UNIVERSIDAD DE LIMA

60 años

DISEÑO DEL SISTEMA Y ARQUITECTURA DEL SOFTWARE

UNIDAD 1: DISEÑO ARQUITECTÓNICO

Temario

- Métodos de Ingeniería de Software.
- Métodos formales.
- Principios de diseño.
- Proceso del diseño de software.
- Proceso de diseño de la arquitectura.
- Estilos de arquitectura.
- Software de gestión de versiones.

UNIVERSIDAD DE LIMA | 60 años

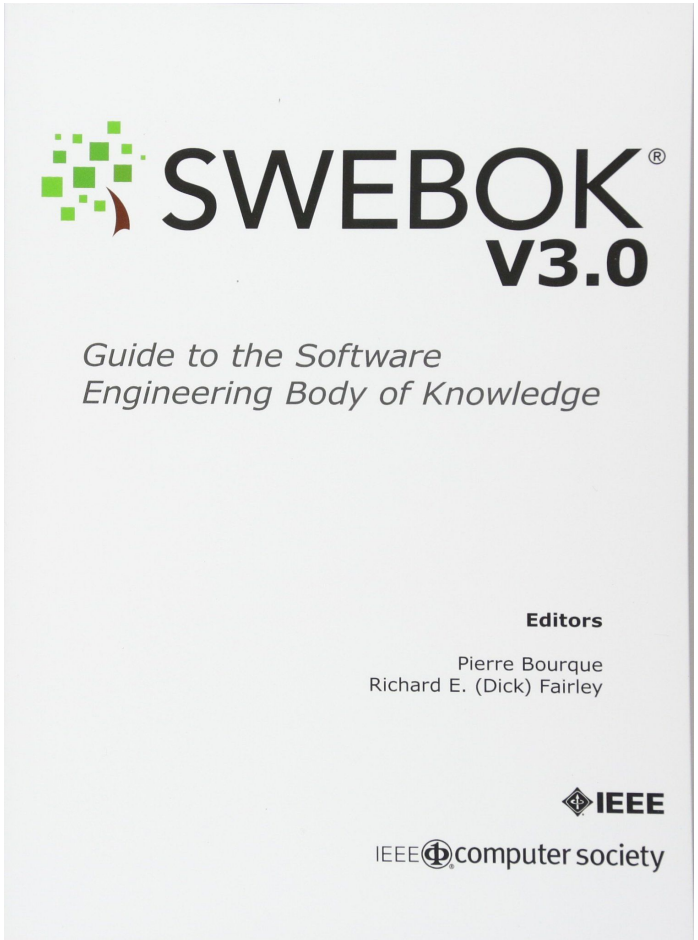
Métodos de Ingeniería de Software

Proporcionan un enfoque organizado y sistemático para desarrollar software.

Métodos de Ingeniería de Software

- Métodos heurísticos
- Métodos formales
- Métodos basado en prototipos
- Métodos ágiles

Pag. 9-7 SWEBOK® Guide V3.0



UNIVERSIDAD DE LIMA | 60 años

Métodos formales

Utilizados para especificar, desarrollar y verificar el software mediante la aplicación de una notación y lenguaje matemático riguroso.

Métodos formales

- Specification Languages
- Program Refinement and Derivation
- Formal Verification
- Logical Inference

Pag. 9-7 SWEBOK® Guide V3.0

Formal Methods: Practice and Experience

JIM WOODCOCK
University of York
PETER GORM LARSEN
Engineering College of Aarhus
JUAN BICARREGUI
STFC Rutherford Appleton Laboratory
and
JOHN FITZGERALD
Newcastle University

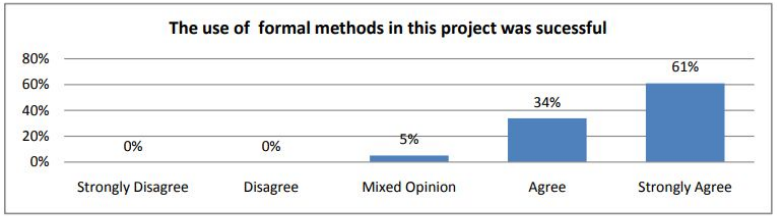
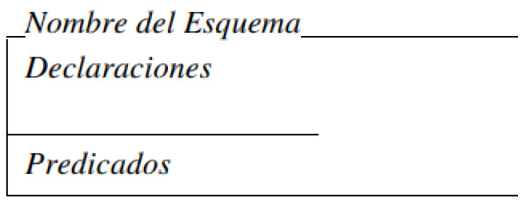
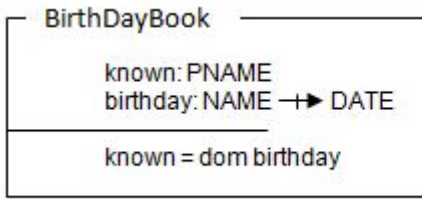
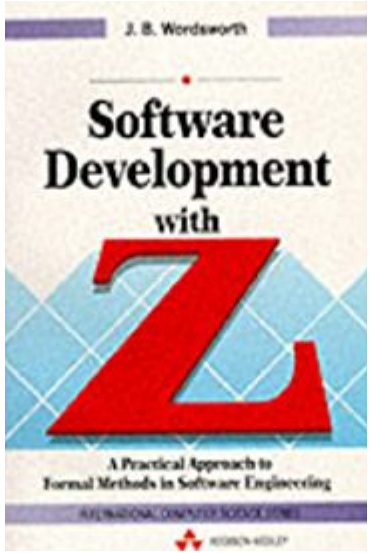


Fig. 7. Overall satisfaction with the used of formal techniques

http://dx.doi.org/10.1145/1592434.1592436







UNIVERSIDAD DE LIMA | 60 años

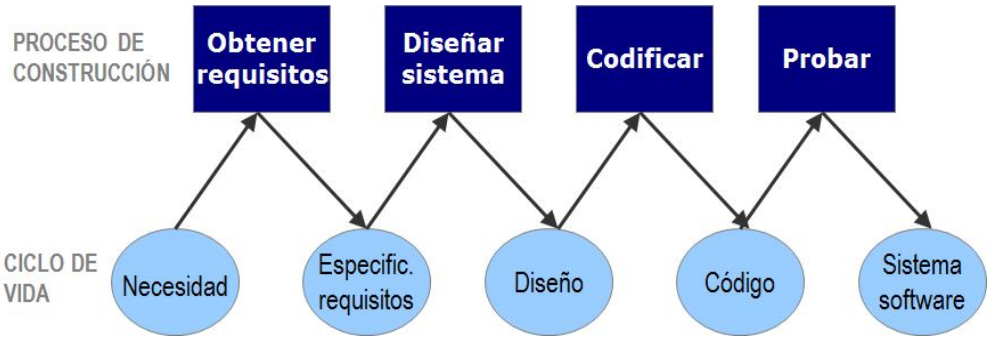
Diseño del Software

El diseño puede ser visto como una forma de resolución de problemas.

Diseño del Software

El diseño se define como "el proceso de definición de la arquitectura, los componentes, las interfaces y otras características de un sistema o componente" y "el resultado de [ese] proceso".
Dos enfoques:

- **El enfoque ágil**: el principal artefacto de diseño es el código mismo.
- **El enfoque "Plan & Document**:, es una etapa que generará artefactos que servirán en la implementación.



UNIVERSIDAD DE LIMA | 60 años

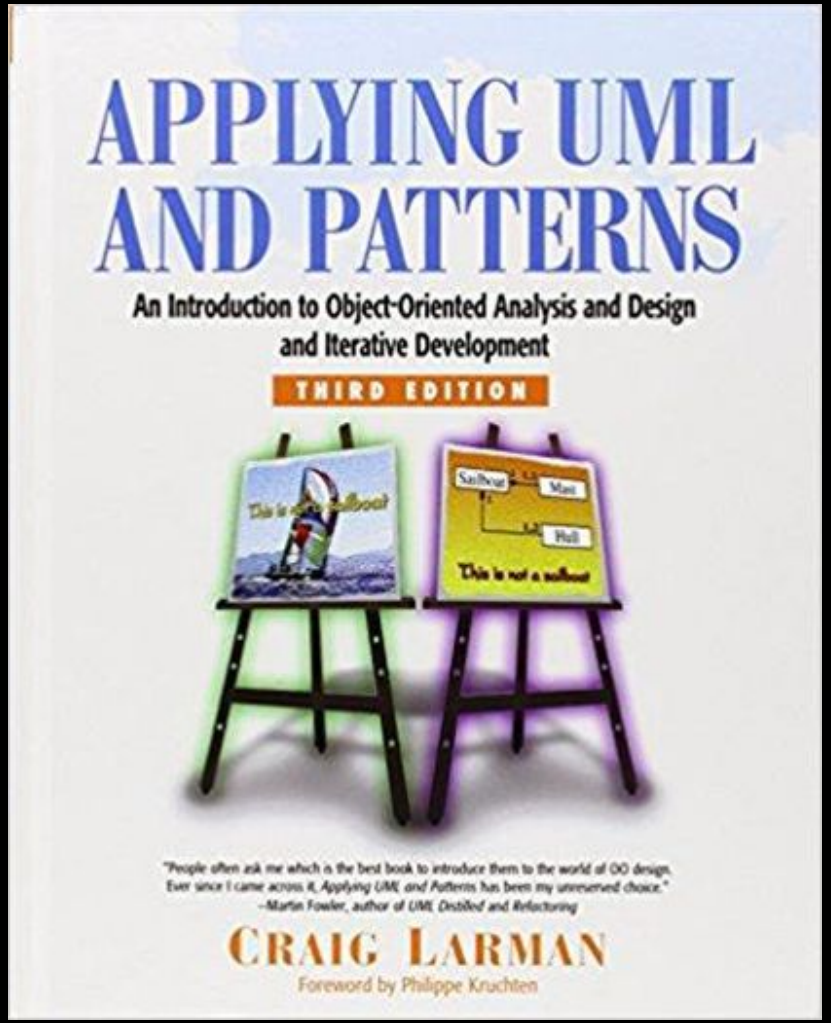
Principios de diseño

Los principios de diseño establecen
una filosofía general que guía el trabajo de diseño que debe ejecutarse.

Principios de diseño de software

Un principio de diseño representa una guía altamente recomendada para dar forma a la lógica de la solución de cierta manera y con ciertos objetivos en mente. Son la caja de herramientas del arquitecto. Guían al arquitecto ante un dilema de diseño.

- Abstracción
- Acoplamiento y cohesión
- Descomposición y modularización
- Encapsulación/ocultación de información
- Separación de Interfaz e Implementación
- Suficiencia y Completitud
- Separación de intereses



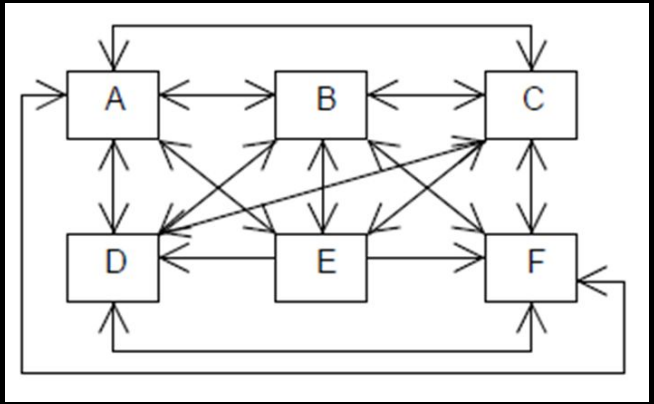
Patrones GRASP

Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 2003

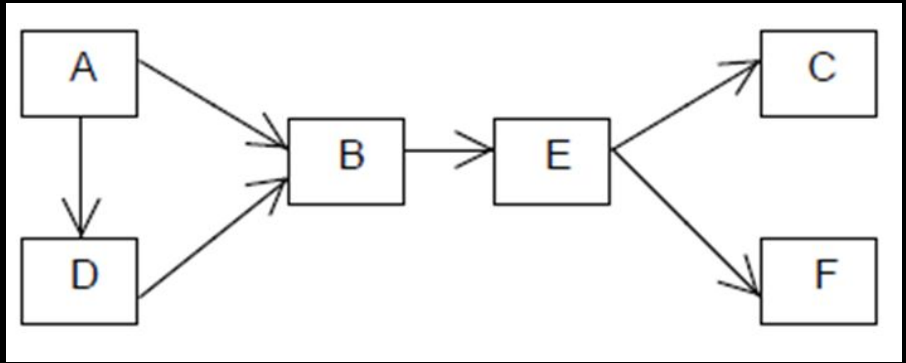
UNIVERSIDAD DE LIMA

60 años

Alto Acoplamiento



Bajo Acoplamiento



Principio: Bajo acoplamiento

¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?

Baja cohesión

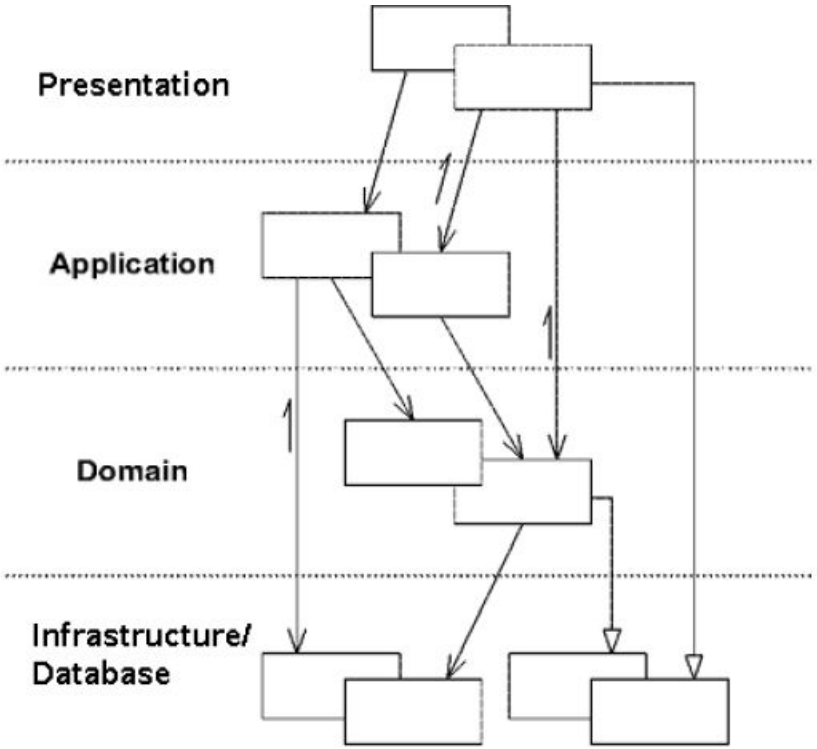
Alta cohesión

Principio: Alta cohesión

¿Cómo mantener la complejidad manejable?

Asignar responsabilidades para que la cohesión se mantenga alta.

Ejemplo - Arquitectura por Capas

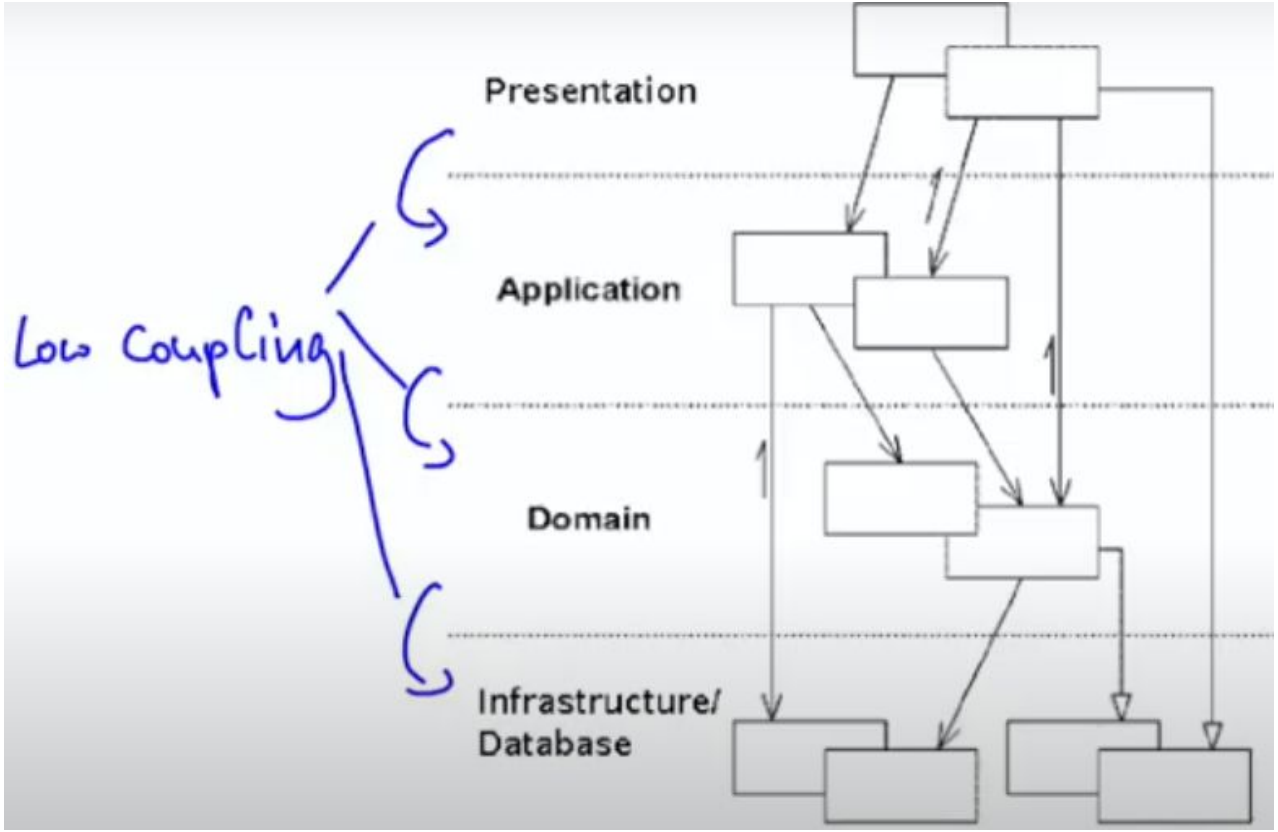


Alto Acoplamiento

Eric Evans, Domain Driven Design, Addison-Wesley, 2004

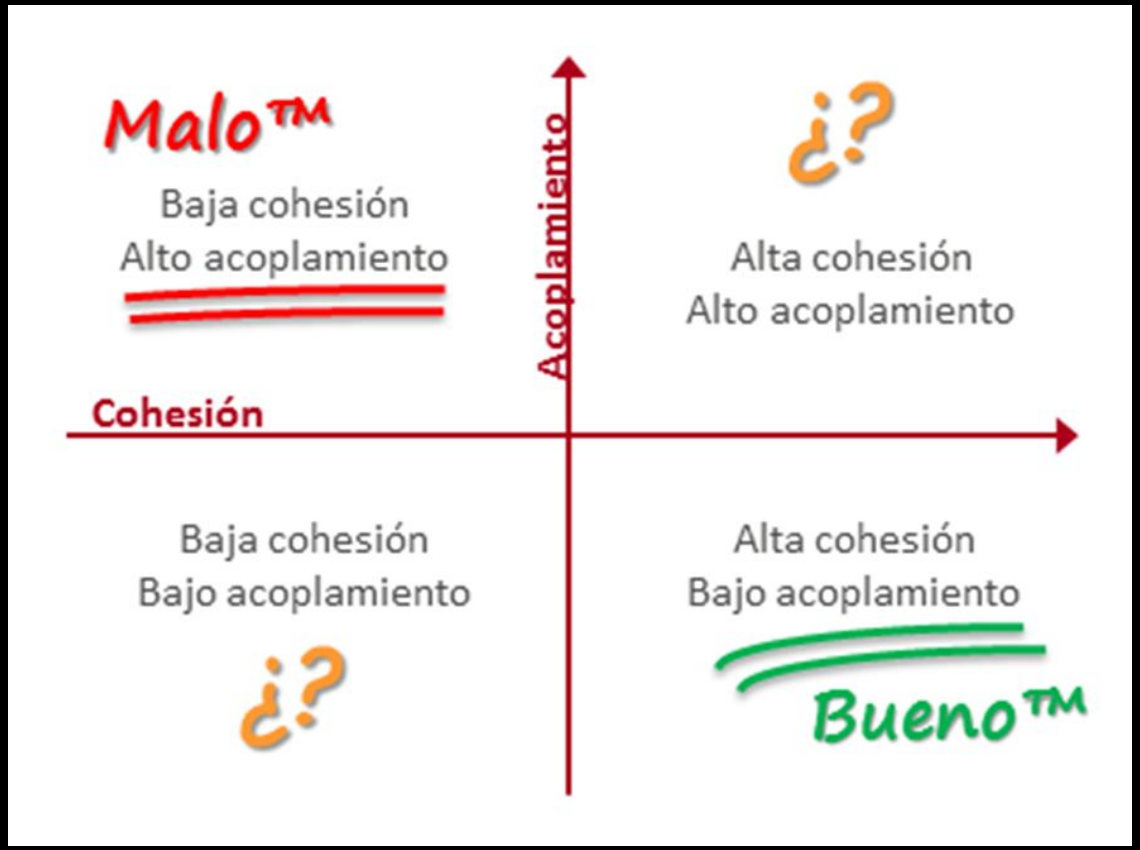
UNIVERSIDAD DE LIMA

Ejemplo - Arquitectura por Capas



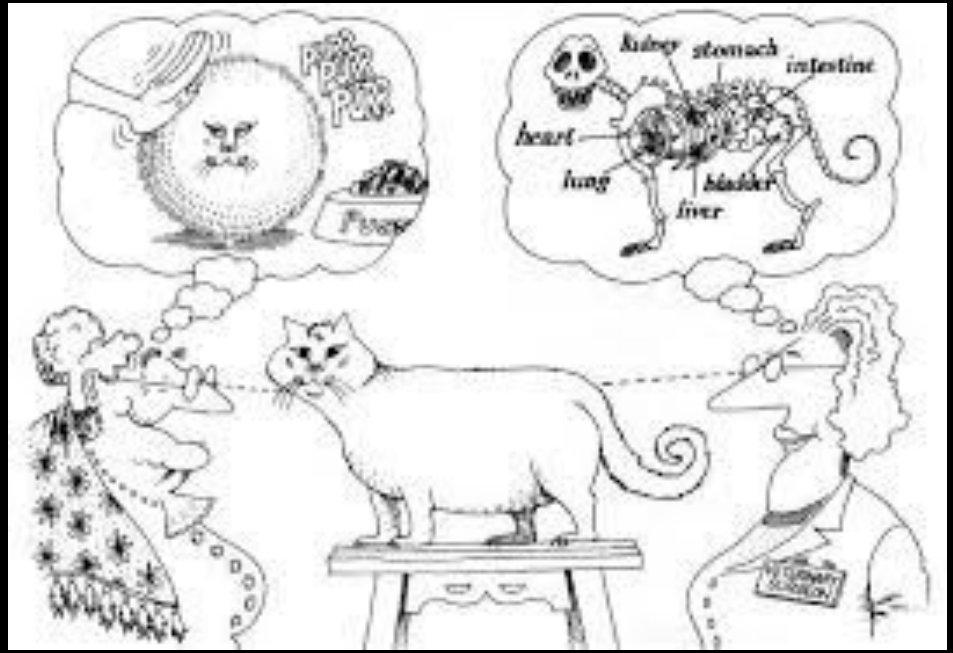
Bajo Acoplamiento

UNIVERSIDAD DE LIMA



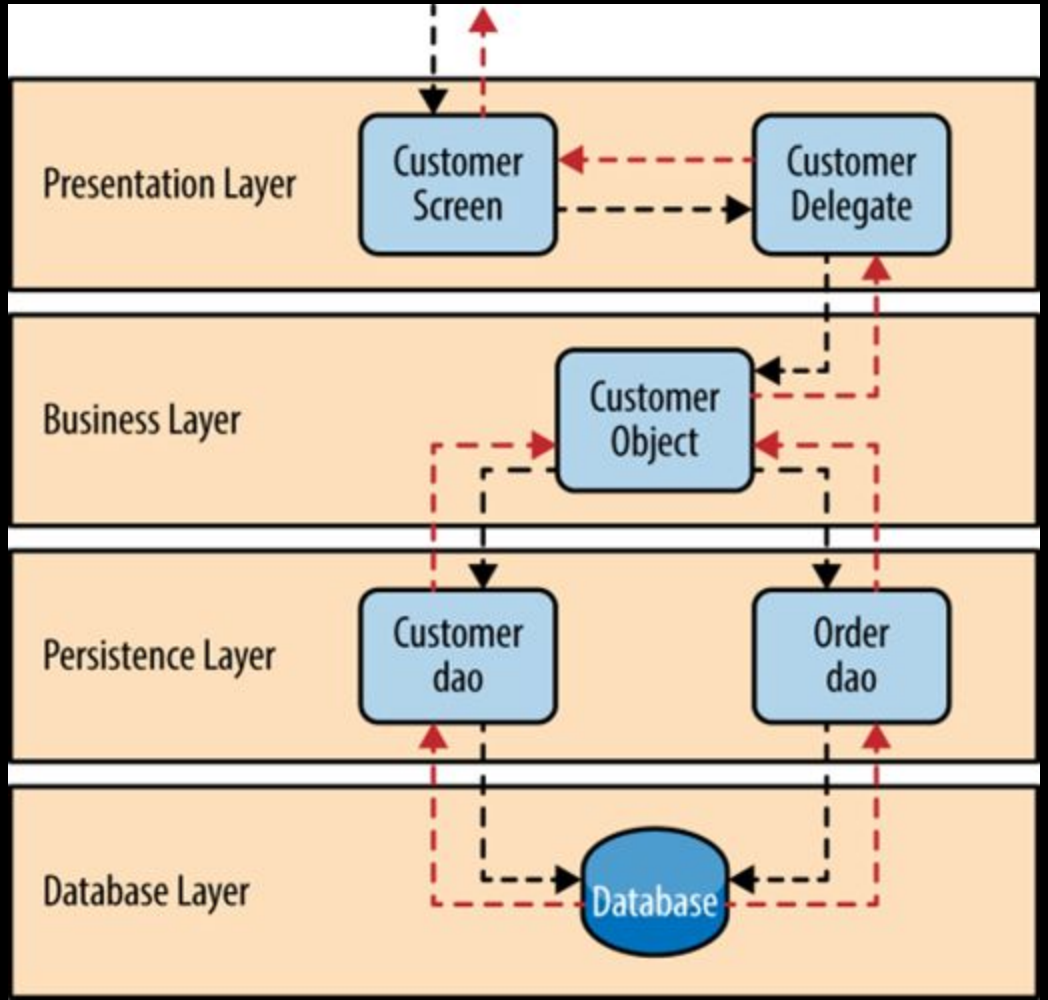
Cohesión vs Acoplamiento

Lo ideal es una alta cohesión y bajo acoplamiento



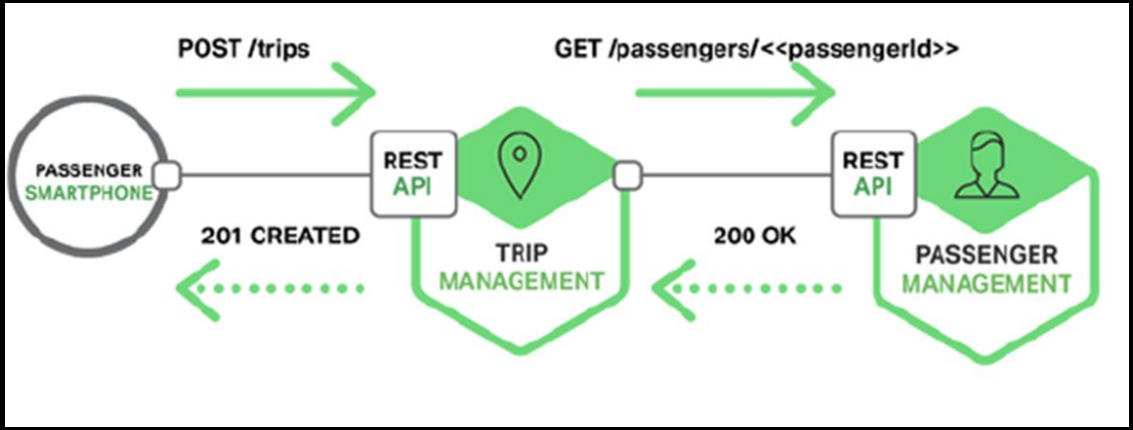
Abstracción

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



Principio: Separación de intereses

La aplicación debe dividirse en función de los tipos de trabajo que realiza.



Principio: Encapsulación

Las diferentes partes de una aplicación deben usar encapsulación para aislarlas de otras partes de la aplicación.

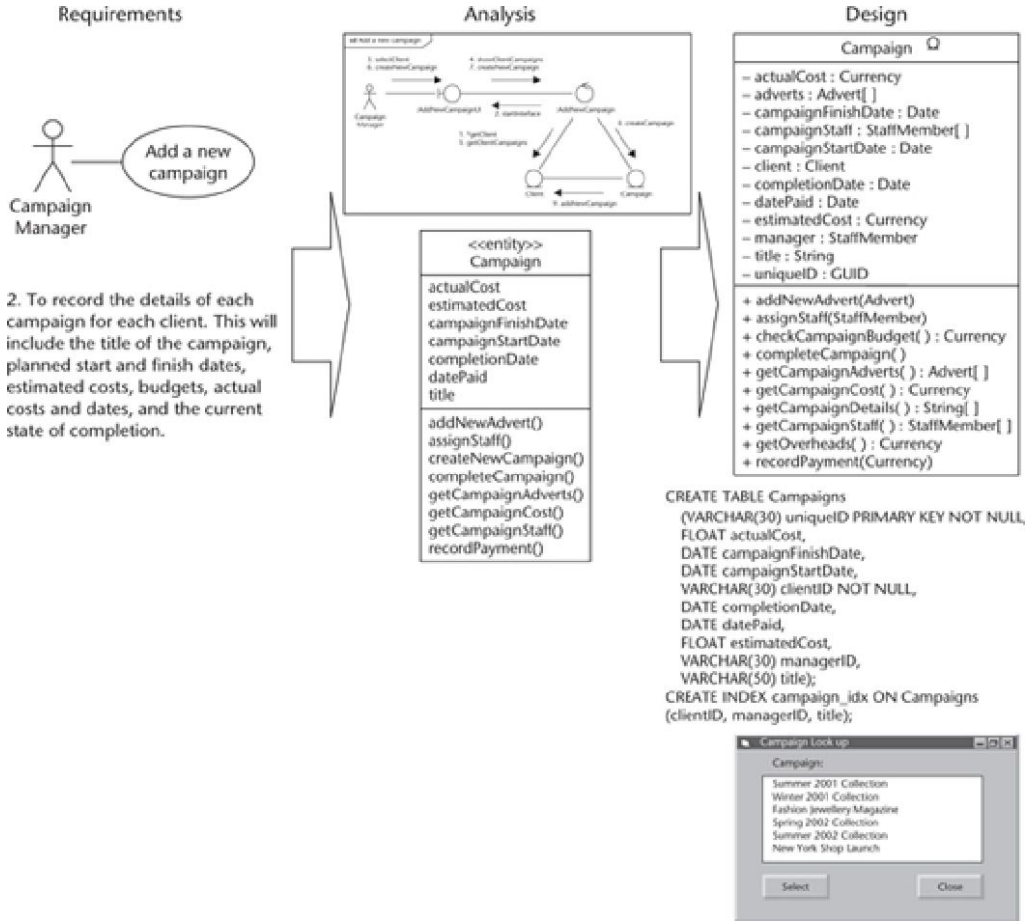
UNIVERSIDAD DE LIMA | 60 años

Proceso del diseño de software

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

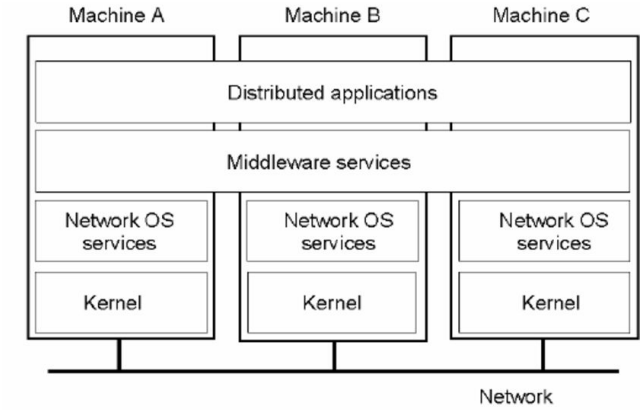
Pasando al diseño: Diferencia entre análisis y diseño

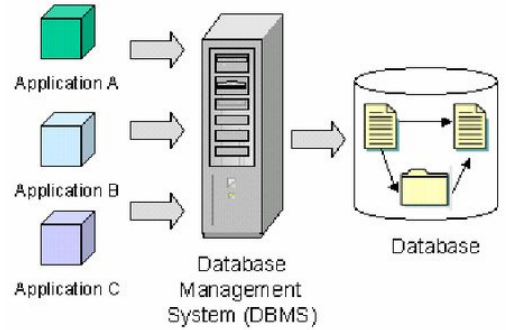
- En el análisis nos enfocamos en el ¿qué es lo que va a realizar nuestro producto? En el diseño nos enfocamos en el ¿cómo lo va a realizar?

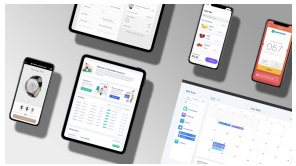


Pasando al diseño: Diferencia entre diseño lógico y físico

- Se debe tomar una decisión sobre el hardware y el software que se utilizarán para desarrollar y entregar el sistema.
- Algunos aspectos del diseño de los sistemas dependen de la elección de la plataforma.
- Estos afectarán la arquitectura del sistema, el diseño de los objetos y las interfaces con varios componentes del sistema.

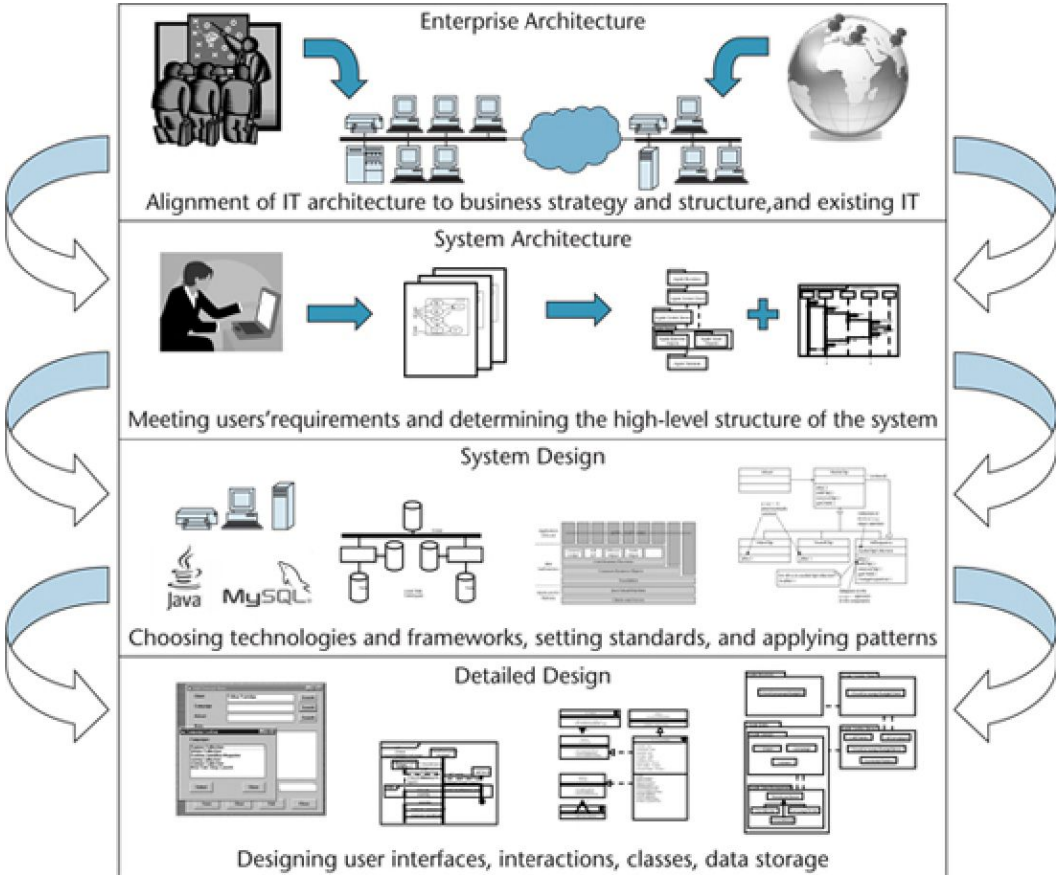






Pasando al diseño: Diferencia entre el diseño del sistema y diseño detallado

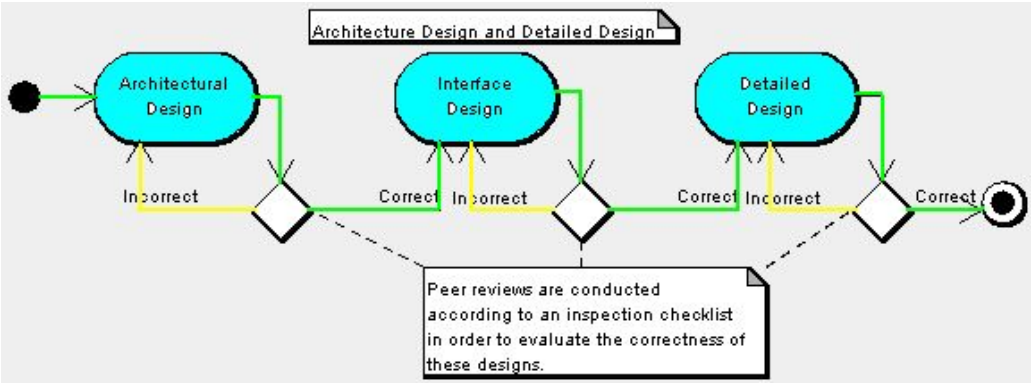
El diseño de sistemas se lleva a cabo en dos niveles: diseño del sistema y diseño detallado. Estas actividades de diseño tienen lugar en el contexto de la arquitectura de la empresa como un todo y la arquitectura del sistema.



Proceso del diseño de software

El diseño de software generalmente se considera un proceso de dos pasos:

- El **diseño arquitectónico** describe cómo se organiza el software en componentes.
- • El **diseño detallado** describe el comportamiento deseado de estos componentes.



https://sce.uhcl.edu/whiteta/sdp/architectureDesignAndDetailedDesign.html#:~:text=Architecture%20design%20is%20more%20abstract,the%20architecture%20that%20is%20specified.

UNIVERSIDAD DE LIMA | 60 años

Proceso de diseño de la arquitectura

El resultado del proceso de diseño de la arquitectura es un modelo de arquitectura que describe cómo el sistema es organizado en un conjunto de componentes que están relacionados

Definiciones de Arquitectura de Softwar

Según: Software Engineering Institute

https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=513807

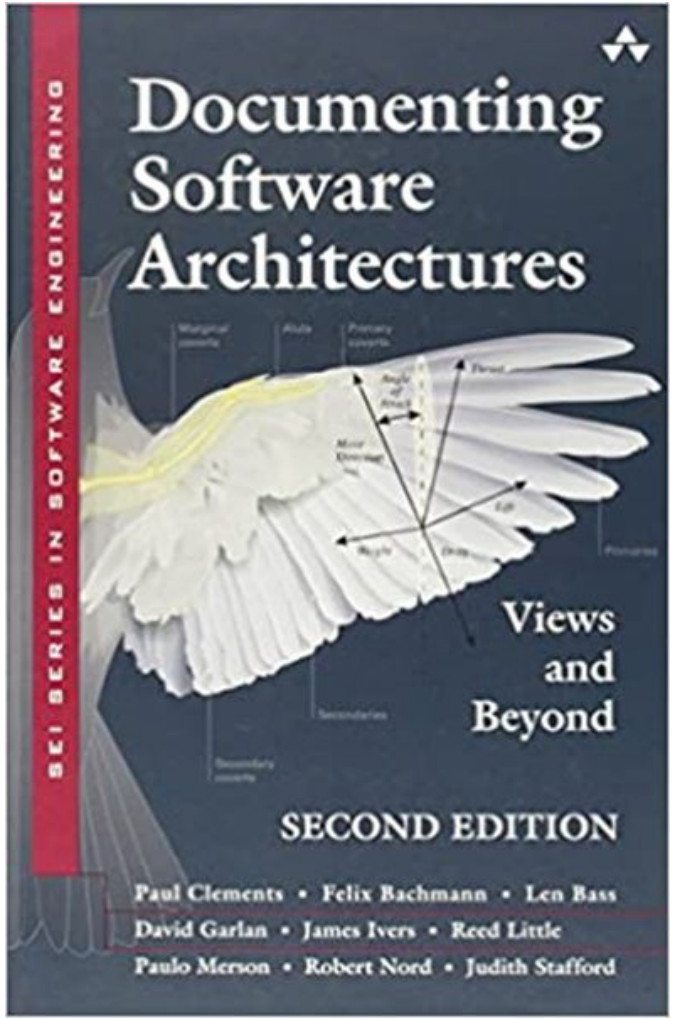
Carnegie Mellon University
Software Engineering Institute

What is your definition of software architecture?

Definiciones de Arquitectura de Software

Según el libro de “Documenting Software Architectures”

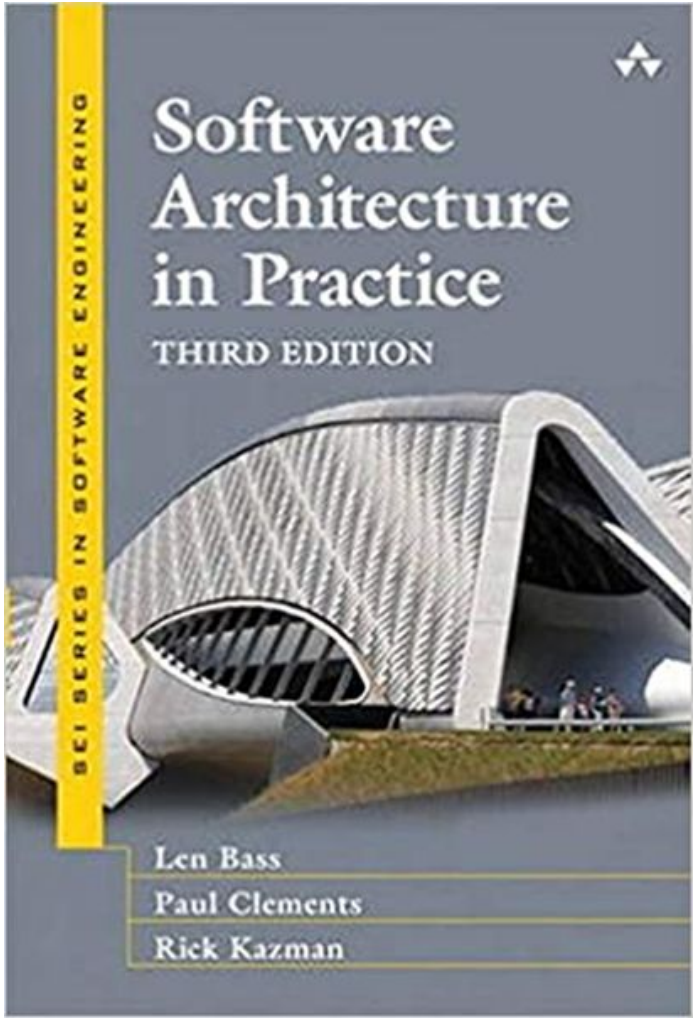
The set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both.



Definiciones de Arquitectura de Software

Según el libro de “Software Architecture in Practice”

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.



Definiciones de Arquitectura de Software

Según el ANSI/IEEE Std1471-2000.

The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution

IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

Sponsor

Software Engineering Standards Committee
of the
IEEE Computer Society

Approved 21 September 2000

IEEE-SA Standards Board

Abstract: This recommended practice addresses the activities of the creation, analysis, and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. A conceptual framework for architectural description is established. The content of an architectural description is defined. Annexes provide the rationale for key concepts and terminology, the relationships to other standards, and examples of usage.
Keywords: architectural description, architecture, software-intensive system, stakeholder concerns, system stakeholder, view, viewpoint

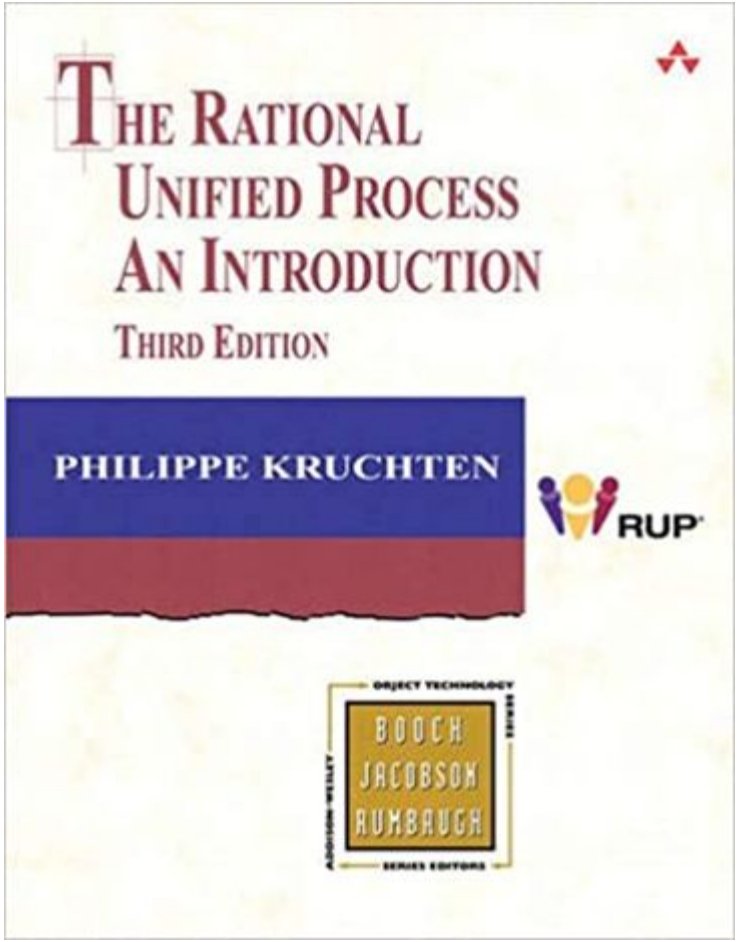
The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-2518-0 SH94869
PDF: ISBN 0-7381-2519-9 SS94869

Definiciones de Arquitectura de Software

Según Rational Unified Process, 1999 - Definición clásica

An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization---these elements and their interfaces, their collaborations, and their composition.



Definiciones de Arquitectura de Software

Según “Arquitectura según ISO/IEC/IEEE 42010:2011”

The architecture of a system constitutes what is essential about that system considered in relation to its environment. There is no single characterization of what is essential or fundamental to a system; that characterization could pertain to any or all of:

- system constituents or elements;
- how system elements are arranged or interrelated;
- principles of the system’s organization or design; and
- principles governing the evolution of the system over its life cycle.


ICS › 35 › 35.080

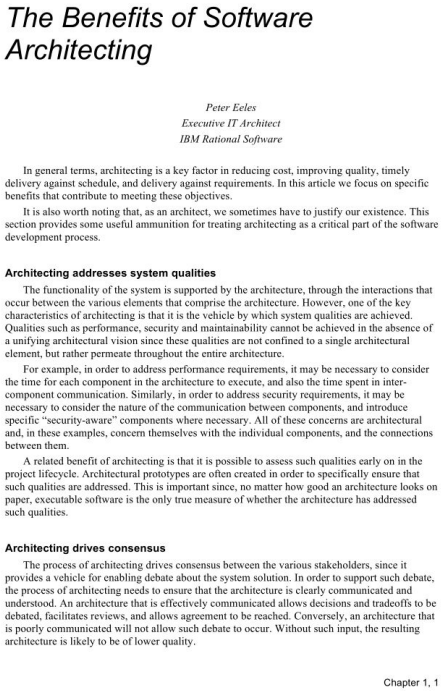
ISO/IEC/IEEE 42010:2011

Systems and software engineering — Architecture description

THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2017. THEREFORE THIS VERSION REMAINS CURRENT.

Beneficios de la arquitectura

La arquitectura satisface los atributos de calidad del sistema. La arquitectura guía el proceso de implementación.



The Benefits of Software Architecting

Peter Eeles
Executive IT Architect
IBM Rational Software

In general terms, architecting is a key factor in reducing cost, improving quality, timely delivery against schedule, and delivery against requirements. In this article we focus on specific benefits that contribute to meeting these objectives.

It is also worth noting that, as an architect, we sometimes have to justify our existence. This section provides some useful ammunition for treating architecting as a critical part of the software development process.

Architecting addresses system qualities

The functionality of the system is supported by the architecture, through the interactions that occur between the various elements that comprise the architecture. However, one of the key characteristics of architecting is that it is the vehicle by which system qualities are achieved. Qualities such as performance, security and maintainability cannot be achieved in the absence of a unifying architectural vision since these qualities are not confined to a single architectural element, but rather permeate throughout the entire architecture.

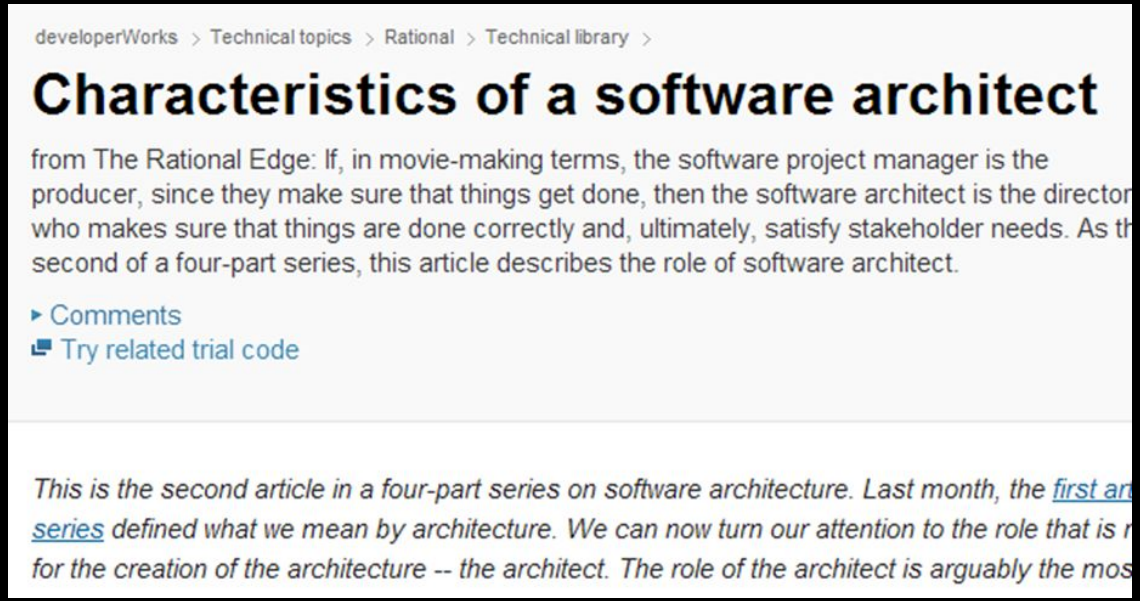
For example, in order to address performance requirements, it may be necessary to consider the time for each component in the architecture to execute, and also the time spent in inter-component communication. Similarly, in order to address security requirements, it may be necessary to consider the nature of the communication between components, and introduce specific "security-aware" components where necessary. All of these concerns are architectural and, in these examples, concern themselves with the individual components, and the connections between them.

A related benefit of architecting is that it is possible to assess such qualities early on in the project lifecycle. Architectural prototypes are often created in order to specifically ensure that such qualities are addressed. This is important since, no matter how good an architecture looks on paper, executable software is the only true measure of whether the architecture has addressed such qualities.

Architecting drives consensus

The process of architecting drives consensus between the various stakeholders, since it provides a vehicle for enabling debate about the system solution. In order to support such debate, the process of architecting needs to ensure that the architecture is clearly communicated and understood. An architecture that is effectively communicated allows decisions and tradeoffs to be debated, facilitates reviews, and allows agreement to be reached. Conversely, an architecture that is poorly communicated will not allow such debate to occur. Without such input, the resulting architecture is likely to be of lower quality.

https://www.researchgate.net/profile/Peter-Eeles-2/publication/330202279_The_Benefits_of_Software_Architecting/links/5c3362eb92851c22a3624ca6/The-Benefits-of-Software-Architecting.pdf



developerWorks > Technical topics > Rational > Technical library >

Characteristics of a software architect

from The Rational Edge: If, in movie-making terms, the software project manager is the producer, since they make sure that things get done, then the software architect is the director who makes sure that things are done correctly and, ultimately, satisfy stakeholder needs. As th second of a four-part series, this article describes the role of software architect.

▸ Comments

Try related trial code

This is the second article in a four-part series on software architecture. Last month, the first ar series defined what we mean by architecture. We can now turn our attention to the role that is r for the creation of the architecture -- the architect. The role of the architect is arguably the mos

El rol del Arquitecto de Software

IEEE defines the term "architect": An architect is the person, team, or organization responsible for systems architecture.

https://research.cs.queensu.ca/home/emads/teaching/readings/CharacteristicsOfASoftwareArchitect.pdf

UNIVERSIDAD DE LIMA

60 años

Requerimientos arquitecturales

Requisito funcional:

- Establecen lo que el sistema debe hacer y cómo debe comportarse o reaccionar ante los estímulos.
- Todo requerimiento funcional se puede descomponer en entradas, lógica de procesamiento y salidas.
- Se verifican a través de pruebas funcionales.

Requisito no funcional

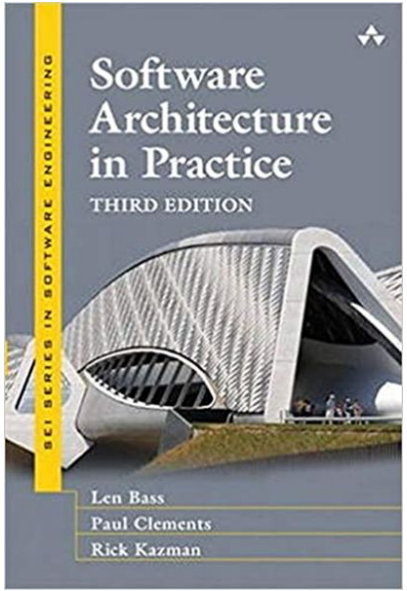
- Requerimientos que se expresan en términos de atributos de calidad(los terminados en –bility).

Requisito no funcional: Quality Attributes:

- including various “-ilities” (maintainability, portability, testability, usability)
- and “-nesses” (correctness, robustness).
- Quality attributes discernible at runtime
  - Performance, security, availability, functionality, usability
- Not discernible at runtime
  - modifiability, portability, reusability, testability)
- architecture’s intrinsic qualities
  - conceptual integrity, correctness, completeness

PART TWO

QUALITY ATTRIBUTES



Requisito no funcional: Escritura estilo tradicional

Atributo: Disponibilidad
"El sistema debe tener una **disponibilidad mensual** de 95 por ciento"

¿Es completo?

"La **disponibilidad** es la cantidad de tiempo que un sistema está funcionando conforme a lo previsto, durante período de tiempo predeterminado."

https://youtu.be/YQV70g9oBFA

Requisito no funcional: Estilo de redacción moderno

Disponibilidad

Cuando en condiciones normales, se produzca una falla irrecuperable en el hardware, debe resultar posible restaurar el sistema a un estado conocido (no anterior a la copia de seguridad del día anterior) en menos de <xx> horas, desde el momento en que el hardware esté disponible.

El enfoque moderno permite encontrar rápidamente la mejor solución para el requisito. Existen catálogos de soluciones.

Escenario 3 - Desempeño

Descripción

Cuando un establecimiento comercial envía una publicación, en los momentos en los que hay mayor tráfico de anuncios (horas pico), las ofertas se hacen visibles para el usuario en la aplicación tras máximo diez (10) segundos de haber sido publicadas.

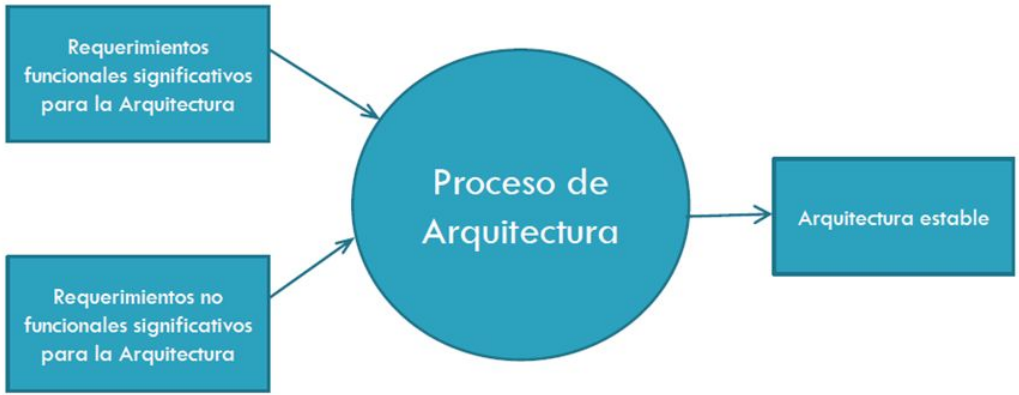
Validación del escenario

- **Origen del estímulo:** Establecimiento comercial.
- **Estímulo:** Enviar una publicación.
- **Entorno:** En los momentos que hay mayor tráfico de anuncios (horas pico).
- **Artefacto:** Las ofertas.
- **Respuesta:** Deben hacerse visibles para el usuario en la aplicación.
- **Medida de la respuesta:** Máximo diez (10) segundos después de haber sido publicadas.

https://sites.google.com/site/wikinower/home/requerimientos-significativos-para-la-arquitectura

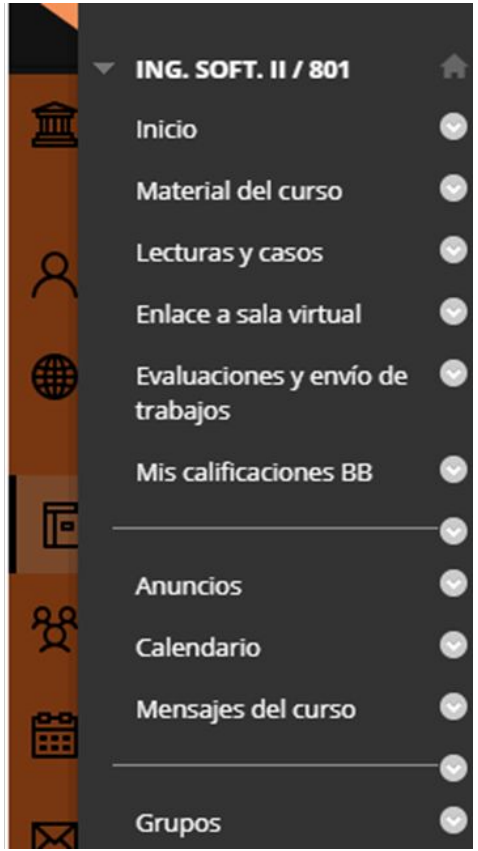
Requisitos significativos para la arquitectura

- Los requisitos arquitectónicamente significativos son aquellos requisitos que juegan un papel importante en la determinación de la arquitectura del sistema.
- Los requisitos arquitectónicamente significativos son un subconjunto de los requisitos que deben cumplirse antes de que la arquitectura se pueda considerar "estable".



Requisitos *Funcionales* significativos para la arquitectura

- Un requerimiento funcional significativo para la arquitectura tiene que tener dos características esenciales:
  - Valor para el negocio.
  - Impacto en los atributos de calidad.
- En inglés
  - Architecturally Significant Requirements (ASR)



¿Cuáles son los Requisitos *Funcionales* significativos para la arquitectura?

¿Quiénes son los interesados claves?

¿Profesor?

¿Alumnos?

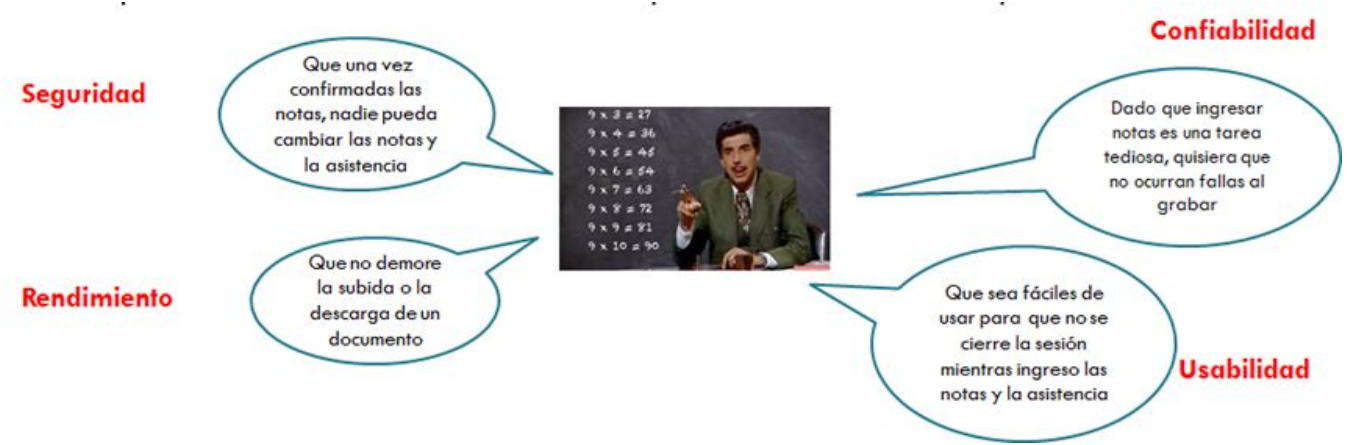
¿Cuáles son los Requisitos *Funcionales* significativos para la arquitectura?

¿Cuáles son los requisitos más usados por el profesor?

¿Profesor?

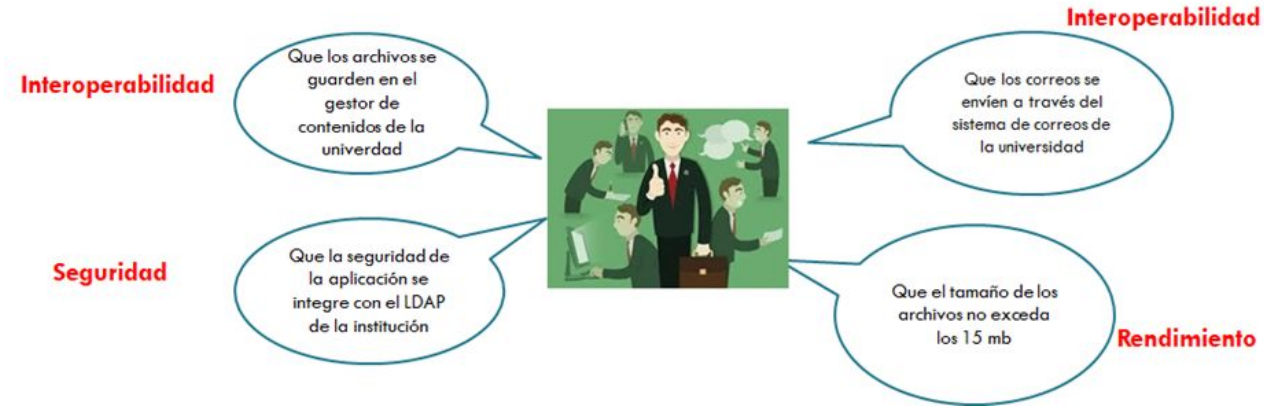
¿Cuáles son los Requisitos *Funcionales* significativos para la arquitectura?

¿Cuáles son los atributos de calidad más valorados por el profesor en relación a los requisitos más usados por él?



¿Cuáles son los Requisitos *Funcionales* significativos para la arquitectura?

¿Cuáles son los atributos de calidad más valorados por el profesor en relación a los requisitos más usados por él?



¿Cuáles son los Requisitos *Funcionales* significativos para la arquitectura?

Podemos escoger entre Registro de notas parciales y de notas finales

	Seguridad	Usabilidad	Rendimiento	Confiabilidad	Interoperabilidad
Registro de notas parciales	x	x		x	
Registro de notas finales	x	x		x	
Registro de asistencia		x		x	
Subir documentos			x		x
Enviar correos	x		x		x

Requisitos NO Funcionales significativos para la arquitectura

Todos los atributos de calidad son significativos para la arquitectura.

Escenario de calidad	#1
Atributo de calidad	Desempeño
Fuente del estímulo	Cuando 1000 profesores de forma concurrente
Estímulo	Suben un archivo de 15 MB
Entorno	Con una velocidad de subida en internet entre 1 y 1.5 Mbps
Artefacto	A través de la página para subir archivos del sistema del aula virtual
Respuesta	El archivo es subido y puesto a disposición de lo estudiantes
Medida de la respuesta	95 % de los archivos están disponibles en menos de 5 segundos.

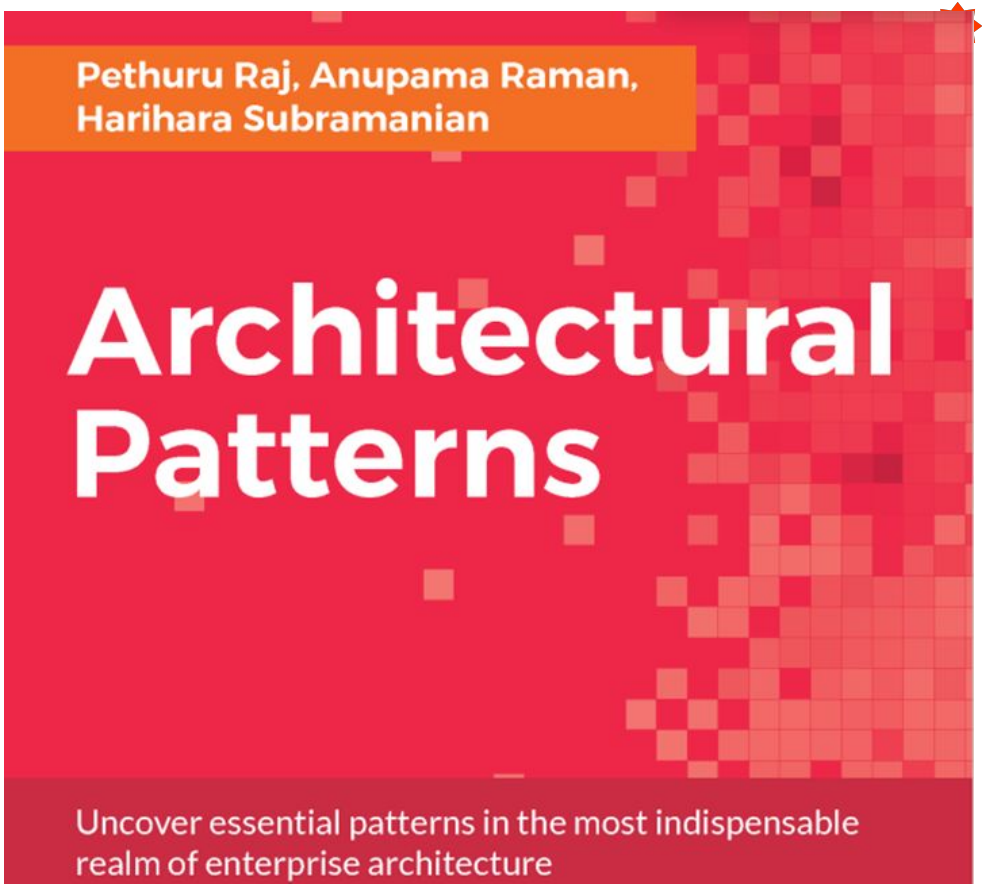
UNIVERSIDAD DE LIMA | 60 años

Estilos de arquitectura

Es una especialización de elementos y tipos de relaciones, junto con un conjunto de restricciones sobre cómo se pueden usar. Proporciona la organización de alto nivel del software

Patrones arquitectónicos de Software

los patrones arquitectónicos se utilizan para decidir hábilmente la arquitectura estable de un sistema en desarrollo. Los patrones arquitectónicos permiten tomar una serie de decisiones en cuanto a la elección de tecnologías y herramientas.



Object-oriented architecture (OOA)

- OOA: Inheritance, polymorphism, encapsulation, and composition
- Producing highly modular (highly cohesive and loosely coupled), usable and reusable software applications.

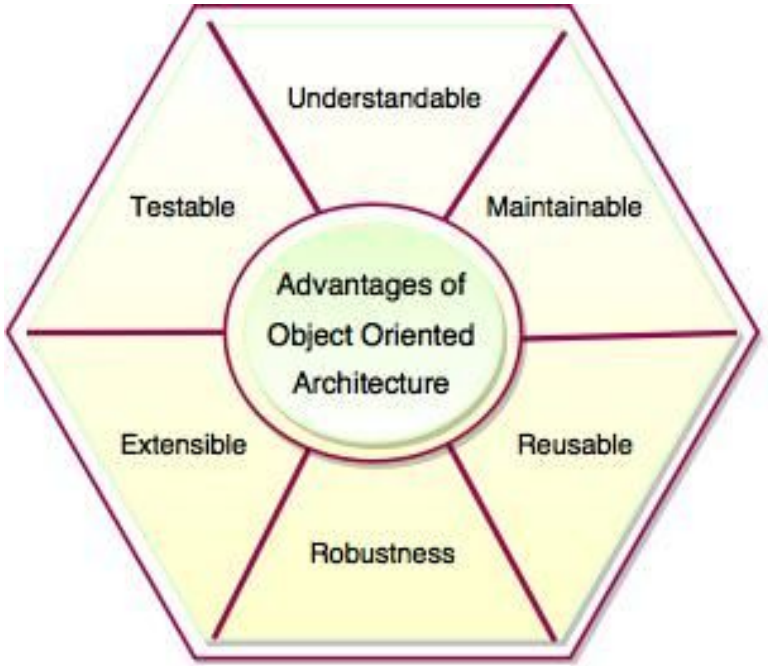
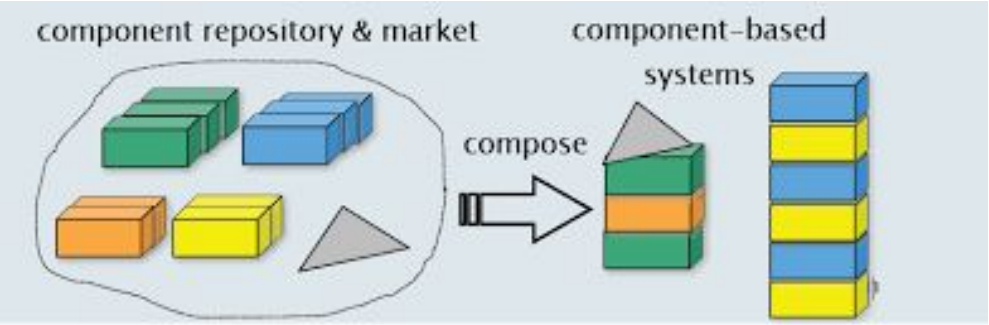
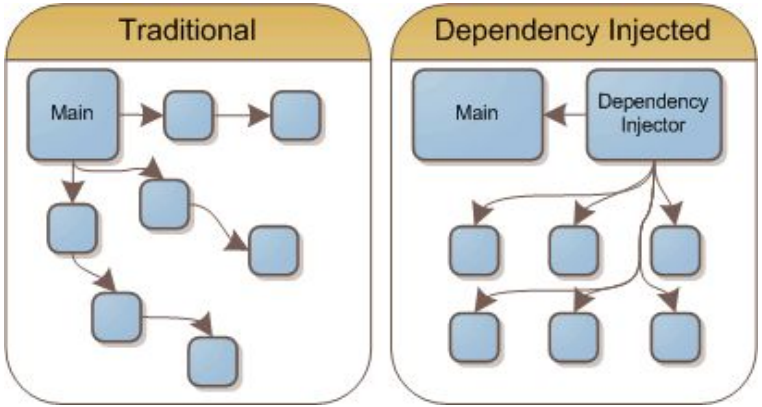


Fig. Advantages of Object Oriented Architecture

Component-based assembly (CBD) architecture

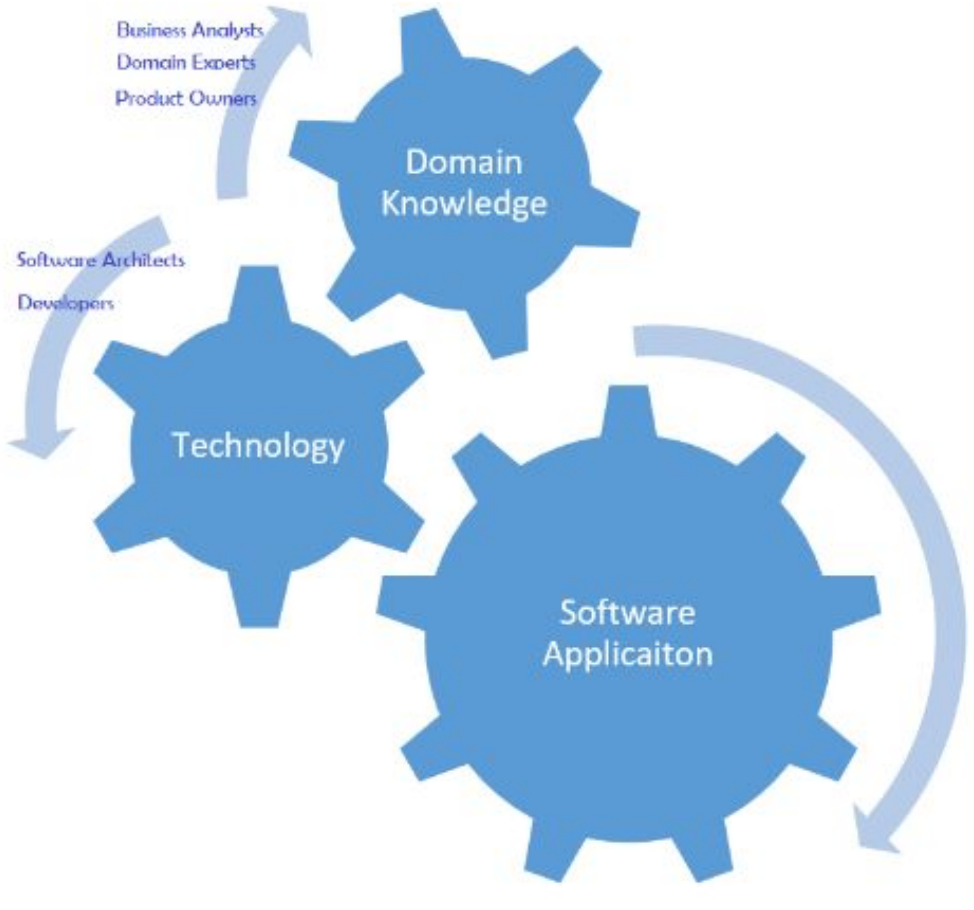


- Building-block for designing and developing enterprise-scale applications
- Components expose well-defined interfaces
- Components are reusable, replaceable, substitutable, extensible, independent
- **Dependency injection (DI)** manage dependencies between components



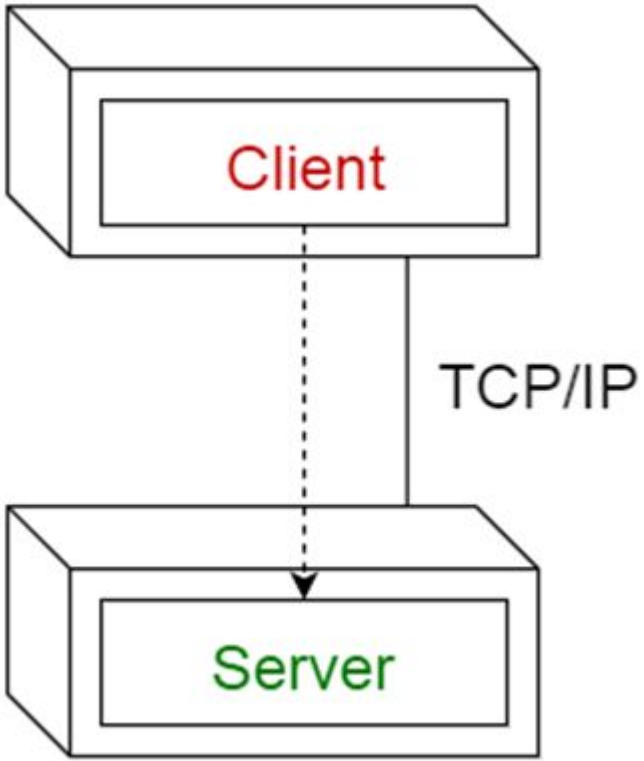
Domain-driven design (DDD) architecture

- es un enfoque orientado a objetos para diseñar software basado en el dominio comercial
- El modelo de dominio se puede ver como un Framework del cual se pueden preparar y racionalizar las soluciones.



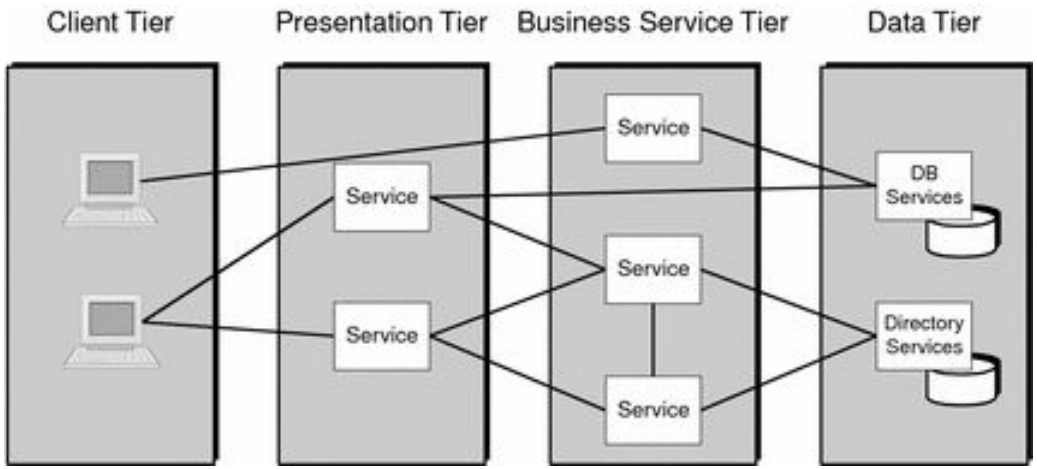
Client/server architecture

- El **cliente** realiza solicitudes al **servidor**.
- Arquitectura de dos niveles
- Ventajas:
  - Mayor seguridad
  - Acceso a datos centralizados
  - Facilidad de mantenimiento
- Desventajas:
  - Impacto negativo en la extensibilidad, escalabilidad y confiabilidad del sistema



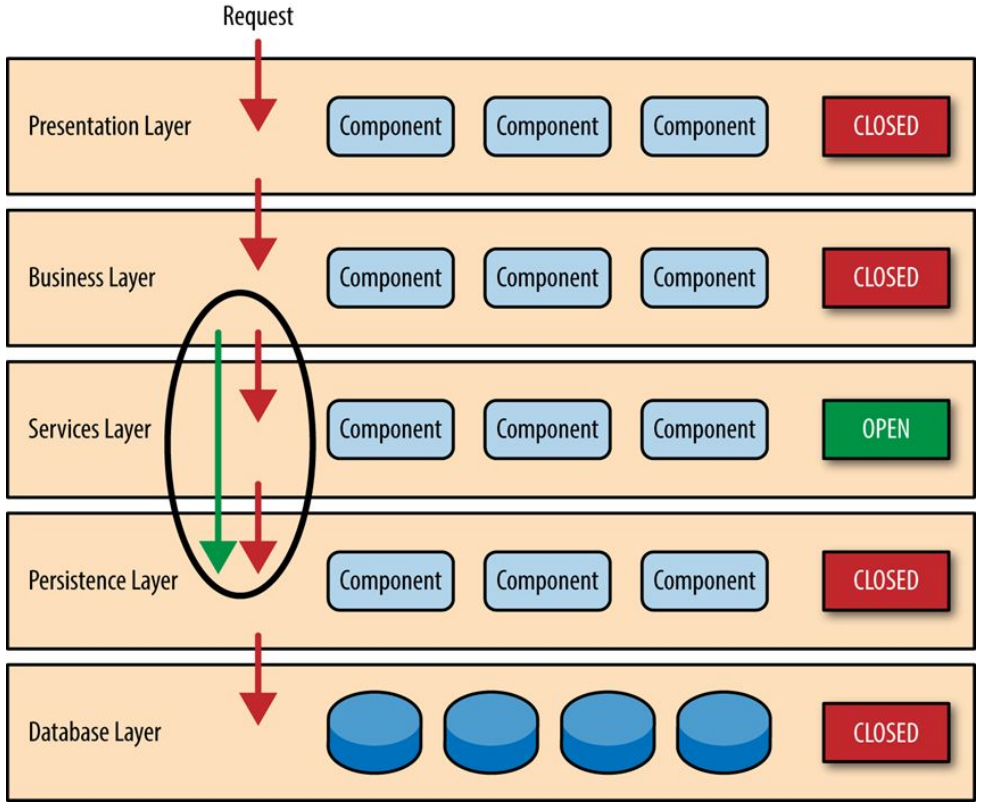
Multi-tier distributed computing architecture

- Los componentes de la aplicación se pueden implementar en varias máquinas.
- Alta disponibilidad, escalabilidad, manejabilidad
- Las aplicaciones web, en la nube y móviles se implementan utilizando esta arquitectura.



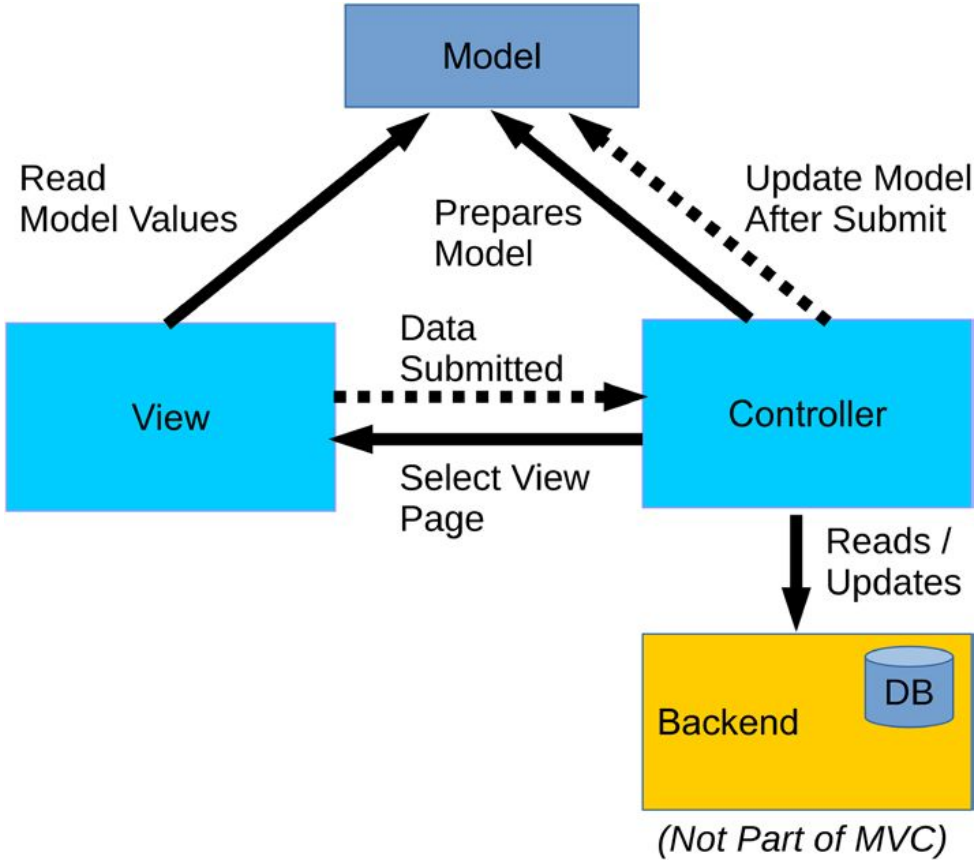
Layered/tiered architecture

- Patrón arquitectónico más utilizado
- Hace la aplicación:
  - mantenible
  - Comprobable
  - Fácil de asignar roles específicos y separados
  - Fácil de actualizar y mejorar las capas por separado



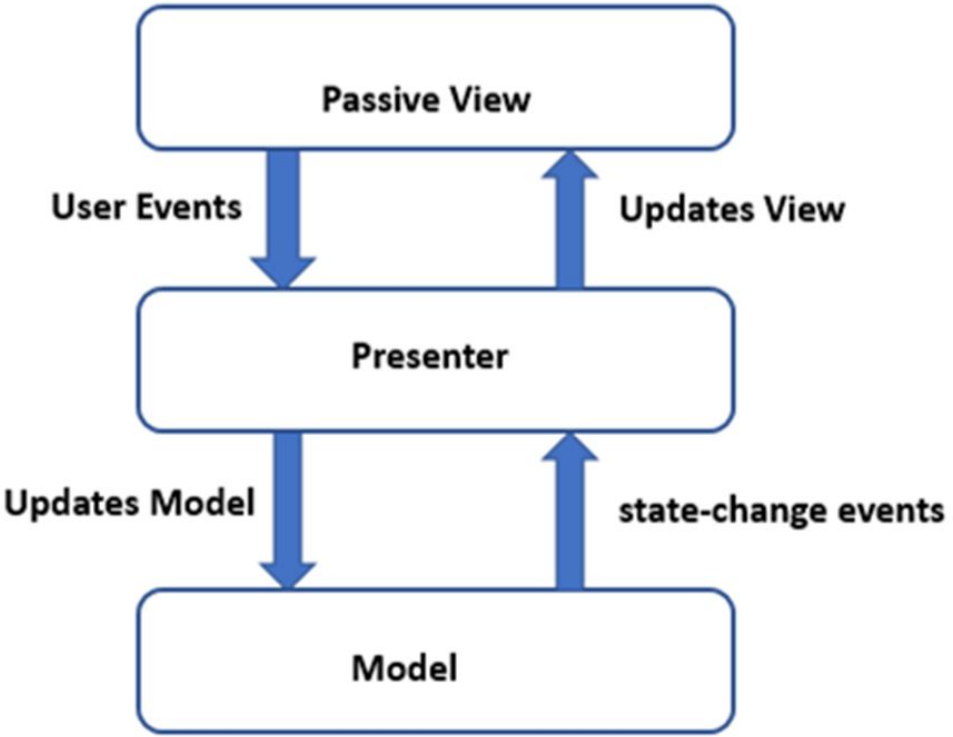
MVC Pattern

- **Modelo**: Gestiona los datos de una aplicación.
- **Vista**: describe la presentación de los datos y los elementos de control al usuario.
- **Controlador**: maneja la entrada del usuario y prepara el conjunto de datos necesario para que la parte de la vista haga su trabajo



The model view presenter (MVP) pattern

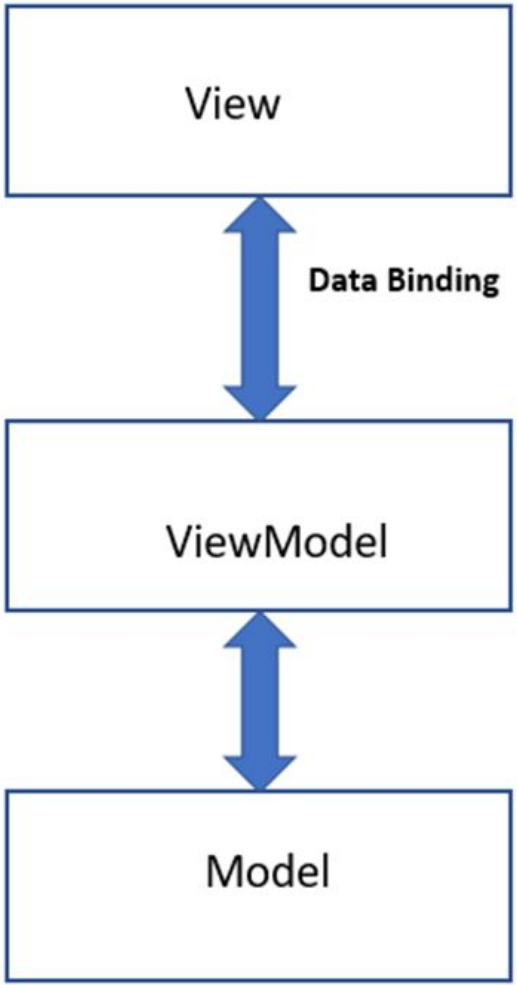
Diseñado principalmente para facilitar la realización de pruebas unitarias automatizadas



The model-view-viewmodel (MVVM) pattern

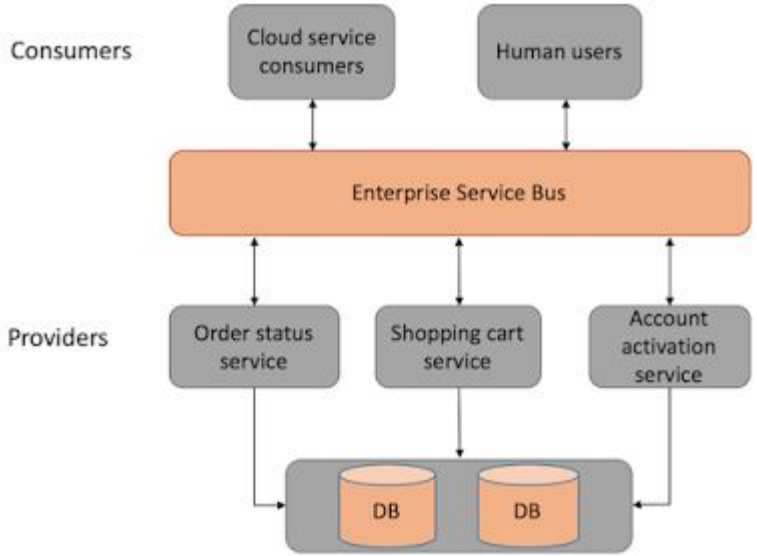
Separación entre los componentes Modelo y Vista

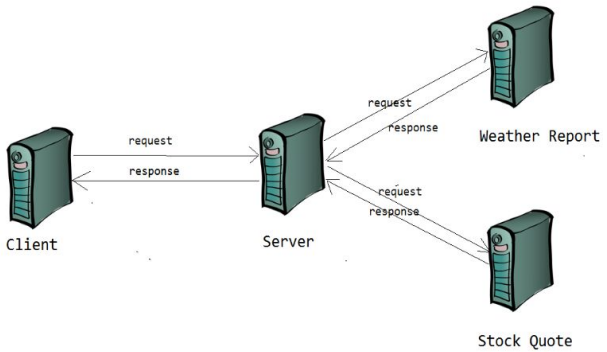
- **Modelo**: lógica de negocio y datos
- Vista: componentes de la interfaz de usuario como CSS, HTML. No realiza ninguna manipulación sobre los datos.
- **ViewModel**: mantiene la vista separada del modelo. Permite la interacción y coordinación entre la Vista y los componentes del modelo



Service-oriented architecture (SOA)

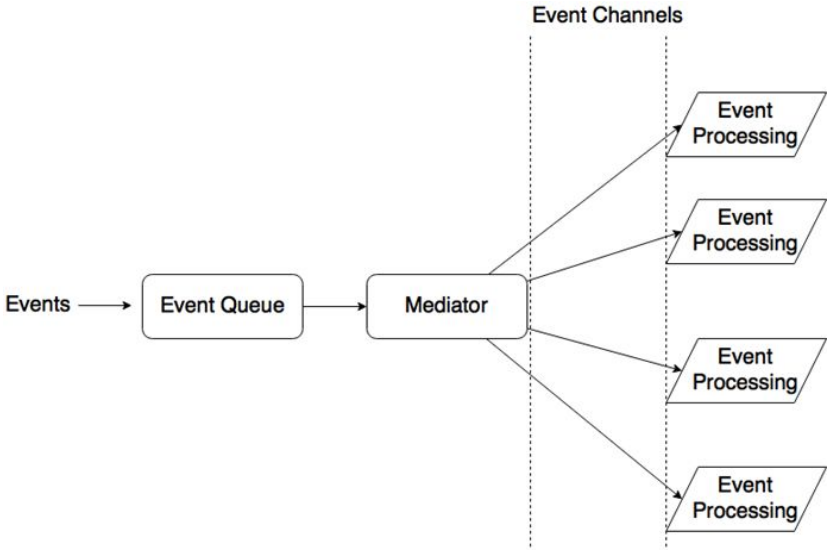
La Arquitectura Orientada a Servicios (SOA) es una estrategia para construir sistemas de software centrados en el negocio a partir de bloques de construcción interoperables y poco acoplados (llamados Servicios) que se pueden combinar y reutilizar rápidamente, dentro y entre empresas, para satisfacer las necesidades comerciales.

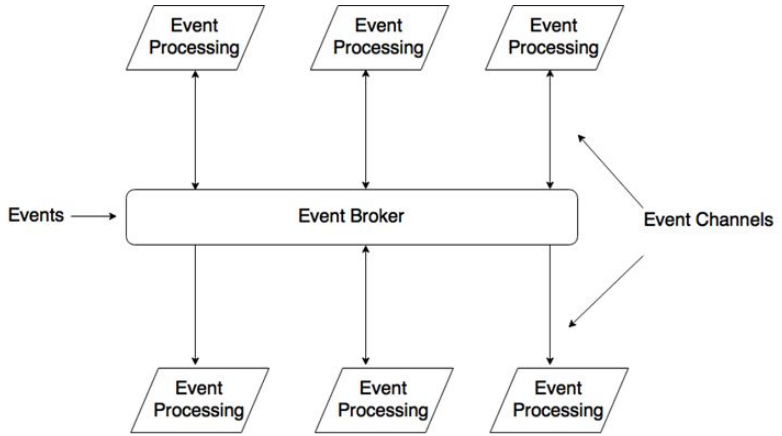




Event-driven architecture (EDA)

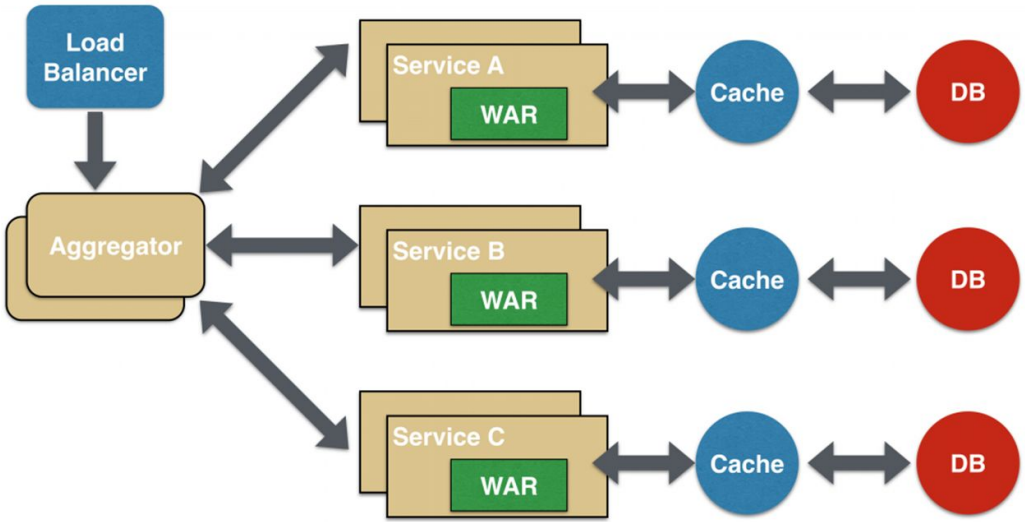
- Es una arquitectura asíncrona y distribuida, pensada para crear aplicaciones altamente escalables.
- Los componentes no se comunican de forma tradicional.
- En esta arquitectura, se espera que las aplicaciones lancen diversos “eventos” para que otros componentes puedan reaccionar a ellos.





Arquitectura Orientada a Microservicios

- Consiste en crear pequeños componentes de software que solo hacen una tarea, la hace bien y son totalmente autosuficientes
- Los Microservicios son Altamente Cohesivos
- Los Microservicios se comuniquen con otros Microservicios para delegar ciertas tareas.



UNIVERSIDAD DE LIMA | 60 años

Software de gestión de versiones

La gestión de versiones es el proceso de realizar un seguimiento de las diferentes versiones de los componentes de software y los sistemas en los que se utilizan estos componentes. También implica garantizar que los cambios realizados por diferentes desarrolladores a estas versiones no interfieran entre sí.

¿Por qué utilizar control de versiones?

La gestión de versiones es el proceso de gestión de líneas de código y líneas base.

Mantenemos nuestro código en un único repositorio.

Cualquier miembro del equipo puede hacer / deshacer modificaciones.

Facilita la integración

Tendremos un registro de **todos** nuestros cambios.

Referencias Bibliográficas

1. Bass, L., Clements, P., & Kazman, R. (2003). Software architecture in practice. Addison-Wesley Professional.
2. Sommerville, I. (2015). Software engineering 10th Edition. ISBN-10, 137035152, 18.
3. Bennett, S., McRobb, S., & Farmer, R. (2010). Object-oriented systems analysis and design using UML. McGraw-Hill.
4. Pressman, R. S., & Troya, J. M. (2010). Ingeniería del software. McGraw-Hill.