PREGRADO

UNIDAD 1 | OVERVIEW

SOFTWARE DESIGN FOUNDATION

SI720 | Diseño y Patrones de Software

Al finalizar la unidad de aprendizaje, el estudiante aplica los principios SOLID y buenas prácticas de arquitectura de información y datos para el diseño de aplicaciones, bajo el paradigma orientado a objetos.

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

¿Qué es la Ingeniería de Software?

La aplicación de un enfoque **sistemático**, **disciplinado** y **cuantificable** al **desarrollo**, **operación** y **mantenimiento** del software; esto es, la aplicación de la ingeniería al software.

IEEE

https://www.ieee.org

Aplicado a través de todo el ciclo de vida, desde requisitos al mantenimiento.

¿Por Qué Ingeniería de Software?

El **propósito** de la ingeniería de software es **controlar la complejidad**, no ***crearla***.

Dra. Pamela Zave

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

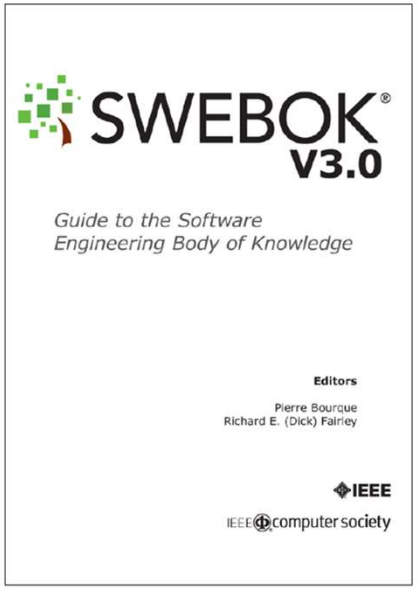
ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

Software Engineering Body of Knowledge

- IEEE Computer Society publicó el **SWEBOK** como un estándar internacional.

- SWEBOK v3, es presentado como un mecanismo para la adquisición de los conocimientos necesarios en el desarrollo de la carrera como un profesional de la ingeniería de software.



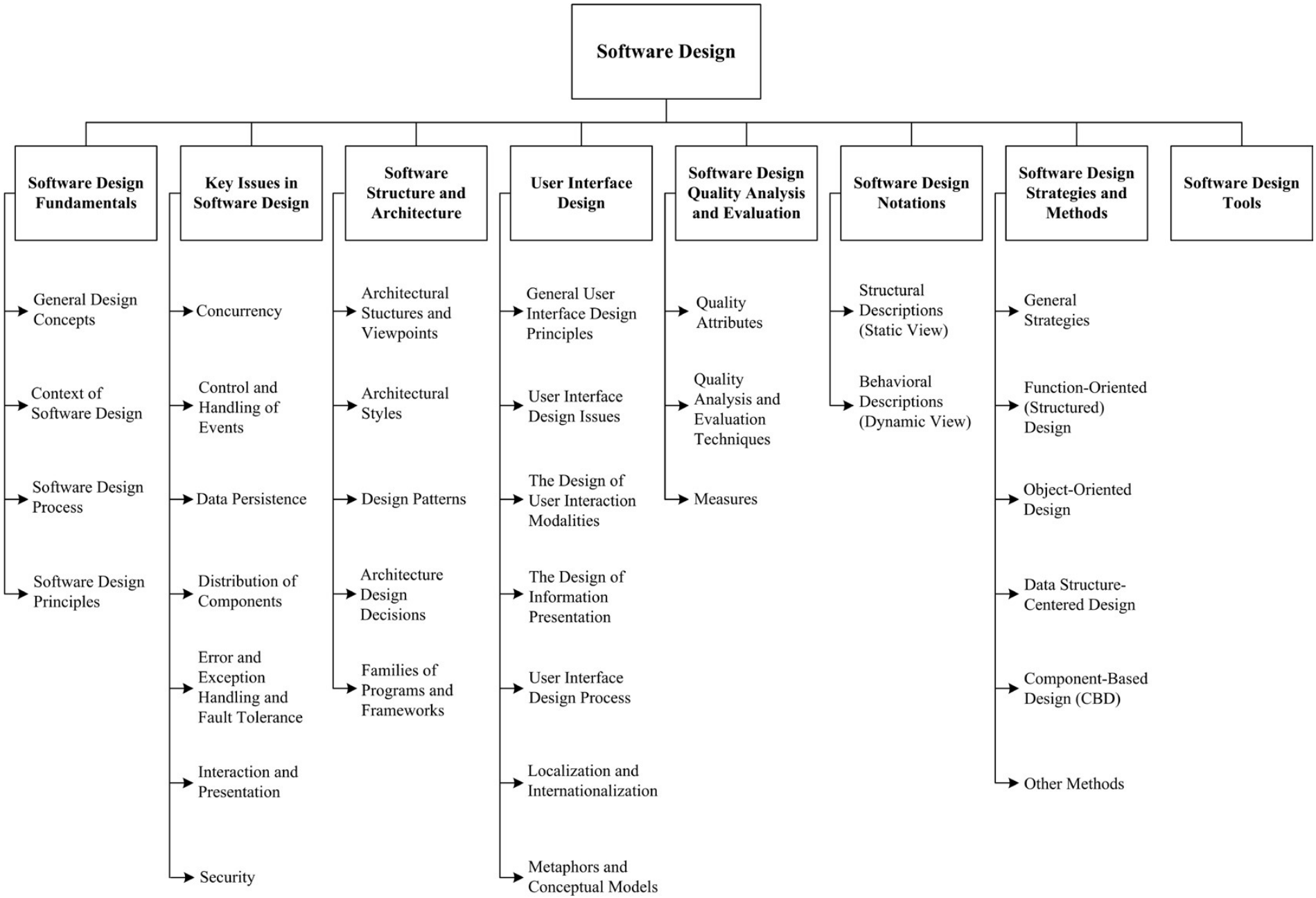
SWEBOK V3, 2014

1. Software Requirements **2. Software Design** 3. Software Construction 4. Software Testing 5. Software Maintenance 6. Software Configuration Management 7. Software Engineering Management 8. Software Engineering Process	9. Software Engineering Models and Methods 10. Software Quality 11. Software Engineering Professional Practice 12. Software Engineering Economics 13. Computing Foundations 14. Mathematical Foundations 15. Engineering Foundations

Knowledge Areas

SWEBOK V3, 2014 - Diseño de Software

TABLE OF CONTENTS


Software Design
Software Design Fundamentals
General Design Concepts
Context of Software Design
Software Design Process
Software Design Principles
Key Issues in Software Design
Concurrency
Control and Handling of Events
Data Persistence
Distribution of Components
Error and Exception Handling and Fault Tolerance
Interaction and Presentation
Security
Software Structure and Architecture
Architectural Stuctures and Viewpoints
Architectural Styles
Design Patterns
Architecture Design Decisions
Families of Programs and Frameworks
User Interface Design
General User Interface Design Principles
User Interface Design Issues
The Design of User Interaction Modalities
The Design of Information Presentation
User Interface Design Process
Localization and Internationalization
Metaphors and Conceptual Models
Software Design Quality Analysis and Evaluation
Quality Attributes
Quality Analysis and Evaluation Techniques
Measures
Software Design Notations
Structural Descriptions (Static View)
Behavioral Descriptions (Dynamic View)
Software Design Strategies and Methods
General Strategies
Function-Oriented (Structured) Design
Object-Oriented Design
Data Structure-Centered Design
Component-Based Design (CBD)
Other Methods
Software Design Tools

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

Instituto de Ingeniería de Software (SEI)

- El **SEI** es un organismo autorizado y de reputación en el mundo en temas de **arquitectura** y **seguridad** de software.
- Este instituto, adscrito a la **Universidad de Carnegie Mellon**, es el pionero en materia de ingeniería de software y el de **mayor prestigio a nivel mundial**.

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

Diseño de Software

¿Qué es?

Diseño de Software

- “Es el proceso de definir la arquitectura, componentes, interfaces y otras características de un sistema o componente”.

- “Es el resultado de dicho proceso”.

Diseño de Software

- El diseño es tanto un **verbo** como un **sustantivo**.
- Verbo porque es un **proceso**, una **actividad**, es decir una creación.
- El **resultado** del proceso es un diseño en sí, por tanto es una **cosa**, un **sustantivo**.

Design is not just what it looks like and feels like. Design is how it works.

Steve Jobs
American entrepreneur and inventor
(1955-2011)

As a process

Visto como proceso, diseño de software es la actividad del ciclo de vida de ingeniería de software en la que se analizan los requisitos de software, a fin de producir una descripción de la estructura interna del software que servirá de base para su construcción.

As a result

Visto como resultado, el diseño de software describe la arquitectura de software – esto es, cómo es software se descompone y organiza en componentes – así como las interfaces entre dichos componentes.

La descripción debería realizarse a un nivel que permita su construcción.

Actividades

El diseño de software consiste de 2 actividades:

- **Diseño arquitectónico** o simplemente **arquitectura**
  (Diseño de Alto nivel)

- **Diseño detallado** o simplemente **diseño**
  (Diseño de Bajo nivel)

Software Architecture

Una arquitectura de software es el conjunto de estructuras necesarias para razonar sobre el sistema, que involucra elementos de software, relaciones entre ellos y propiedades de ambos.

Conceptos útiles en diferentes niveles de abstracción:

Architectural styles (Architectural Design)

Design Patterns (Detailed Design)

Arquitectura vs Diseño

¿Existe alguna diferencia?

Arquitectura

Diseño Detallado

Granularidad

Diseño de alto nivel vs. Diseño de bajo nivel (detallado)

Arquitectura vs Diseño

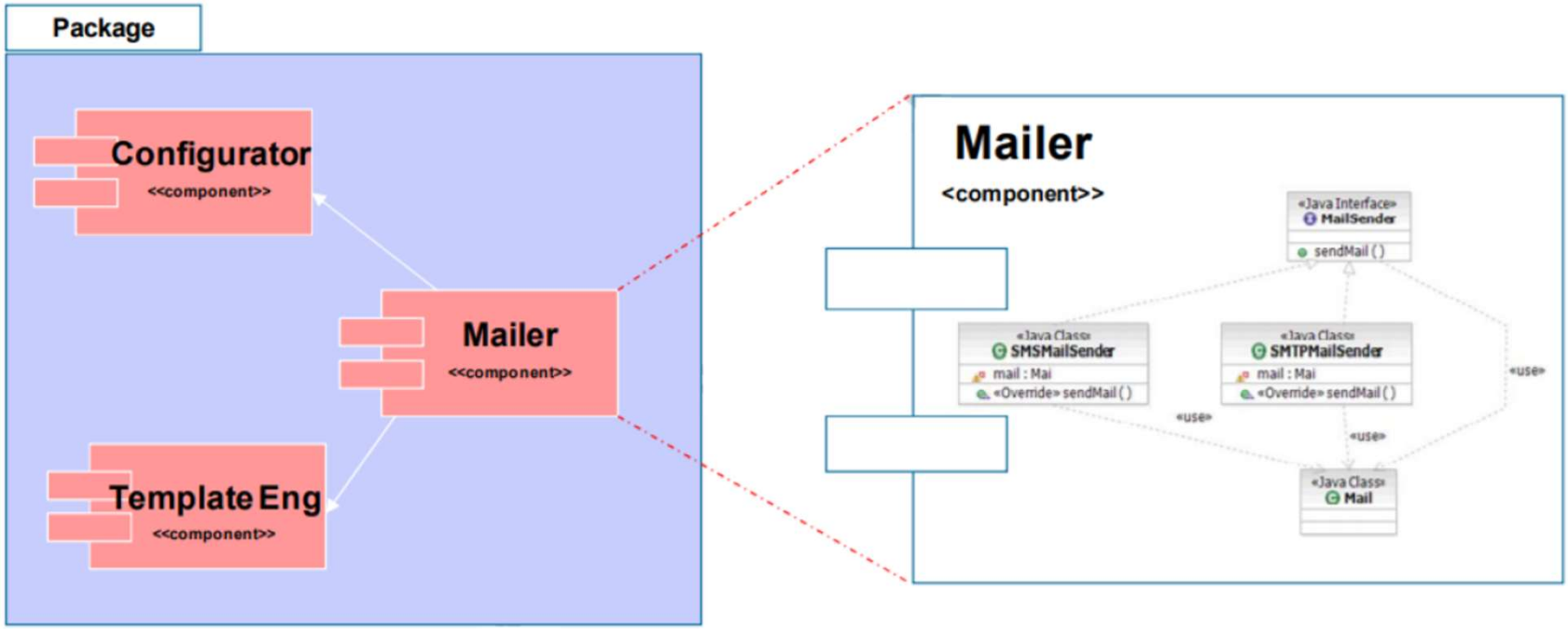
Toda arquitectura es diseño, pero
no todo diseño es arquitectura.

Grady Booch

La arquitectura y el diseño difieren en tres áreas:

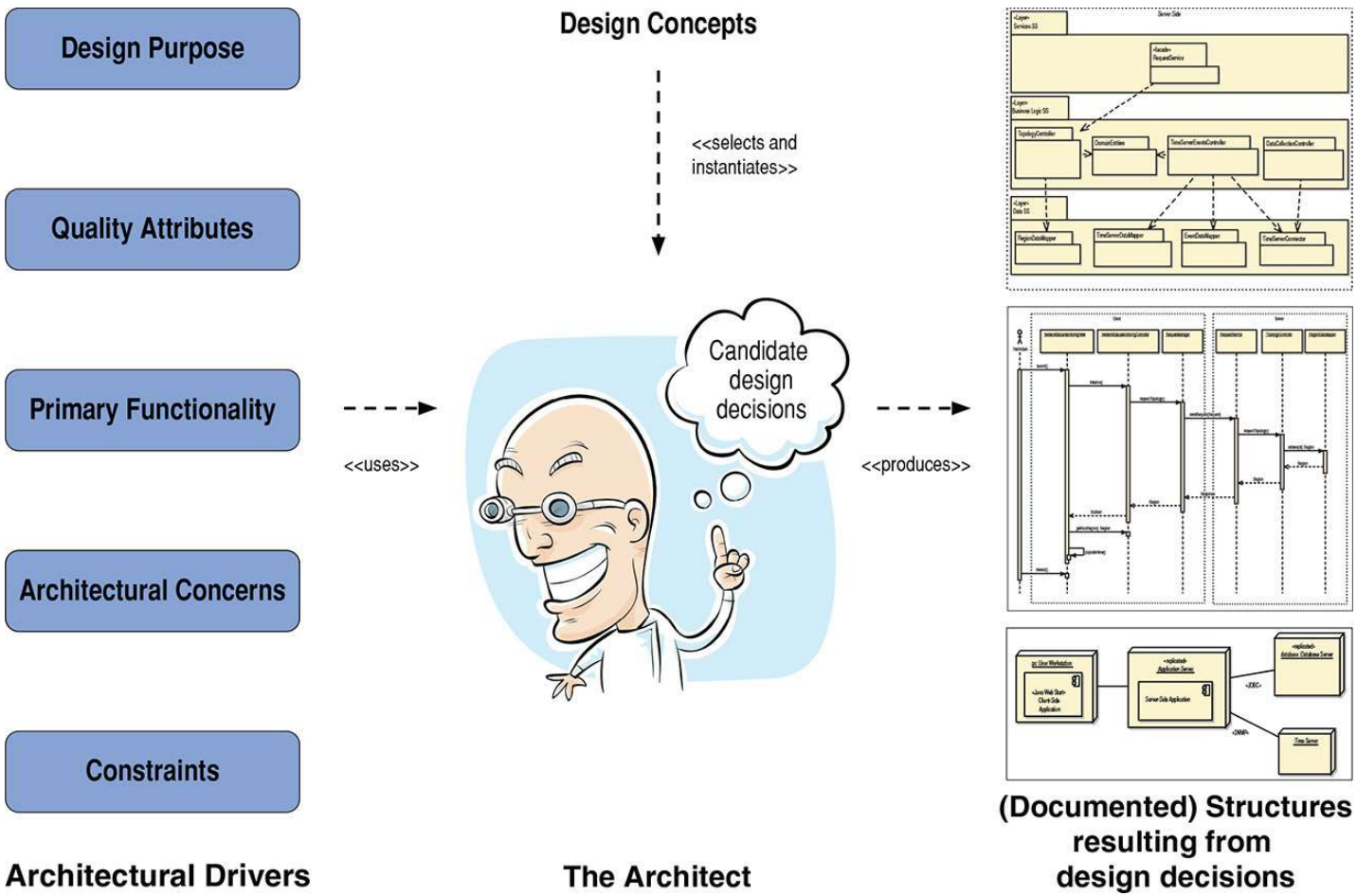
	Arquitectura	Diseño
Nivel de Abstracción	Alto Nivel	Bajo nivel. Enfoque especifico de detalles.
Entregables	Planificación de subsistemas, interfaces con sistemas externos, servicios, componentes reutilizables, prototipo arquitectónico.	Diseño detallado de componentes. Especificación de codificación.
Áreas de Enfoque	Selección de tecnologías, requerimientos no funcionales, manejo de riesgos.	Requerimientos funcionales.

Arquitectura vs Diseño

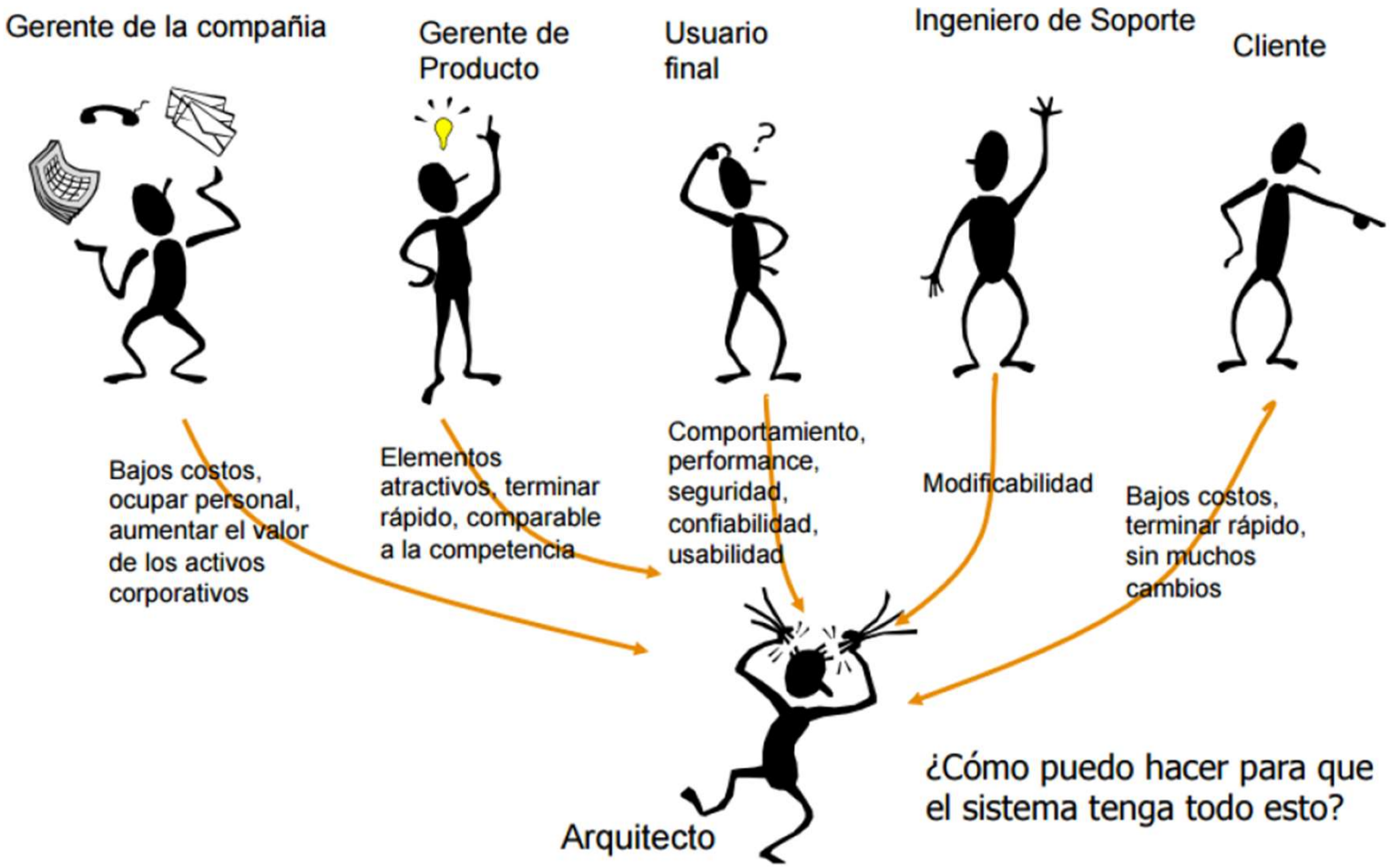


Arquitectura **Diseño**

Diseño Arquitectural



Preocupaciones de los Stakeholders



AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

Principle

"A comprenhensive and fundamental law, doctrine, or assumption"

- Merriam-Webster's Collegiate Dictionary

Software design principles

Nociones clave que proporcionan las bases para diversas aproximaciones y conceptos de diseño de software.

Software design principles

Abstraction

Coupling and cohesion.

Decomposition and modularization.

Encapsulation/information hiding.

Separation of interface and implementation.

Sufficiency, completeness, and primitiveness.

Separation of concerns.

Abstraction

Es una vista de un objeto, la cual se enfoca en la información relevante para un propósito particular e ignora el resto de información.

Mecanismos de abstracción

Parametrización: Se abstrae de los detalles de representación de datos, representando éstos como parámetros nombrados.

Especificación: Tres clases: procedimental, de datos, de control o iteración.

Coupling and Cohesion

Coupling: “A measure of the interdependence among modules in a computer program”.

Cohesion: “A measure of the strength of association of elementos within a module”.

Decomposition and modularization

Descomponer en módulos significa que el software de gran envergadura se divide en un número de componentes más pequeños los cuales se pueden nombrar y que tienen interfaces que describen las interacciones entre dichos componentes.

La meta usualmente es ubicar las funcionalidades y responsabilidades en diferentes componentes.

Encapsulation and information hiding

Agrupar y empaquetar los detalles internos de una abstracción haciéndolos inaccesibles para entidades externas.

Separation of interface and implementation

Separar la interfaz de la implementación implica definir un componente especificando una interfaz pública (conocida por los clientes) que esté separada de los detalles de cómo están implementados dicho componente.

Sufficiency, completeness, and primitiveness

Alcanzar suficiencia y completez significa asegurarse que un componente de software captura todas las características importantes de una abstracción y nada más. Primitivismo significa que el diseño debería basarse en patrones que sean fáciles de implementar.

Separation of concerns

Un *concern* es un área de interés con respecto al diseño de un software que es relevante para uno o más de sus stakeholders. Cada vista de la arquitectura enmarca uno o más concerns. Separar los concerns en vistas permite que los stakeholders interesados se enfoquen en un conjunto reducido de cosas en un momento y ofrece un medio para administrar la complejidad.

SOLID software design principles

	Principle	Description
S	Single responsibility	Una clase debería tener una y solo una razón para cambiar, lo cual quiere decir que debería tener solo una función.
O	Open/closed	Los objetos de software deberían ser abiertos a la extension pero cerrados a la modificación.
L	Listkov substitution	Los objetos del mismo tipo deberían poder reemplazarse por otros de la misma categoría sin alterar la función del programa.
I	Interface segregation	Ningún cliente debería ser forzado a depender de métodos que no usa. Las interfaces del programa siempre deberían mantenerse pequeñas y separadas entre sí.
D	Dependency inversion	Los módulos de alto nivel no deberían depender de los módulos de bajo nivel, sin embargo ambos deberían depender de abstracciones.

Key Issues

Quality concerns (i.e. Performance, security, reliability, usability)

How to decompose, organize, and package software components.

Concurrency.

Control and Handling of Events.

Data Persistence.

Distribution of Components.

Error and Exception Handling, and Fault Tolerance.

Interaction and Presentation.

Security.

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

DISEÑO DE SOFTWARE

PRINCIPIOS

ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

Software Design Strategies and Methods

General Strategies

Divide-and-conquer, stepwise refinement,

top-down vs. bottom-up,

heuristics, patterns, pattern language based,

Iterative and incremental approach.

Software Design Strategies and Methods

Function-Oriented (Structured) Design

Object-Oriented Design

Data Structure-Centered Design

Component-Based Design (CBD)

Aspect-oriented Design

AGENDA

INGENIERÍA DE SOFTWARE

SWEBOK V3

SEI

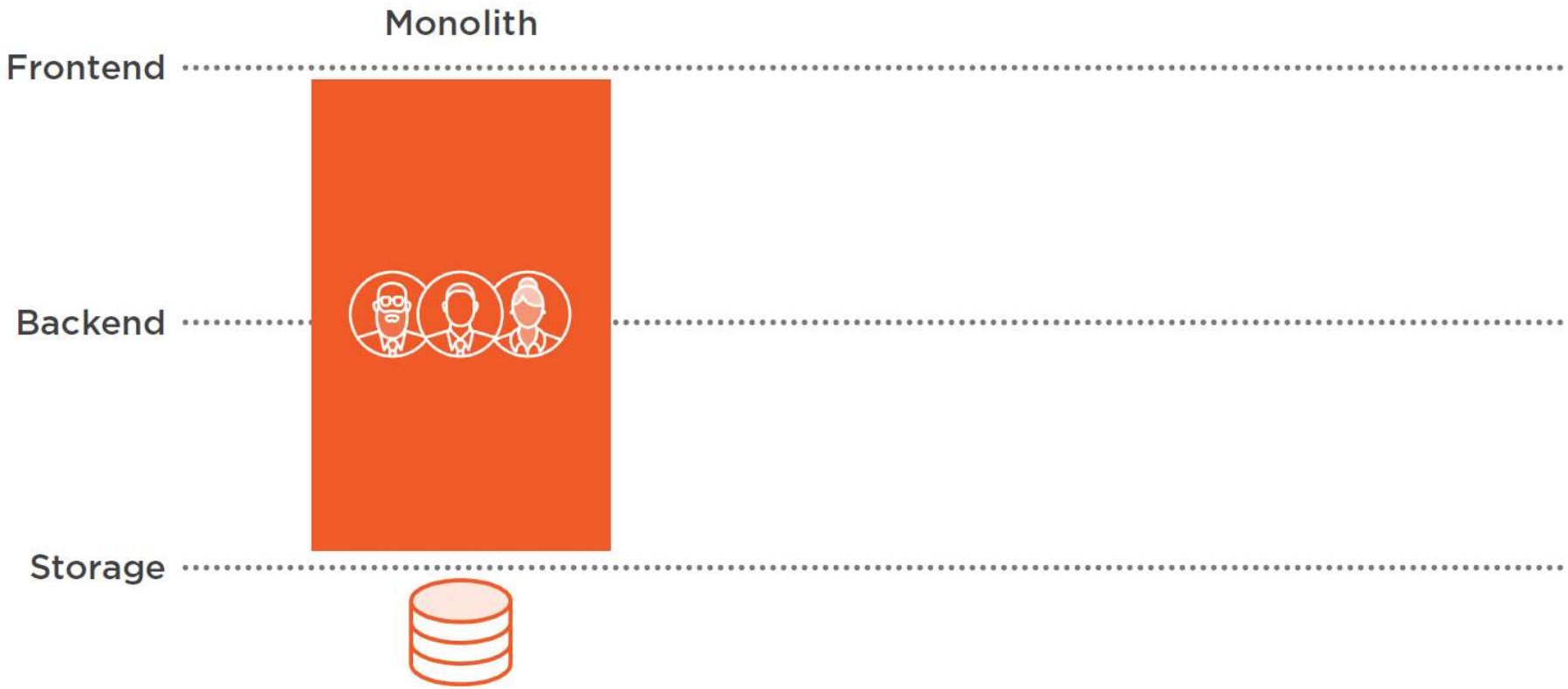
DISEÑO DE SOFTWARE

PRINCIPIOS

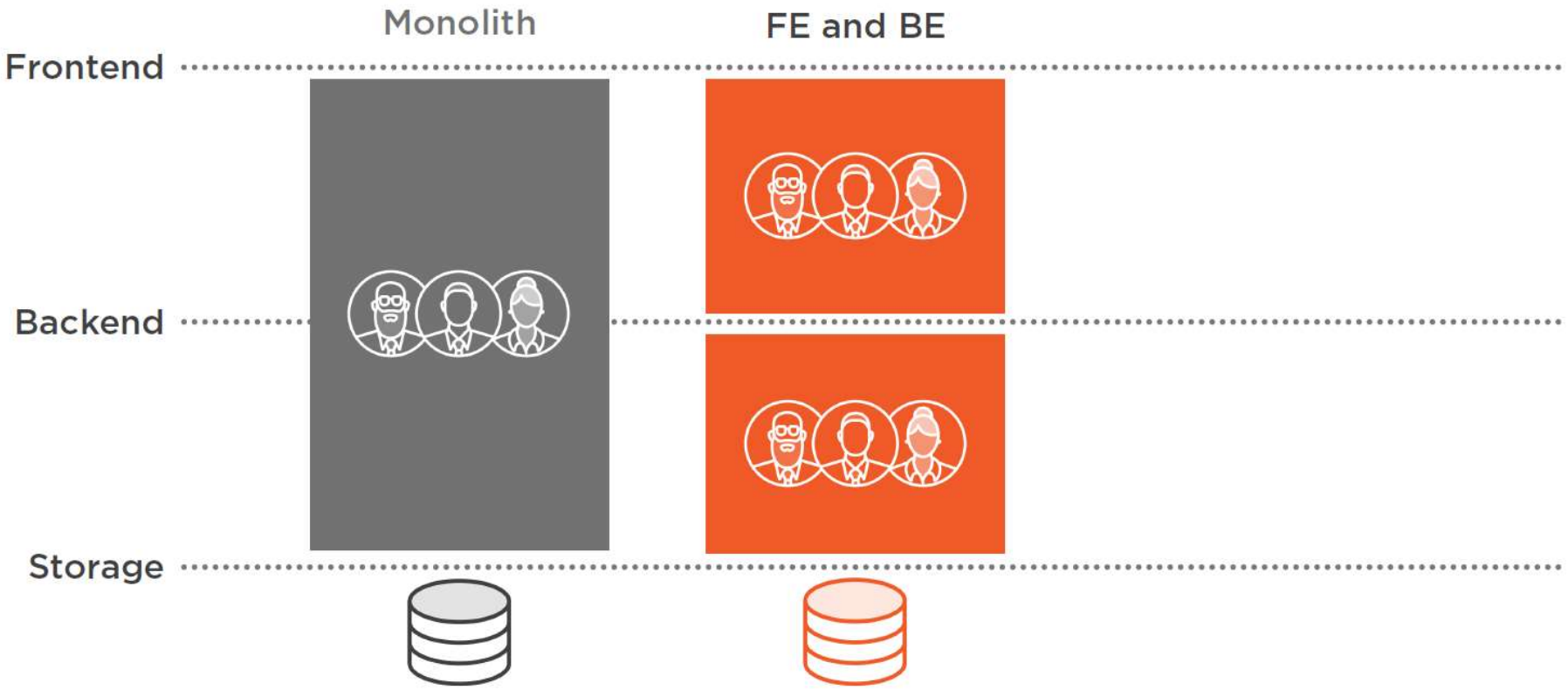
ESTRATEGIAS

EVOLUCIÓN DE ESTRUCTURA DE APLICACIONES

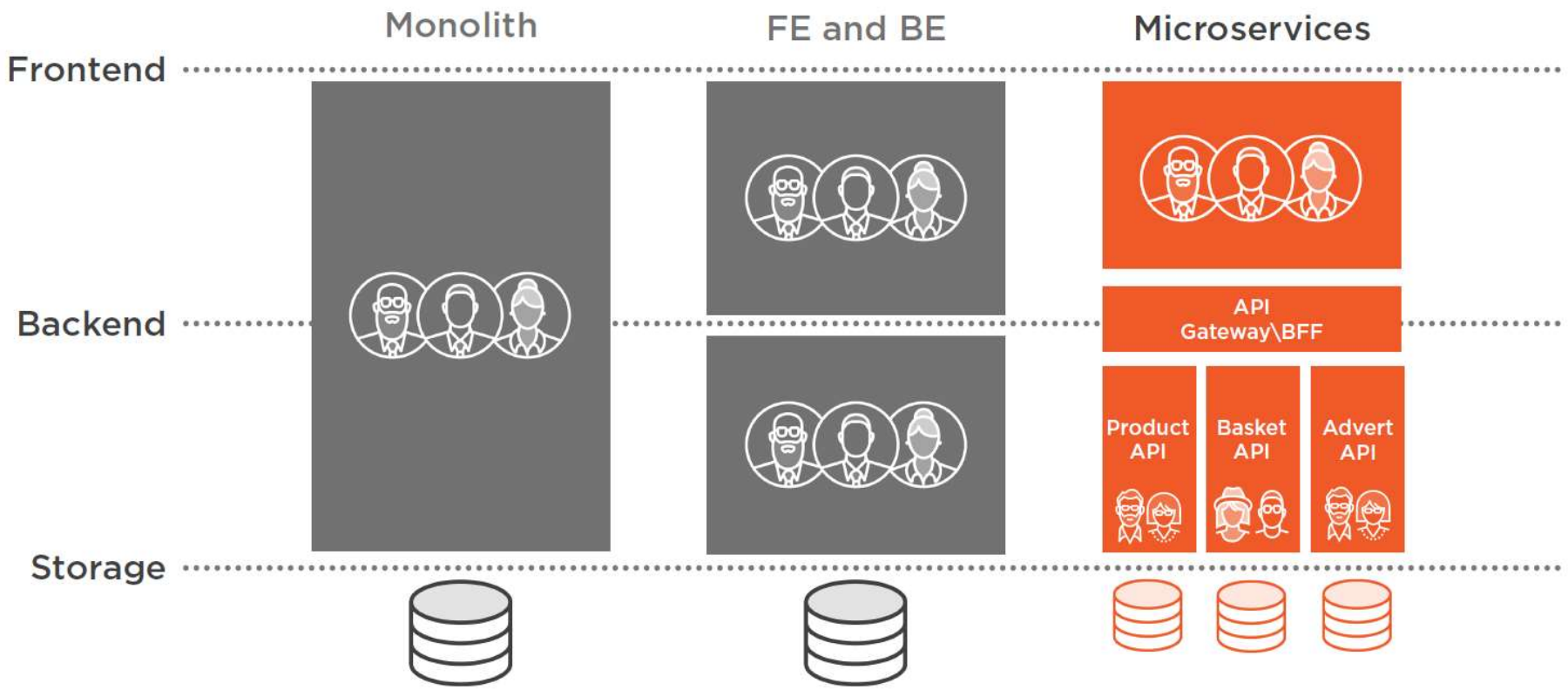
Evolución de la Estructura de Aplicaciones



Evolución de la Estructura de Aplicaciones



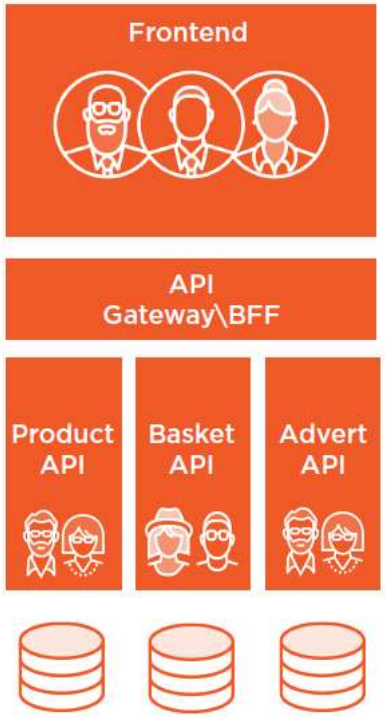
Evolución de la Estructura de Aplicaciones



Monolith Frontends

Monolith

Frontends



Scaling issues

- Frontend application
- Frontend team

Communication issues

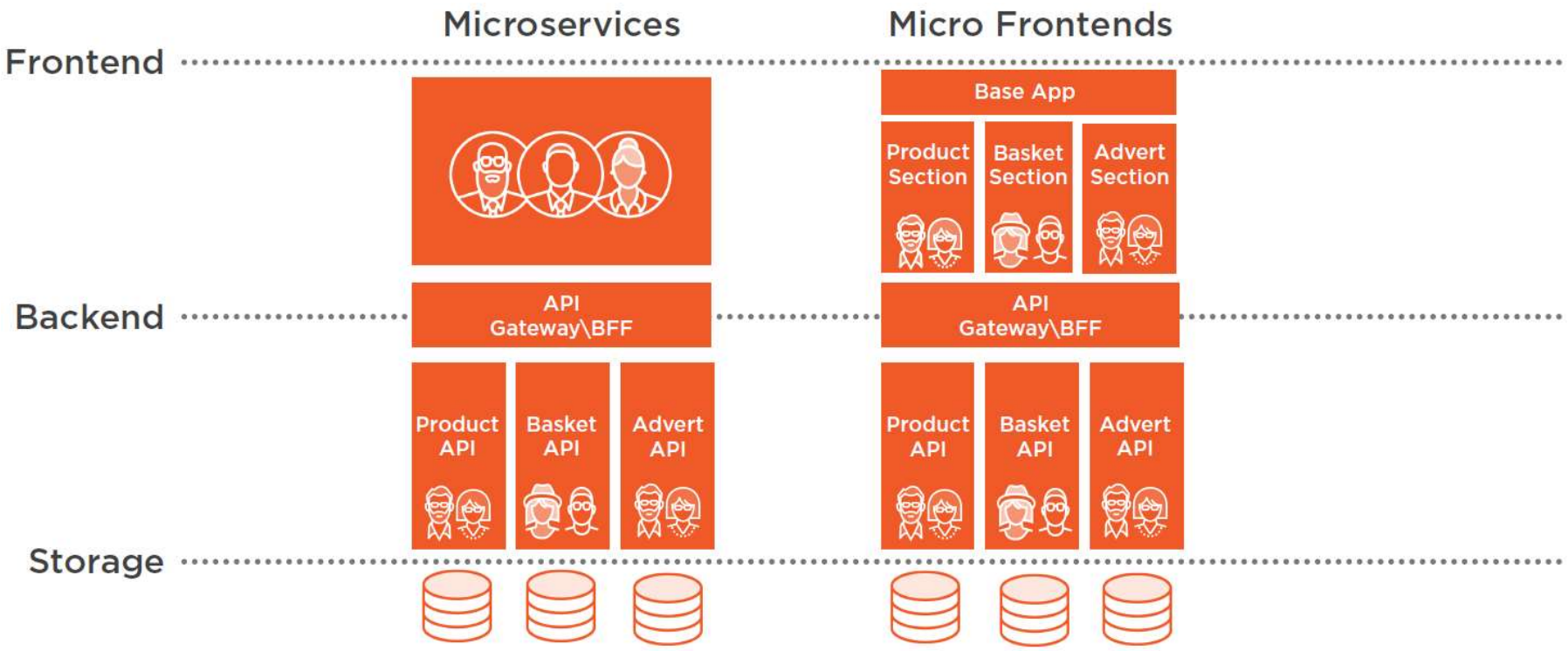
- Multiple teams for one feature change
- Exhausts time
- Backend teams are not customer focused

Code and testing complexity

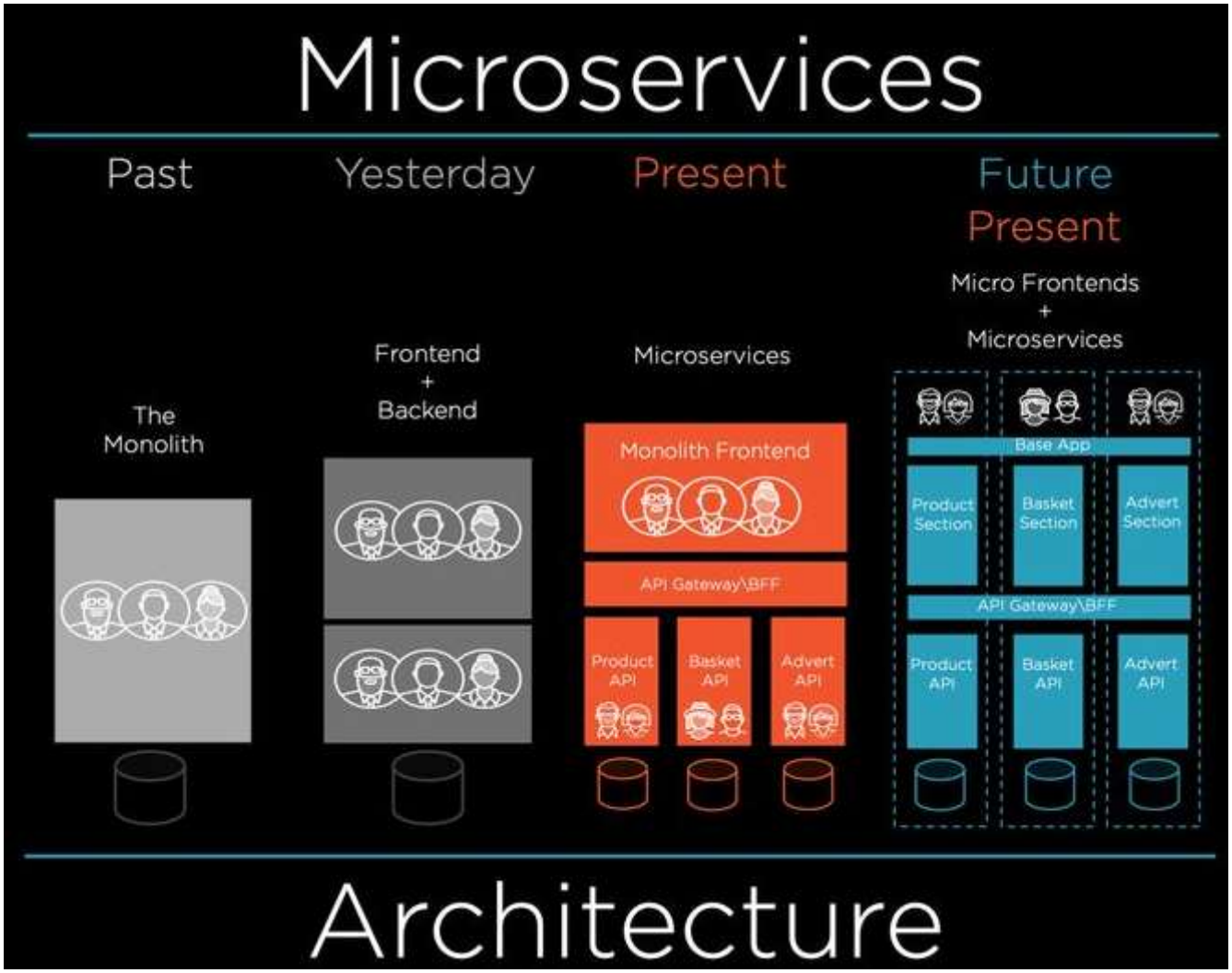
- Increased risk
- Slows continuous delivery

Advantages of monolith frontends

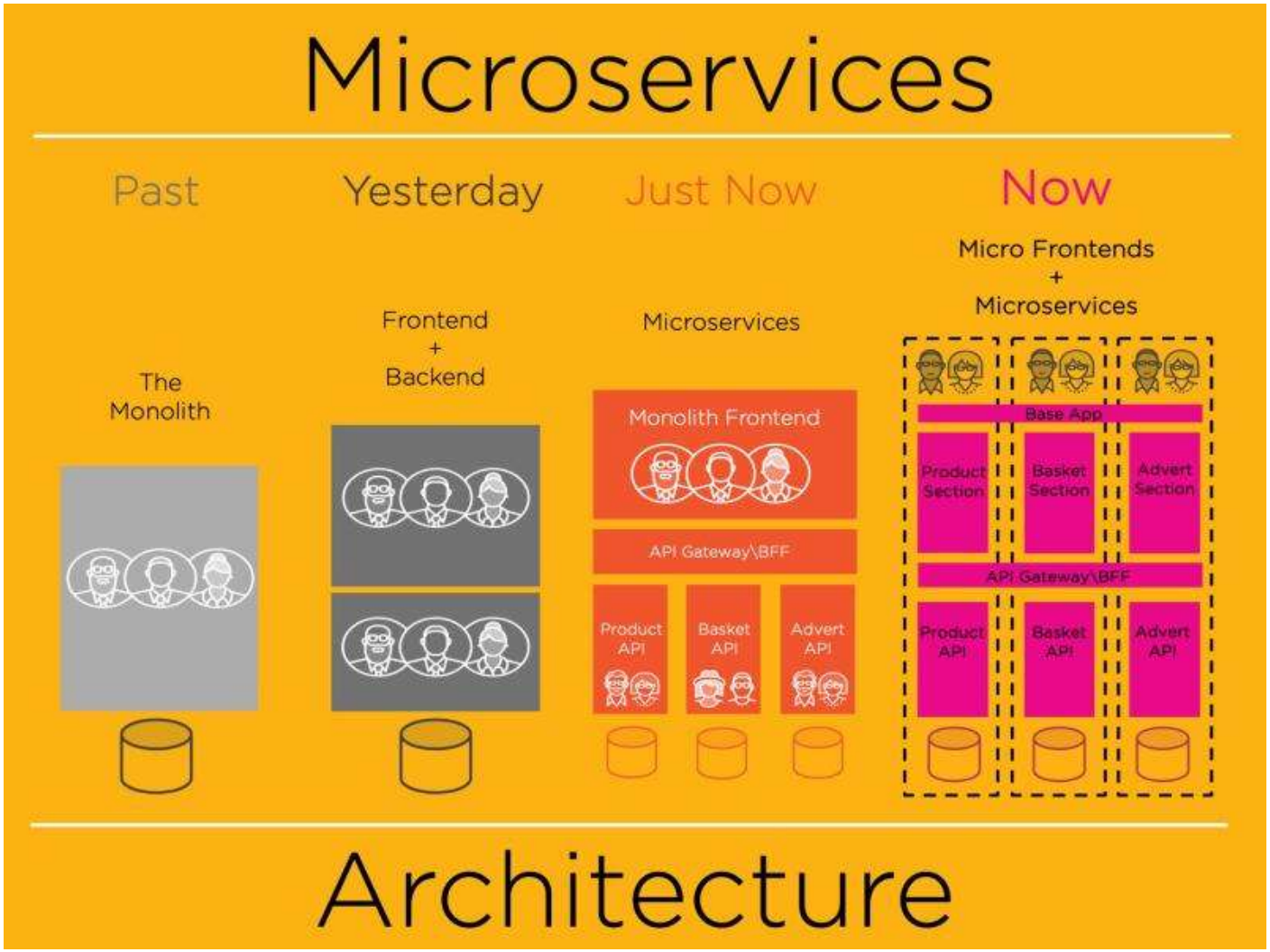
Micro Frontends



Micro Frontends + Microservices



Micro Frontends + Microservices



RESUMEN

Recordemos

Ingeniería de Software,

SWEBOK, SEI

Diseño de Software

Arquitectura vs Diseño

Principios de Diseño, SOLID

Estrategias

Evolución de Estructura de Aplicaciones

REFERENCIAS

Para profundizar

https://www.computer.org/education/bodies-of-knowledge/software-engineering

PREGRADO

Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova