Universidad Rafael Landívar
Tradición Jesuita en Guatemala

# Análisis y Diseño II

## Fundamentos

# Agenda

1. Bienvenida.
2. Toma de decisiones.
3. RAP.
4. Mini-lecture.
5. Casos.
6. Pasos siguientes.

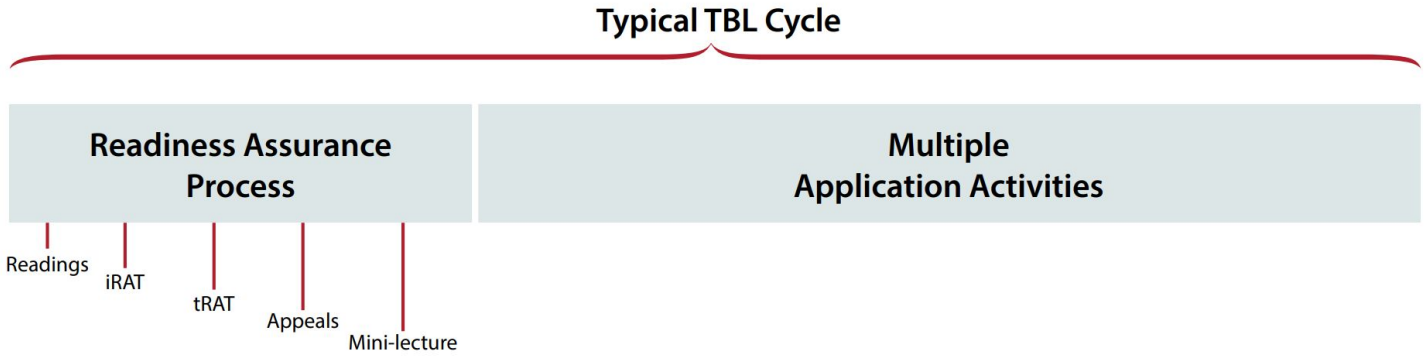

# Toma de decisiones

## Microenseñanzas

# Selección de temas

| Fecha | Tema | Actividad de evaluación |
|---|---|---|
| Semana 1<br>15 – 19 mayo | Introducción: definición de arquitectura de software, perfil del arquitecto, prácticas de ingeniería, operaciones y devops, fundamentos.<br>Pensamiento arquitectónico: arquitectura y diseño, equilibrio de arquitectura y codificación.<br>Modularidad: definición, cohesión, acoplamiento, métricas, transición a componentes. | **Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos.<br>**Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos. |
| Semana 2<br>22 - 26 mayo | Características arquitectónicas: operativas, estructurales, transversales.<br>Extracción del dominio, de los requerimientos, explícitas e implícitas.<br>Caso de Sand. de Silicón. | |
| Semana 3<br>29 mayo – 2-junio | Fundamentos: patrones, unitary, cliente servidor, desktop + database server, browser + webserver, three-tier, monolithic vs. distributed, falacias de diseño. | **Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos.<br>**Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos<br>**Parcial I** |
| Semana 4<br>5 – 9 junio | Estilos de arquitectura por capas. | **Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos.<br>**Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos |
| Semana 5<br>12 – 16 junio | Estilo de arquitectura basada en servicios. | |
| Semana 6<br>19 – 23 junio | Estilo de arquitectura basada en microservicios.<br>Cómo elegir el estilo de arquitectura adecuada. Caso de estudio. | **Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos.<br>**Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos<br>**Parcial II** |
| Semana 7<br>26 – 30 junio | Decisiones de arquitectura: anti-patrones de diseño, registros de decisiones de arquitectura, estructura básica (ADR), analizando riesgos de la arquitectura con modelos ágiles, diagramación y presentación de arquitecturas (UML, C4, ArchiMate). | **Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos.<br>**Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos |
| Semana 8<br>3 – 7 julio | Equipos de arquitectos, negociación y liderazgo.<br>Validación y verificación | **Examen final.** |

# RAP



# ¿Cómo es mi preparación con la lectura?

Students, drag the icon!

# iRAT

decisiones

# gRAT

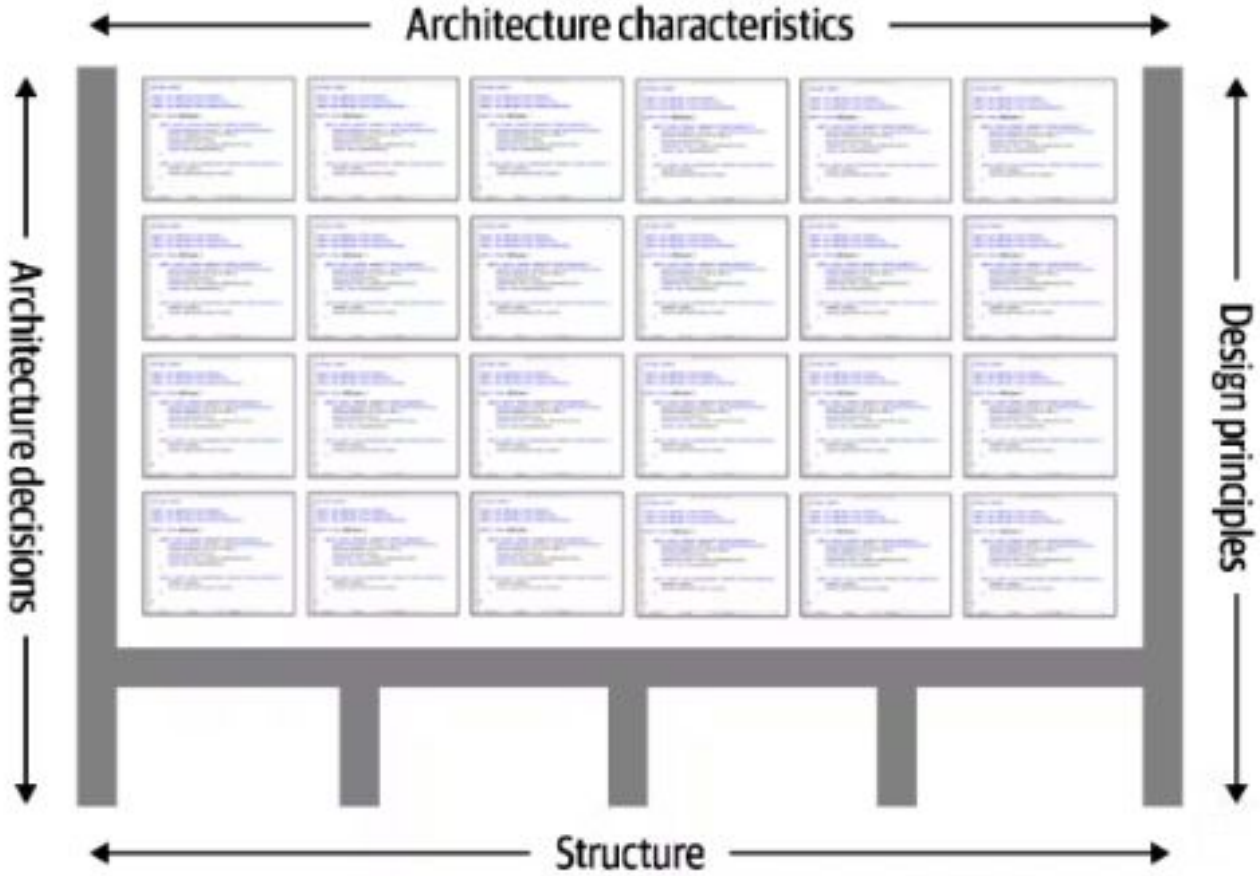
compensacion

# Apelaciones

Students, write your response!


# Mini-lecture

## Arquitectura de Software


Architecture characteristics
Architecture decisions
Design principles
Structure

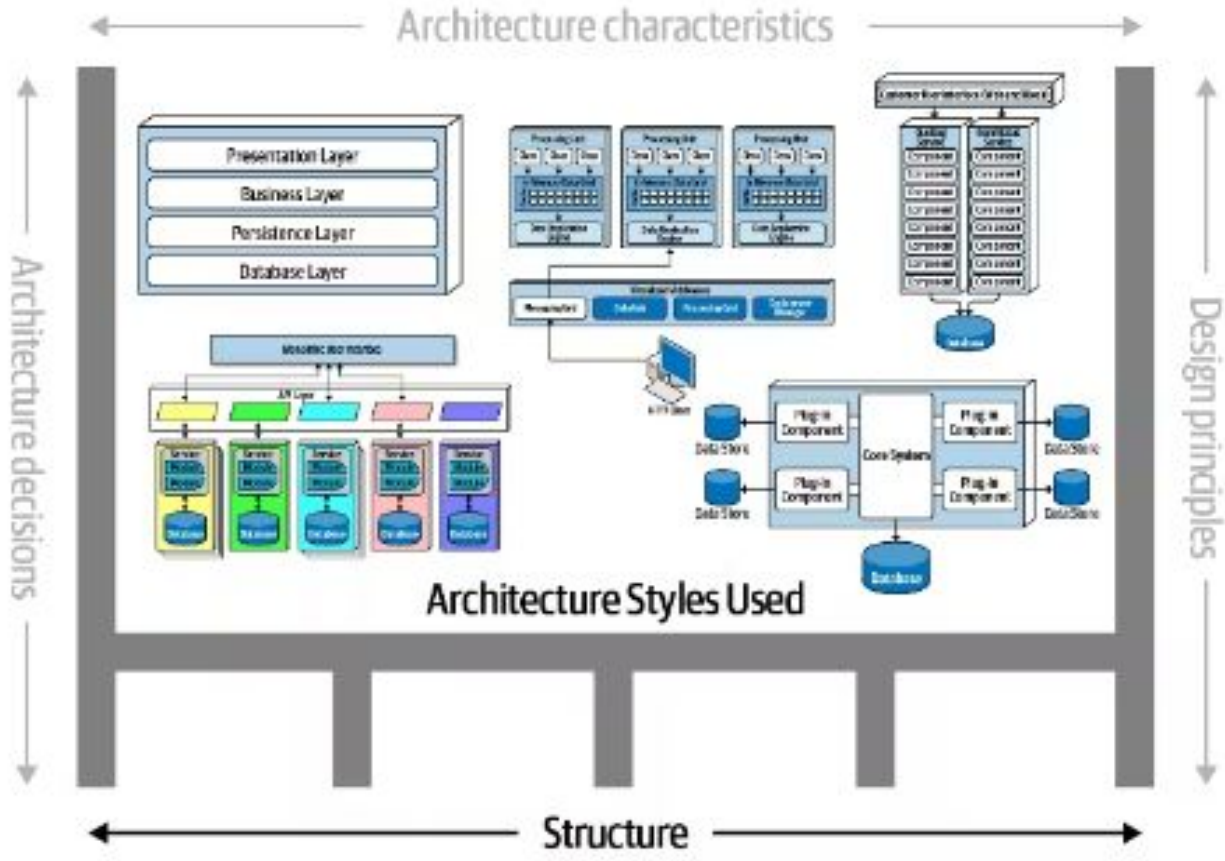
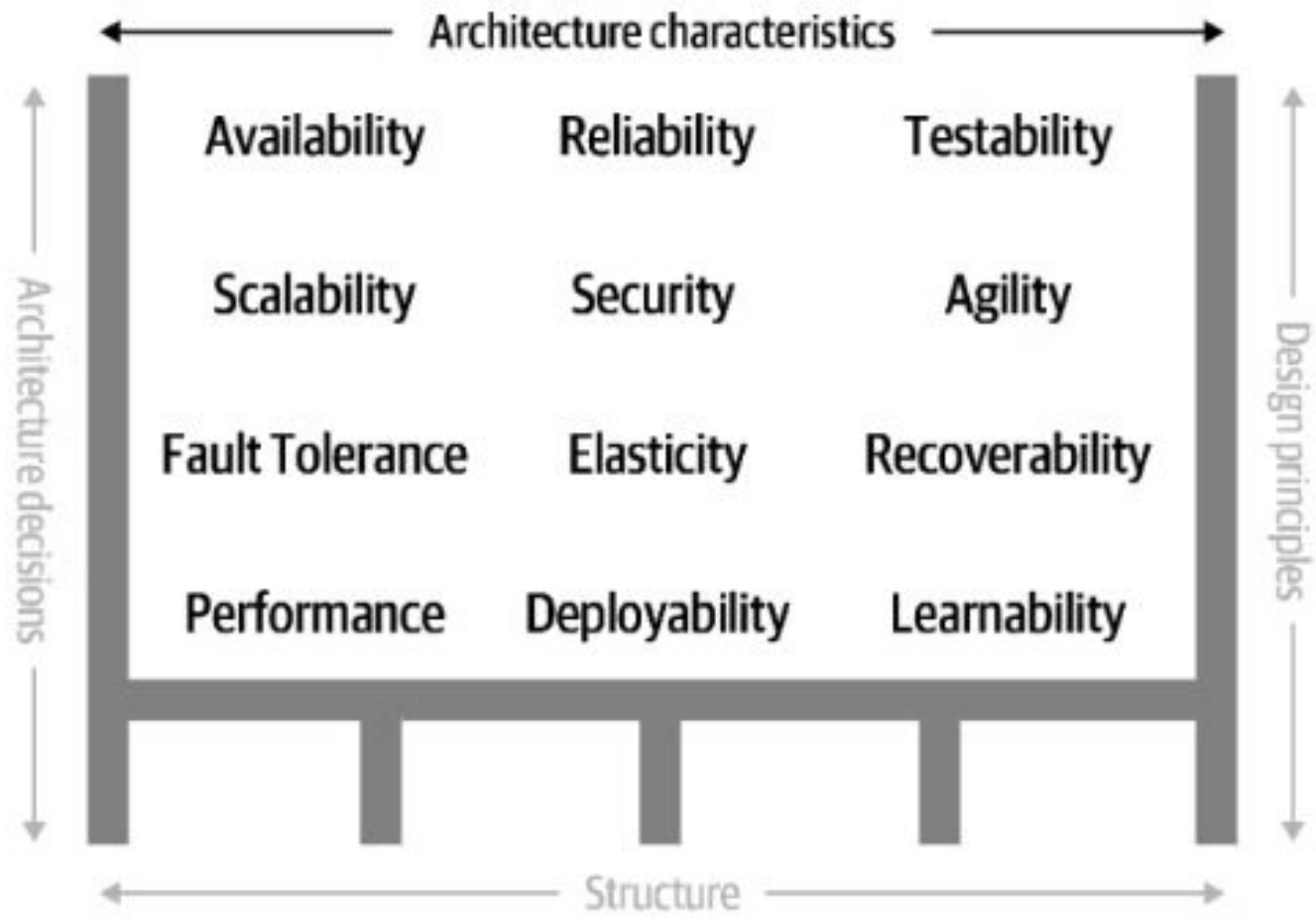



*Figure 1-3. Structure refers to the type of architecture styles used in the system*


Architecture characteristics
Availability
Reliability
Testability
Scalability
Security
Agility
Fault Tolerance
Elasticity
Recoverability
Performance
Deployability
Learnability
Architecture decisions
Design principles
Structure

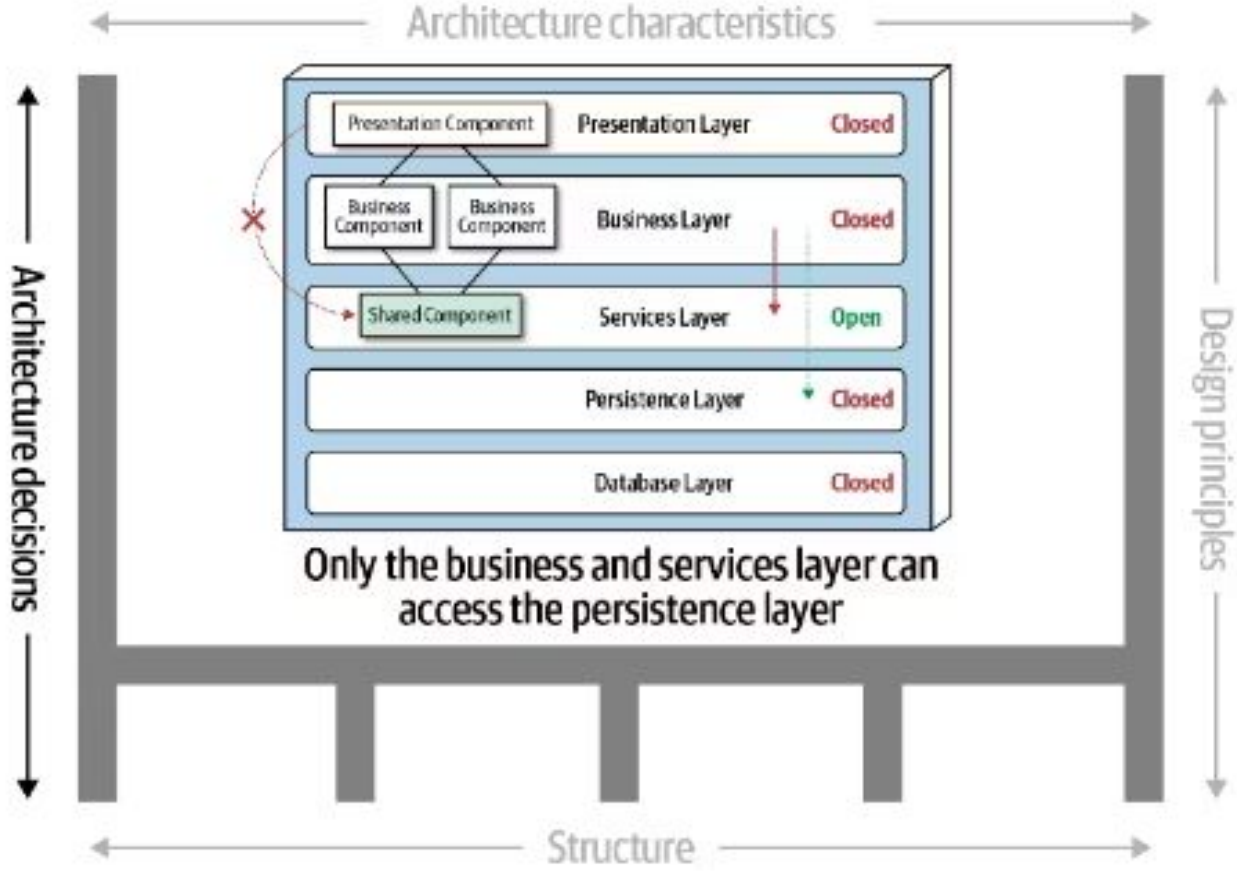



*Figure 1-5. Architecture decisions are rules for constructing systems*

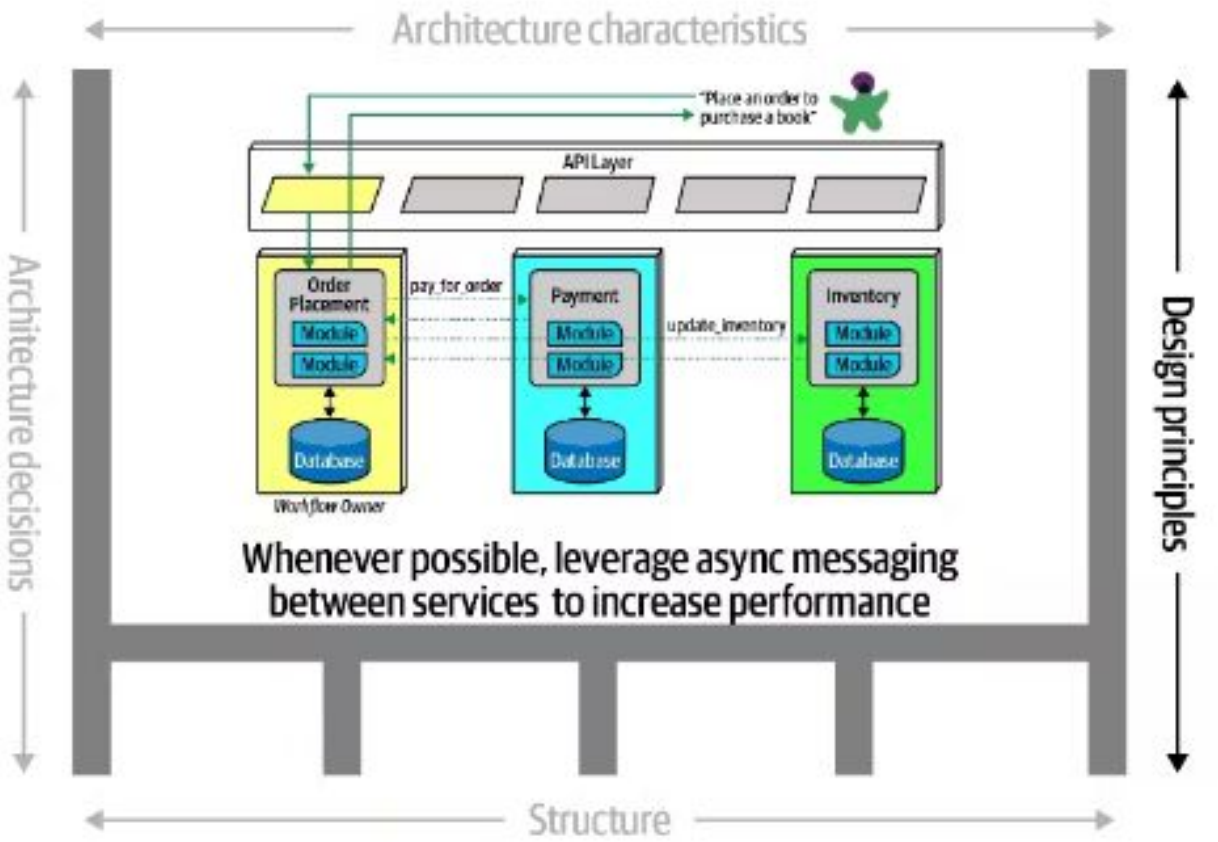


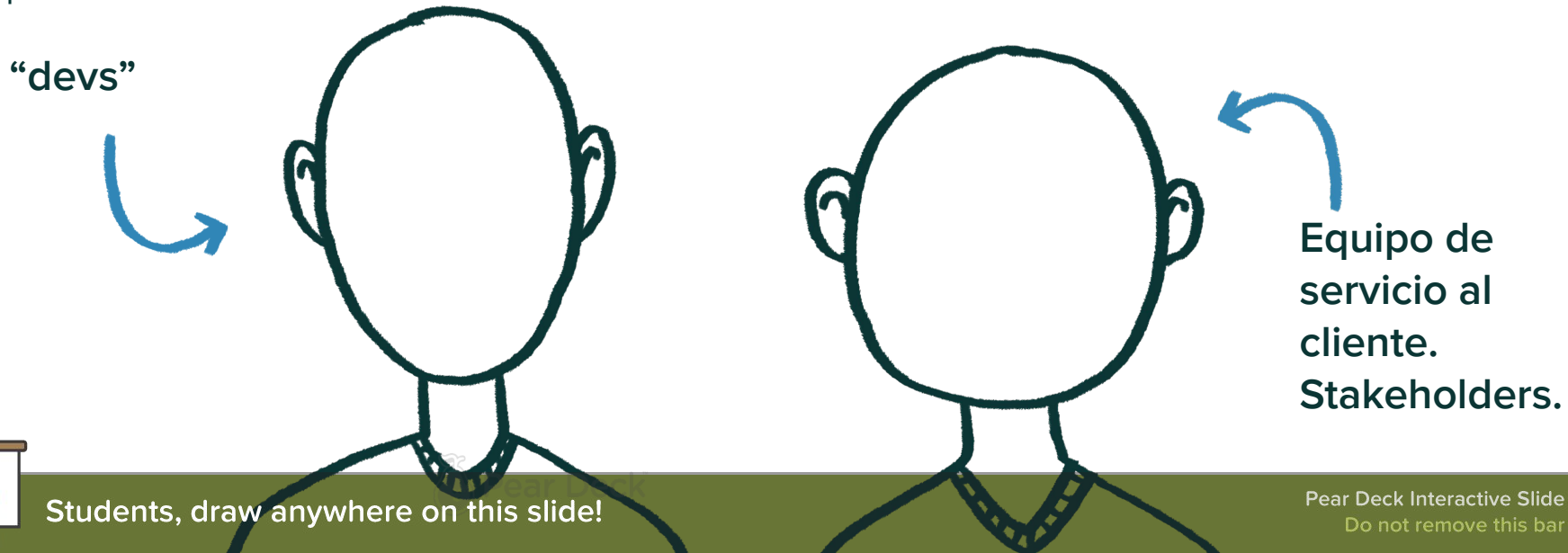

*Figure 1-6. Design principles are guidelines for constructing systems*

# Casos

De aplicación.

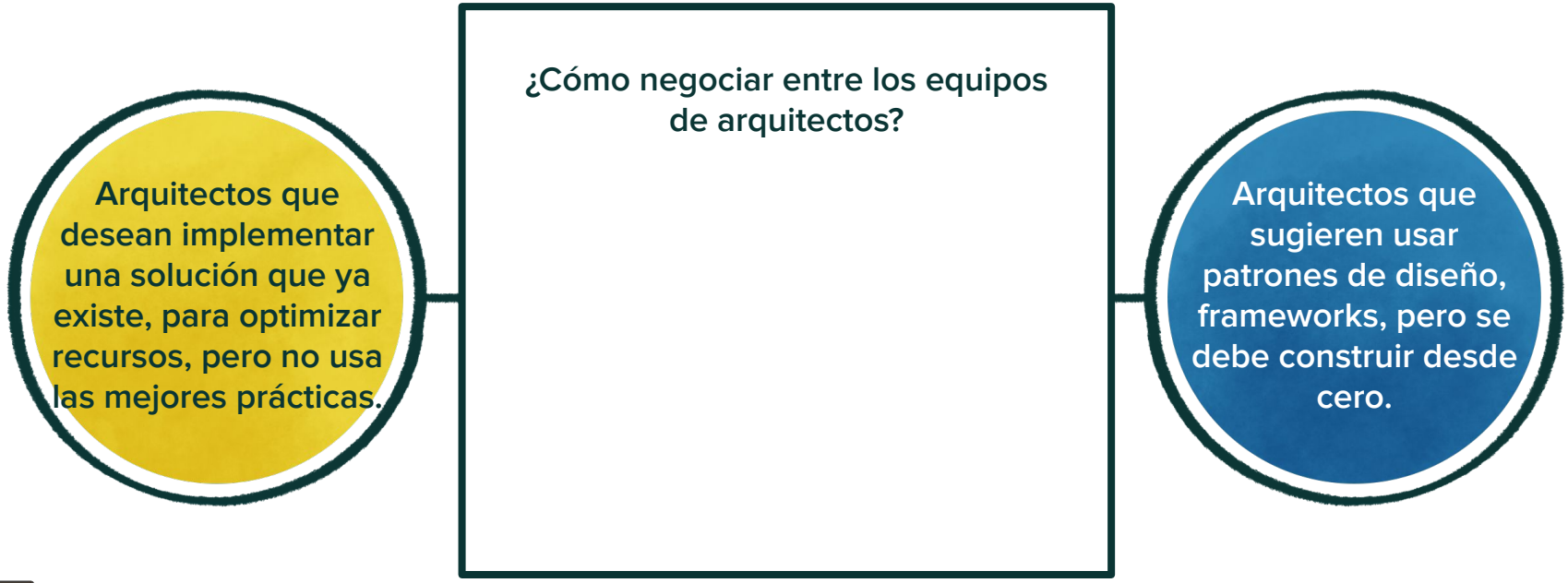
# Analice los puntos de vista

**Ofrezca recomendaciones para ambos casos**: en una empresa de servicios digitales para agricultores pequeños, ninguna de las apps liberadas ha funcionado. Los usuarios no las pueden utilizar, no las comprenden y cuando servicio al cliente lo reporta a los devs se alarga la lista del product backlog. Servicio al cliente pidió al líder de transformación digital que los devs ahora vayan a campo y ellos implementen las soluciones, además, contratar a una empresa externa que diseñe la arquitectura de las soluciones.



# Analice y recomiende

¿Qué está faltando en este escenario?




Students, draw anywhere on this slide!

# Explique

¿Cómo sirve el caso para comprender los conceptos siguientes?

- Contexto.
- Toma de decisiones.
- Evolución.
- Análisis de la arquitectura.

**HISTORY: PETS.COM AND WHY WE HAVE ELASTIC SCALE**

The history of software development contains rich lessons, both good and bad. We assume that current capabilities (like elastic scale) just appeared one day because of some clever developer, but those ideas were often born of hard lessons. Pets.com represents an early example of hard lessons learned. Pets.com appeared in the early days of the internet, hoping to become the Amazon.com of pet supplies. Fortunately, they had a brilliant marketing department, which invented a compelling mascot: a sock puppet with a microphone that said irreverent things. The mascot became a superstar, appearing in public at parades and national sporting events.

Unfortunately, management at Pets.com apparently spent all the money on the mascot, not on infrastructure. Once orders started pouring in, they weren't prepared. The website was slow, transactions were lost, deliveries delayed, and so on…pretty much the worst-case scenario. So bad, in fact, that the business closed shortly after its disastrous Christmas rush, selling the only remaining valuable asset (the mascot) to a competitor.

What the company needed was elastic scale: the ability to spin up more instances of resources, as needed. Cloud providers offer this feature as a commodity, but in the early days of the internet, companies had to manage their own infrastructure, and many fell victim to a previously unheard of phenomenon: too much success can kill the business. Pets.com and other similar horror stories led engineers to develop the frameworks that architects enjoy now.

# Realice una comparación de funciones

1. Arquitectura de software

2. Ingeniería de software

Students, draw anywhere on this slide!

¿Puede un modelo de proceso ayudar a mejorar una arquitectura? ¿Cómo?

Students, write your response!

# Provea un ejemplo de cada ley de la arquitectura de software.

1

**Ley 1:** Todo en la arquitectura de software es una compensación.
**Corolario**: si un arquitecto cree que ha descubierto un diseño sin compensaciones, posiblemente no ha identificado las compensaciones aún.

2

**Ley 2:** "Por qué" es más importante que "cómo".
¿Qué relación tiene con la modelación visual?

Students, draw anywhere on this slide!


# Información

## Laboratorio 19-mayo-2023.

# Pasos siguientes

Lectura previa. Solo evaluación individual tendrá puntos.

- Google Cloud Developer primera asignación.
- Pensamiento arquitectónico.
- Modularidad.

# ¿Cómo me siento con relación a lo aprendido hasta ahora?

# Créditos

Universidad Rafael Landívar
Tradición Jesuita en Guatemala