta, con el número
Contexto	Las fuerzas en juego, como hechos
Decisión	En voz activa: “Vamos a…”
Estado	Propuesto, aceptado; obsoleto o reemplazado, con la referencia
Consecuencias	Todas: buenas, malas y neutras

- Se numeran; los números no se reutilizan.
- Una o dos páginas, en frases completas.
- Como una conversación con quien llegue después.
- Las consecuencias de un ADR suelen ser el contexto del siguiente.

El curso agrega las alternativas consideradas

Un ADR se lee, sobre todo, cuando alguien quiere cambiar la decisión. Esa persona trae una alternativa y necesita saber:

- Si **se consideró y se descartó**, y por qué. Si la razón ya no vale, tiene el argumento para reabrirla.
- Si **nadie la consideró**. Entonces vale la pena discutirla.
- Qué **costo** aceptó el equipo, para no descubrirlo como un defecto.

> **Idea clave**
>
> Un ADR es la frase de trade-off de la semana 2 con formato: para cumplir *este escenario*, elegimos *esto* en vez de *aquello*; ganamos *esto*; aceptamos *esto otro*.

El template del curso

code/week-07/adr-template.md (resumido)

```
# ADR 0000: <la decisión, en una frase corta>
## Contexto
<Qué pasa y por qué hay que decidir ahora.>
**Escenario:** <el escenario, con su medida>
## Alternativas consideradas
### A. <nombre de la alternativa>
- **Gana:** <qué mejora>
- **Cuesta:** <qué empeora>
- **Por qué no:** <si se descartó>
## Decisión
- **Alternativa elegida:** <letra>
<Vamos a... porque...>
## Estado
Propuesto
## Consecuencias
- **Buena:** ...   - **Mala:** ...
```

El revisor comprueba la forma

- Las cinco secciones, en orden.
- El escenario tiene un número.
- La elegida y al menos dos más.
- Toda alternativa gana y cuesta.
- Una consecuencia buena y una mala.
- Un reemplazado apunta a su reemplazo.

Estados: `Propuesto`, `Aceptado`, `Rechazado`, `Obsoleto`, `Reemplazado por el ADR NNNN`.

Un ADR completo: el contexto del 0003 de Yupana

adr/0003-tabla-de-trabajos-y-workers-de-calificacion.md

```
Reemplaza al ADR 0001. Desde el piloto cambiaron tres cosas. Yupana se usa en
6 cursos, con tareas de hasta 300 estudiantes. Las tareas de Algoritmos
tienen casos de prueba grandes, y calificar un envío tarda en promedio 12 s.
...
**Escenario:** en los 30 minutos antes de un cierre llegan 600 envíos (20 por
minuto) que tardan en promedio 12 s en calificarse; la API confirma cada
envío en 1 s o menos (p95), el veredicto aparece en 2 min o menos (p95) y no
se pierde ningún envío.

A ese ritmo llegan 240 s de trabajo de calificación por cada minuto de reloj:
cuatro procesadores ocupados todo el tiempo, sin contar la API ni la base de
datos.
```

- Hechos, una restricción y un escenario con medida. Ninguna opinión.
- La cuenta ($20 \cdot 12 = 240$ s por minuto) es parte del contexto: es lo que vuelve insostenible la decisión anterior.

El 0003: la alternativa que no se eligió

adr/0003-tabla-de-trabajos-y-workers-de-calificacion.md

```
### C. Un broker de mensajes (NATS JetStream) y workers suscritos

- **Gana:** el broker entrega cada trabajo a un worker sin consultas
  periódicas, y vuelve a entregar el que no fue confirmado.
- **Cuesta:** un servicio más que instalar, vigilar y respaldar. Y el envío
  (en PostgreSQL) y su mensaje (en el broker) ya no se escriben en una
  transacción: un reinicio entre los dos deja un envío confirmado que nadie
  va a calificar.
- **Por qué no:** frente a una medida de 2 min, los 2 s de la consulta
  periódica no importan; el servicio adicional y la escritura en dos lugares
  sí, para un equipo de dos personas.
```

- No dice que un broker sea malo: dice que para este escenario y con este equipo no compensa.
- Se eligió B, una tabla de trabajos. Y quien lo lea en dos años sabe cuándo reabrir la decisión.

El 0003: las consecuencias malas

adr/0003-tabla-de-trabajos-y-workers-de-calificacion.md

```
- **Mala:** aparecen los estados "pendiente" y "en calificación". La
  aplicación web tiene que mostrarlos y preguntar por el veredicto.
- **Mala:** la tabla de trabajos trae decisiones que antes no existían y que
  este ADR no toma: cuántos workers, qué se hace con un trabajo que un worker
  tomó y no terminó, y si todos los trabajos valen lo mismo.
```

Guarden la segunda

Abre tres preguntas y no responde ninguna. En la sesión B, una evaluación va a encontrar cinco riesgos: tres son esas preguntas.

Error común

Vender la decisión en las consecuencias. Si no tuviera costos, no habría hecho falta decidir.

Un ADR no se reescribe



Cuando cambia el contexto

- Se escribe un ADR **nuevo**, que dice a cuál reemplaza y qué cambió.
- Cuando se acepta, en el anterior cambia una sola línea: el estado.
- Mientras el nuevo esté solo propuesto, el anterior sigue vigente.

Idea clave

Su valor está en poder reconstruir **qué se sabía cuando se decidió**. El 0001 no fue un error: con su escenario, era la decisión correcta.

Qué merece un ADR

Tres preguntas:

1. ¿**Cuesta cambiarla**?
2. ¿Alguien va a preguntar **por qué está así**?
3. ¿**Parece rara** sin su contexto?

Merece un ADR	**No lo merece**
Dónde vive cada dato y quién lo escribe	Cómo se llama una tabla
Si una operación es síncrona o asíncrona	El nombre de la función
Una táctica que cuesta otro atributo	Un parámetro que se ajusta midiendo
Una regla que se decide no cumplir	Lo que el linter ya impone

En Tinkuy: el Tinkuy inicial no tiene ningún ADR. Desde esta semana, cada entrega lleva sus decisiones en `docs/adr/`, y las de entregas anteriores no se reescriben.

Dos errores al escribir un ADR

El ADR 0004 de `errors/`. Contexto: “La calificación es lenta y los estudiantes se quejan.” Una alternativa, NATS: “es rápido y moderno.” Decisión: “Decidimos usar NATS.” Estado: “Decidido”.

```
cd errors && node ../check-adr.ts adr (extracto)

adr/0004-usar-un-broker.md:7: [alternativas] hay 1 alternativa: hacen falta la elegida y al menos dos que se descartaron
...
adr/0004-usar-un-broker.md: ADR 0004, estado desconocido, 1 alternativa (sin elegida): 6 errores
```

Sin alternativas

Sin lo que se descartó, nadie puede saber si la decisión sigue vigente cuando cambie el contexto.

Documentar al final

Para justificar lo que ya se hizo. Se nota: las descartadas no tienen nada bueno. El ADR sirve cuando se escribe **mientras** se decide.

SECCIÓN 06

Diseñar con un método: ADD

06

Diseño guiado por atributos (ADD): siete pasos

1. **Revisar las entradas**: propósito, funciones, escenarios, restricciones.
2. **Fijar el objetivo**: uno o dos escenarios.
3. **Elegir qué elemento** se refina.
4. **Elegir conceptos de diseño**: estilos, tácticas, tecnologías.
5. **Instanciar** elementos, con responsabilidades e interfaces.
6. **Dibujar las vistas y registrar** las decisiones.
7. **Analizar** y decidir si hace falta otra iteración.

El paso 1 se hace al empezar; cada iteración repite del 2 al 7.

Ya lo conocen

Paso 4: estilos y tácticas. Paso 6: C4 y un ADR.

Lo que el método ordena

Se empieza por los escenarios, no por la tecnología. Cada decisión queda atada a un escenario. Se sabe cuándo parar.

Una iteración sobre Yupana: el paso 4

Objetivo: E1, que tiene dos medidas: la confirmación (tiempo bloqueado) y el veredicto (capacidad).

Táctica (semana 6)	Confirmación en 1 s	Veredicto en 2 min
Más recursos	No: el pedido sigue esperando	En parte
Una cola: recibir ahora y calificar después	Sí	No sola: ordena la espera
Varias copias del cómputo	No sola	Sí, mientras haya núcleos

Recién ahora aparece la tecnología

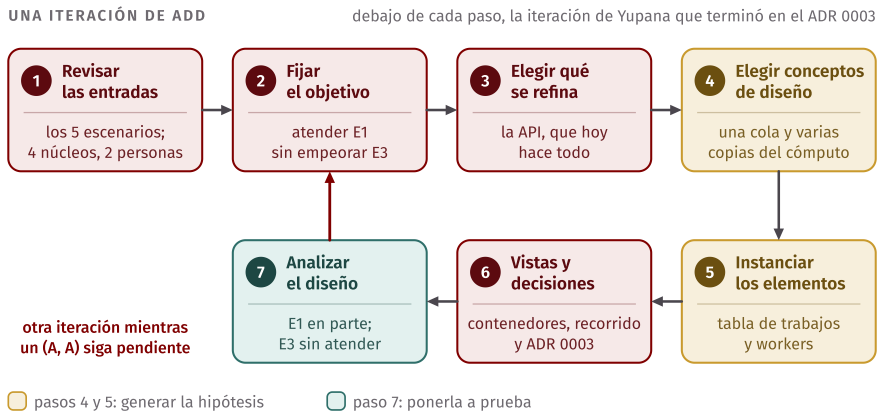
Con qué se implementa la cola, una tabla o un broker: las alternativas del ADR 0003.

El paso 7: ¿hace falta otra iteración?

Escenario	Antes	Después de la iteración
E1, confirmación en 1 s	No atendido	Atendido: confirmar es una transacción corta
E1, veredicto en 2 min	No atendido	En parte: depende de cuántos workers corran
E3, un worker muere	No existía	No atendido: falta la regla para un trabajo abandonado
E2, programa hostil	En parte	En parte: el código ya no corre en la API

- Mientras haya un (A, A) “no atendido” o “en parte”, falta una iteración.
- El diseño termina cuando los escenarios prioritarios tienen respuesta, no cuando se acaban las ideas.

La iteración de Yupana, paso por paso



Generar y probar

Una decisión de diseño es una hipótesis

“Con una tabla de trabajos, la confirmación tarda menos de 1 s en un cierre” puede ser falso.

Prueba	Qué es	Cuesta	Qué tan segura deja
Análisis	Una cuenta o un recorrido	Minutos	Detecta lo que no puede funcionar
Prototipo	Lo mínimo para ver el comportamiento	Horas o días	Confirma el mecanismo
Medición	El diseño implementado, con carga	Días	La única que confirma una medida

Se usan en ese orden. El proyecto del curso es este ciclo, en cada entrega.

SECCIÓN 07

Evaluar antes de construir

07

Evaluar antes de construir

Quien diseñó ya está convencido. Una evaluación es el mismo análisis con otras personas en la sala.

Forma	Quiénes	Dura	Alcanza para
Revisión entre pares	Quien diseñó y uno o dos colegas	Una hora	Un ADR, un cambio en un módulo
Recorrido de escenarios	El equipo	Una o dos horas	Un rediseño dentro de un equipo
ATAM	Evaluadores, diseñadores y stakeholders	Días	Sistemas grandes o caros

Error común: evaluar contra requisitos que nadie escribió

“Esto no escala.” “Sí escala.” Sin escenarios, una evaluación es una discusión de gustos.

SECCIÓN 08

ATAM

08

ATAM: Architecture Tradeoff Analysis Method

Para qué sirve

Para entender las consecuencias de las decisiones de arquitectura a la luz de los requisitos de atributos de calidad. Es un método de **identificación de riesgos**.

El nombre: además de mirar cada atributo, muestra dónde mejorar uno empeora otro.

Quiénes

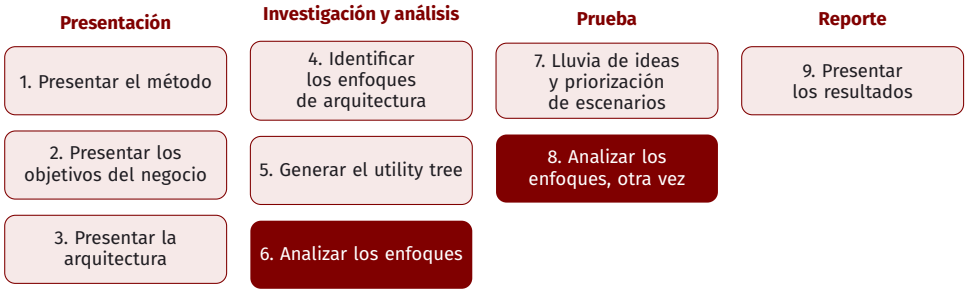
- El **equipo evaluador**, que no diseñó.
- **Quienes deciden**: dirección del proyecto, cliente, arquitecto.
- Los **stakeholders**: quienes usan, operan, mantienen y prueban.

Qué hace falta

- Objetivos del negocio.
- La arquitectura descrita.
- Escenarios; si no hay, el método los produce.

Kazman, Klein y Clements, *ATAM: Method for Architecture Evaluation*, SEI, 2000.

Los nueve pasos



- El paso 4 **lista** los enfoques; todavía no los analiza.
- El 6 y el 8 son la misma actividad, con escenarios de distinto origen.
- En la práctica, dos fases: un grupo pequeño (pasos 1 a 6) y, semanas después, todos los stakeholders.

Por qué se analiza dos veces

Paso 5: el utility tree

Lo construyen quienes diseñaron: tiene los escenarios en los que ya pensaron.

Paso 7: la lluvia de ideas

Escenarios de quien opera, da soporte o usa: de **uso**, de **crecimiento** y **exploratorios**.

- Se vota: cada persona recibe votos por el 30 % de la cantidad de escenarios, redondeado hacia arriba.
- Si la lista votada coincide con el utility tree, el paso 8 confirma.
- Si no, esa es la noticia: los stakeholders creen importante algo que quienes diseñaron no tenían en la lista.

En la industria: la agenda típica del reporte del SEI son tres días, con una pausa de varias semanas después del primero. El método completo se usa donde un error de arquitectura cuesta mucho más.

Lo que produce

Hallazgo	Qué es	Pregunta que lo detecta
Riesgo	Una decisión (o una que falta) que puede impedir un escenario	¿Qué decisión hace que no se cumpla?
No riesgo	Una decisión que está bien, bajo un supuesto	¿Qué tendría que cambiar para que deje de estarlo?
Punto de sensibilidad	Una propiedad de la que depende mucho un atributo	¿Qué parámetro mueve mucho la medida?
Punto de trade-off	Un punto de sensibilidad para dos atributos o más	¿A qué otro atributo afecta?

Las cuatro listas clasifican decisiones.

Cómo se escriben y cómo se relacionan

- Un riesgo tiene tres partes: la **decisión**, la **respuesta** del atributo y el **razonamiento** que las une.
- Un no riesgo se anota igual: deja dicho qué supuesto hay que volver a mirar.
- No son excluyentes. Un punto de sensibilidad dice dónde mirar; el valor que tiene hoy puede ser un riesgo.
- Los **temas de riesgo**: lo que varios riesgos tienen en común, comparado con los objetivos del negocio.

Error común: anotar síntomas en vez de decisiones

"El sistema podría ser lento en los cierres" no dice qué decisión lo causa: nadie sabe qué cambiar.

En Tinkuy: dos ejemplos

“La pasarela se llama dentro de la transacción”

Punto de sensibilidad para la latencia: el tiempo de la pasarela entra completo en la respuesta.

Riesgo para la disponibilidad: la compra necesita que la pasarela responda.

“TTL de cinco segundos en la cache del catálogo”

Punto de trade-off entre rendimiento y actualidad de los datos: subirlo quita carga y alarga el tiempo en que se ven asientos que ya no están.

Los demás hallazgos de Tinkuy son el trabajo del laboratorio.

El recorrido de un escenario: E1 sobre Yupana

Quien diseñó recorre el escenario sobre los diagramas. Quien evalúa hace las preguntas del atributo:

- ¿A qué ritmo llega? **20 por minuto**, durante 30 min. Lo dice el escenario.
- ¿Cuánto tarda cada uno? **12 s**.
- ¿Cuántos a la vez? Uno por worker. ¿Y cuántos workers hay?
  El ADR: “es configuración del despliegue”. La vista de despliegue: 2.
- ¿Dónde espera lo demás? En la tabla de trabajos, en orden de llegada.

> **El hallazgo sale de cruzar tres documentos**
>
> El ADR, la vista de despliegue y el escenario. Ninguno, solo, muestra el problema.

La cuenta

Con λ envíos por minuto, k workers y s segundos por envío, los workers califican $\mu = 60\,k/s = 60 \cdot 2/12 = 10$ por minuto. Si $\lambda > \mu$, al final de T minutos quedan $(\lambda - \mu)\,T$ pendientes. Para no acumular hacen falta $k_{\text{mín}} = \lceil \lambda \cdot s/60 \rceil$ workers.

```
make queue ARRIVALS=20 MINUTES=30 WORKERS=2 SECONDS=12

llegan 20 por minuto durante 30 min; 2 workers, 12 s por trabajo
capacidad:            10 por minuto
utilización:          2
pendientes al final:  300
espera del último:    30 min
workers para no acumular: 4
```

30 minutos contra una medida de 2

No hizo falta construir ni medir nada para saberlo.

Lo que se anota

- **R1, riesgo**: correr 2 workers. El veredicto tarda hasta 30 min contra una medida de 2. Y 4 workers ocuparían los 4 núcleos.
- **S1, punto de sensibilidad**: el número de workers y el límite de tiempo de cada tarea. El doble de segundos por envío pide el doble de workers.
- **N1, no riesgo**: confirmar con una transacción de dos filas. Supuesto: la base de datos no está saturada.
- **T1, punto de trade-off**: un sandbox nuevo por envío. Seguridad contra rendimiento.

Dos cosas del recorrido

El análisis es rudimentario a propósito: sirve para descartar. Y nadie propuso una solución: eso es otra decisión, con su ADR.

E2 y E3: lo que encontró el recorrido

	Tipo	Decisión	Razonamiento
N2	no riesgo	Ejecutar cada programa sin red y con límites de CPU, memoria y procesos	No puede abrir conexiones ni agotar la máquina. Supuesto: el kernel no tiene una falla conocida
R2	riesgo	Los workers usan el mismo usuario de PostgreSQL que la API	Si un programa escapa del sandbox, lee todos los envíos y cambia cualquier veredicto
R3	riesgo	Montar en el sandbox la carpeta del caso, con la salida esperada	El programa puede leerla e imprimirla: “aceptado” sin resolver la tarea
R4	riesgo	Falta decidir qué pasa con un trabajo que un worker tomó y no terminó	Si el worker muere, el envío queda sin veredicto para siempre; el escenario pide 5 min
T2	punto de trade-off	Cuántas veces se reintenta un trabajo abandonado (sin decidir)	Más intentos recuperan más caídas (E3); un envío que mata al worker se ejecuta de nuevo cada vez (E2)

- N2 y R2 juntos: el sandbox está bien bajo un supuesto, y si el supuesto falla no hay una segunda barrera *(mínimo privilegio, semana 6)*.
- R4 es una decisión sin tomar: también es un riesgo.

El paso 7 en Yupana: el escenario que nadie había pensado

Escenario propuesto	Clase	Votos
Un docente corrige un caso de prueba cuando ya hay 250 envíos: se recalifican en 60 min y los envíos nuevos mantienen sus 2 min	de uso	4
Dos tareas cierran a la misma hora: 40 envíos por minuto	de crecimiento	2
El disco del servidor se llena durante un cierre	exploratorio	1
Se agrega la detección de copias entre envíos	de crecimiento	1

- 4 personas, 4 escenarios: 2 votos cada una (el 30 % de 4 es 1,2).
- El primero lo propuso el docente. Para quienes diseñaron, los envíos llegaban de los estudiantes, de a uno.
- **R5, riesgo**: atender en orden de llegada, sin prioridades. 250 trabajos a 10 por minuto son 25 min en los que todo envío nuevo espera.
- El segundo no encontró nada nuevo: es otra manera de mover S1. Es lo que el paso 8 espera.

Temas de riesgo

Tema	Riesgos	Objetivo que amenaza
La tabla de trabajos se adoptó como almacenamiento y nadie decidió sus reglas de cola	R1, R4, R5	El veredicto llega mientras el estudiante todavía trabaja
El límite de confianza rodea al worker, y debería rodear al programa del estudiante	R2, R3	Un veredicto dice la verdad

- “Tenemos cinco riesgos” no le dice nada al negocio. “La noche de un cierre van a esperar media hora” sí.
- El primer tema ya estaba escrito: es la consecuencia mala del ADR 0003. Las consecuencias malas de un ADR son lo primero que mira la siguiente evaluación.

Lo que ATAM no hace

- **No dice si la arquitectura es buena.** Yupana salió con cinco riesgos: cinco cosas que mirar.
- **No da números.** La cuenta dice “no alcanza”; no dice cuál será el p95.
- **No reemplaza a medir.** Un no riesgo es una hipótesis que pasó un análisis.
- **No arregla nada.** Localiza riesgos; qué hacer con ellos se decide después.

Idea clave

Dice **dónde mirar**: convierte “¿esta arquitectura está bien?” en una lista corta de decisiones atadas a escenarios.

SECCIÓN 09

Una versión liviana

09

Una versión liviana para un equipo pequeño: una hora

Minutos	Qué se hace	Paso del método
5	Para qué existe el sistema: dos o tres objetivos	2
10	La arquitectura, usando solo los diagramas	3
5	Listar los enfoques: estilo, datos, llamadas externas, tácticas	4
30	Un recorrido de 10 minutos por cada escenario (A, A)	5 y 6
10	Agrupar los riesgos en temas y elegir cuál va primero	9

Hace falta alguien que no haya diseñado la parte que se evalúa. Su trabajo es preguntar, no defender.

La pregunta de la versión liviana

Por cada escenario (A, A)

¿Qué decisión de nuestro diseño lo hace posible, y cuál lo pone en peligro?

- La primera mitad produce los no riesgos y los puntos de sensibilidad.
- La segunda mitad produce los riesgos.
- La misma decisión en las dos mitades de dos escenarios distintos: un punto de trade-off.

Lo que se pierde: el paso 7

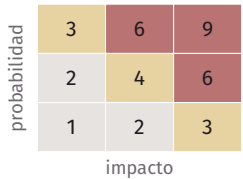
Si hay diez minutos más, valen para eso. En Yupana encontró el único riesgo que quienes diseñaron no habrían visto.

Risk storming *(opcional)*

Empieza por los **diagramas** *(Simon Brown)*:

1. Dibujar los diagramas, en varios niveles.
2. Cada persona, por separado y en silencio, escribe sus riesgos con su prioridad.
3. Las notas se pegan sobre el diagrama.
4. Revisar lo que marcó una sola persona, y donde no coinciden en la prioridad.

Las discrepancias son lo más útil.



6 o 9: alta; 3 o 4: media; 1 o 2: baja.

Trade-off: es más rápido y no necesita escenarios; a cambio, los riesgos no quedan atados a una medida. Sirve para decidir dónde mirar; un recorrido dice qué tan grave es.

Laboratorio: un ATAM reducido sobre Tinkuy

Dos parejas: una **presenta** su Tinkuy y la otra lo **evalúa**; después cambian. Material: `code/week-07/`.

1. Antes: actualizar contexto y contenedores, y agregar `docs/c4/components.md`. `make check-c4 DOCS=...`
2. Presentación (10 min): objetivos, los tres escenarios (A, A) y la arquitectura, solo con los diagramas.
3. Quien evalúa lista los enfoques de arquitectura y dónde se ve cada uno.
4. Recorrer cada escenario (A, A) y llenar la tabla (`atam-template.md`).
5. Cada persona propone un escenario nuevo. Votar dos y recorrerlos.
6. Agrupar los riesgos en temas. `make check-atam DOCS=...`
7. Cambiar de papeles.
8. Un ADR para el riesgo más grave (`adr-template.md`), con al menos tres alternativas.

Para revisar el ADR de la otra pareja: ¿alguien que no estuvo hoy entendería por qué se decidió así?

Para la próxima clase

- Terminar la tabla y el ADR del laboratorio.
- La próxima semana se entrega **P2 · La apertura de ventas**. Sus ADR siguen el template de hoy.
- Semana 8: repaso y examen parcial. Resolver las preguntas de repaso de las guías de las semanas 1 a 7.

> **Pregunta**
>
> Hace seis meses se aceptó un ADR que ya no se cumple. ¿Qué se hace con él, y por qué no se edita?

Lecturas de hoy: *Software Architecture in Practice*, cap. 20, 21 y 22; Nygard, *Documenting Architecture Decisions*; Brown, *C4: diagramas*; Kazman, Klein y Clements, *ATAM: Method for Architecture Evaluation*, secciones 3 y 7.