Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

INGENIERÍA DE SOFTWARE I

Arquitectura de software

Semana 11

SABERES PREVIOS

- ¿Qué abarca la arquitectura de software?
- ¿Para qué se usa la arquitectura de software?

LOGRO

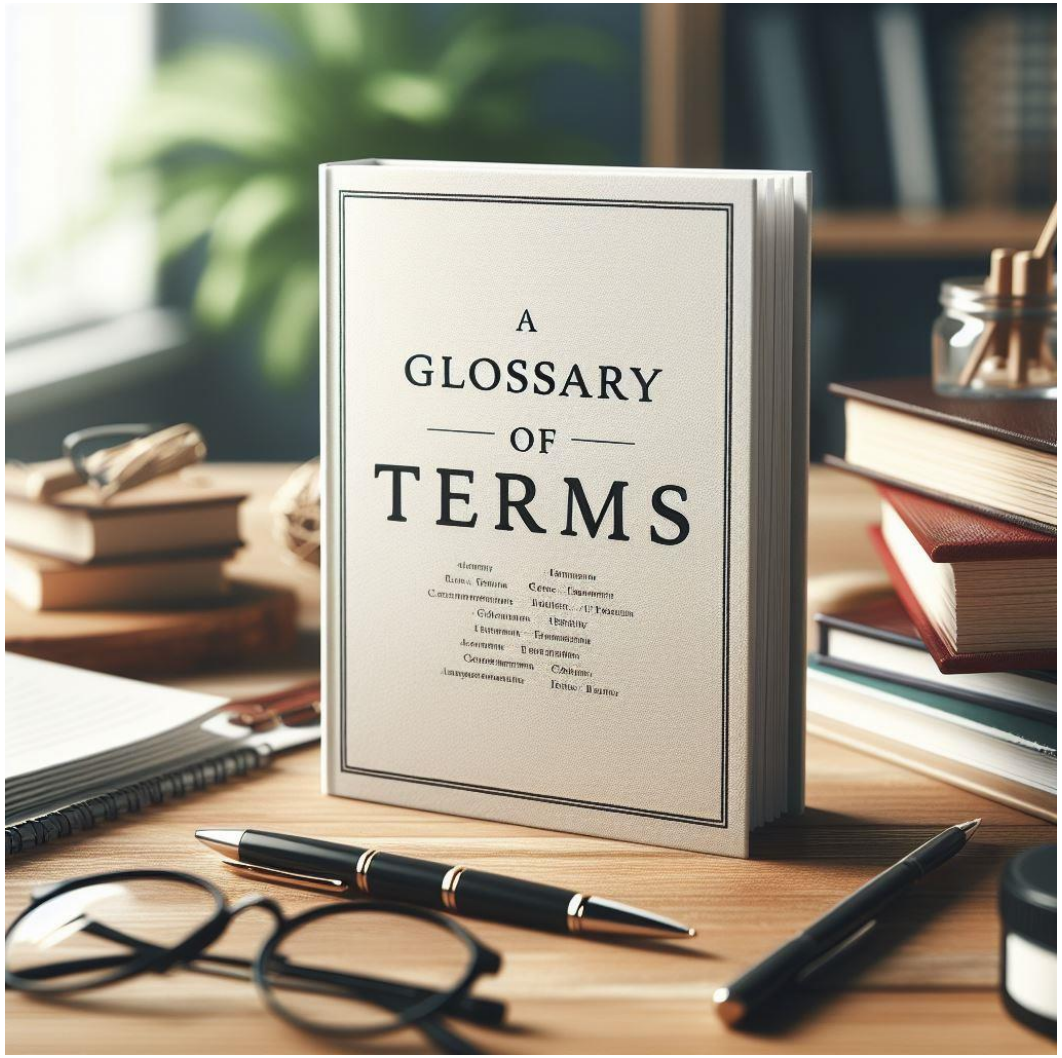
- Al término de la sesión, los estudiantes reconocen los fundamentos de la arquitectura de software.

TEMARIO

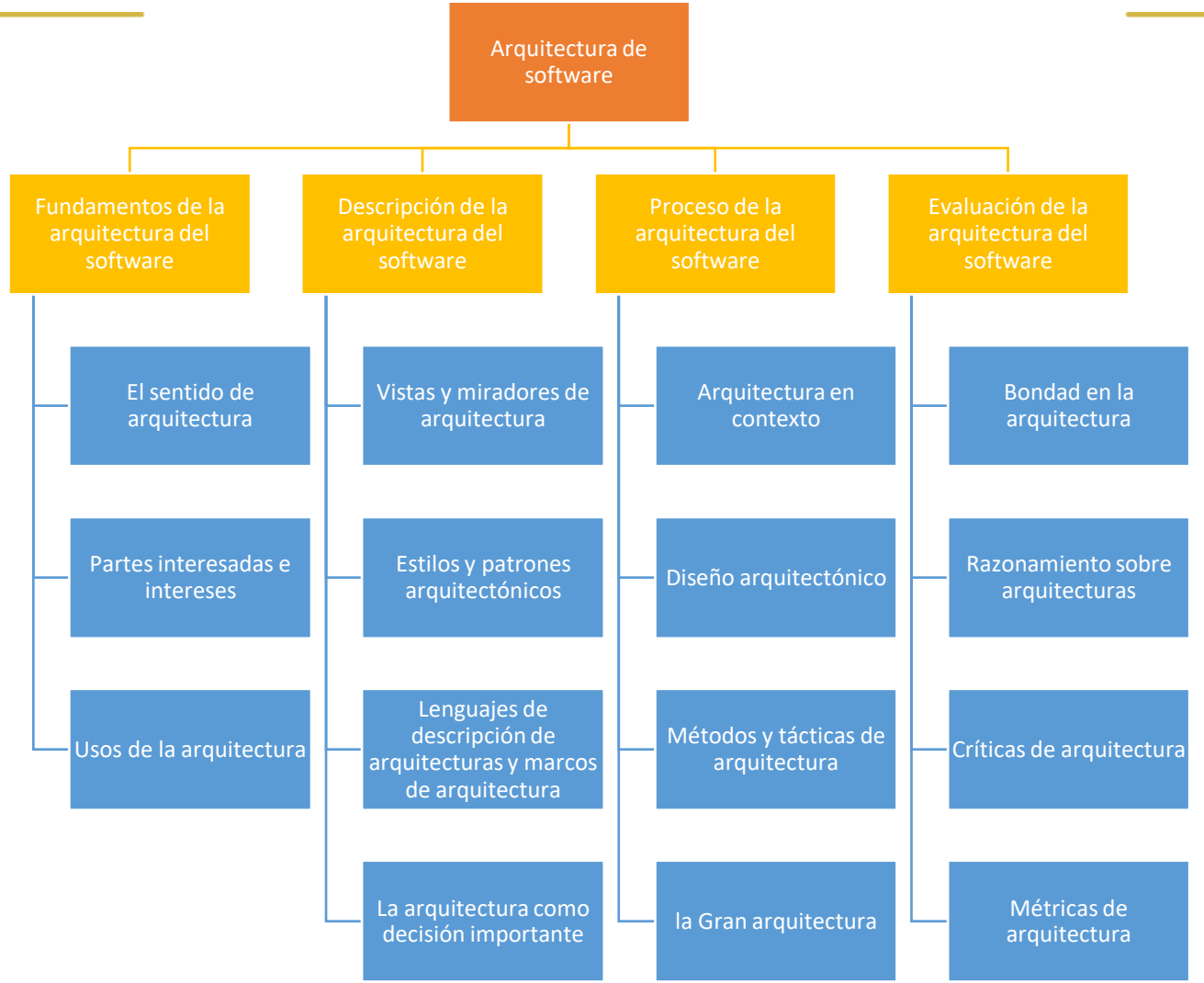
1. Fundamentos de arquitectura de software
2. Descripción de la arquitectura de software
3. Proceso de arquitectura de software
4. Evaluación de la arquitectura de software

Glosario

Acrónimos	
AD	Descripción de la arquitectura
ADL	Lenguaje de descripción de Arquitecturas
API	Interfaz de programación de Aplicaciones
ASR	Requisito de relevancia Arquitectónica
IDL	Lenguaje de descripción de interfaces
MVC	Modelo vista controlador
KA	Área de Conocimiento



Desglose de temas para la KA de arquitectura de software



1. Fundamentos de arquitectura de software

1.1 Sentido de “arquitectura”

Primero:

- “Arquitectura" suele referirse a una **disciplina**: el arte y la ciencia de construir cosas, en este caso, **sistemas intensivos en software**.
- La disciplina implica
  - Conceptos
  - Principios
  - Procesos y
  - Métodos
- Que la comunidad ha descubierto y adoptado.

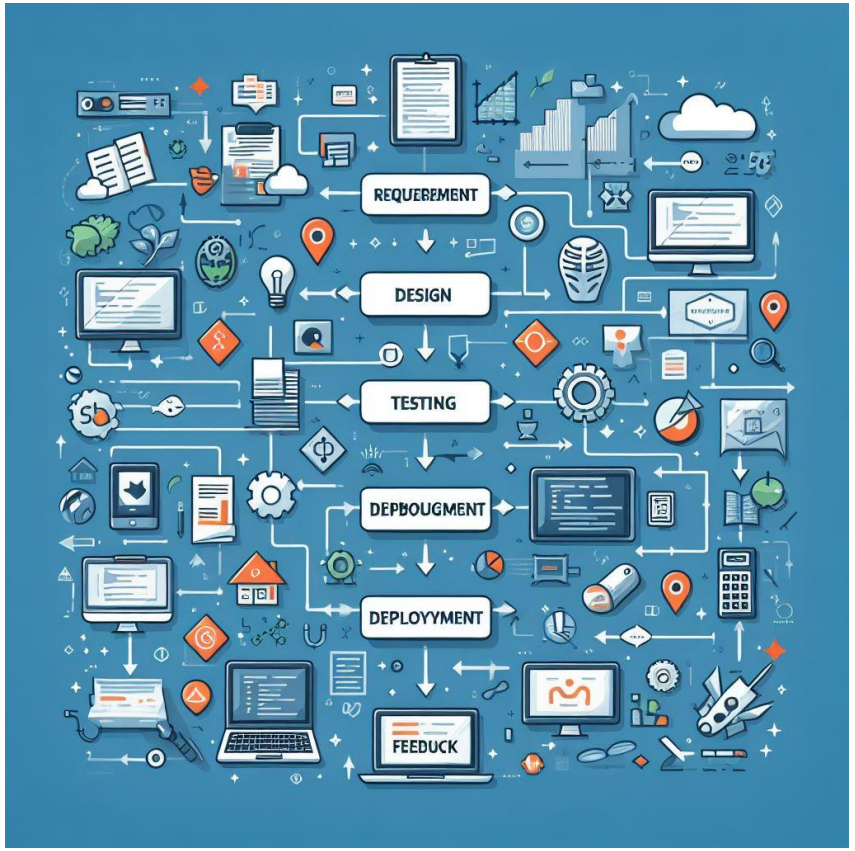
1.1 Sentido de “arquitectura”

En segundo lugar

La arquitectura se refiere a los diversos **procesos** a través de los cuales se realiza esa disciplina.

En esta KA, distinguimos el **diseño de la arquitectura** como una fase específica del ciclo de vida que abarca un **conjunto concreto de actividades**, y lo diferenciamos de los **procesos de arquitectura** más amplios que abarcan todo el ciclo de vida.

Ambos se discuten en el tema Procesos de Arquitectura de Software.



1.1 Sentido de “arquitectura”

En tercer lugar

- "arquitectura" se refiere al resultado de **aplicar la disciplina y los procesos** de diseño arquitectónico para concebir **arquitecturas de sistemas de software**.
- **Las arquitecturas** como resultados se expresan en **descripciones** de arquitectura.
- El concepto de arquitectura ha evolucionado y hoy en día se utilizan muchas definiciones. Una de las primeras definiciones de arquitectura, de **1990,** hacía hincapié en la **estructura del software.**

1.1 Sentido de “arquitectura”

La arquitectura en el contexto del software se refiere a la **estructura organizativa de un sistema** o componente, como se define en IEEE Std 610.12-1990.

- Sin embargo, esta definición inicial **no captura completamente la evolución** del pensamiento sobre arquitectura.
- La arquitectura de software implica un **estudio más amplio** de las estructuras y arquitecturas de software, que van más allá de la estructura del código.

Se considera que la arquitectura de software comprende el **conjunto de estructuras** necesarias para **razonar sobre el sistema**, incluyendo elementos de software, relaciones entre ellos y propiedades de ambos.

1.1 Sentido de “arquitectura”

En los años 90

La arquitectura de software se desarrolló como una **disciplina más amplia** al **reconocer** que muchas estructuras de sistemas de software **van más allá** de la mera estructura del **código**.

Esto generó debates que condujeron a **definiciones actuales** que resaltan la importancia de los conceptos fundamentales de un **sistema en su entorno**, expresados mediante **elementos, relaciones y principios** de diseño y evolución.

1.1 Sentido de “arquitectura”

Las ideas clave de la definición actual incluyen:

- Arquitectura (de un sistema). **conceptos o propiedades fundamentales** de un sistema en su entorno plasmados en sus elementos, relaciones y en los principios de su diseño y evolución.
  - Las ideas clave de esa definición son las siguientes:
    - (1) La arquitectura trata de lo que es **fundamental para un sistema** de software; no todos los elementos, interconexiones o interfaces son se considera fundamental.
    - (2) La arquitectura considera un sistema en su entorno. Al igual que la arquitectura de edificios, la arquitectura de software **mira hacia el exterior**; tiene en cuenta el **contexto** de un sistema más allá de sus límites para considerar a las **personas, las organizaciones, el software, el hardware y otros dispositivos** con los que el sistema debe interactuar.

1.1 Sentido de “arquitectura”

En resumen

La arquitectura de software **no se limita** a la estructura del **código**, sino que abarca todas las **estructuras necesarias** para **comprender y diseñar un sistema de software** en su totalidad, incluyendo su contexto y relaciones con otros elementos del entorno.

1.2 Partes interesadas e intereses

Un sistema de software tiene muchas partes interesadas con distintos papeles e intereses en relación con ese sistema. Estos distintos intereses se denominan preocupaciones, siguiendo la separación de preocupaciones de Dijkstra:

- El pensamiento inteligente implica estudiar un aspecto aislado de un objeto para comprenderlo en profundidad, reconociendo la importancia de enfocarse en una sola preocupación a la vez.
- La separación de preocupaciones consiste en abordar un aspecto específico sin distraerse con otros, lo que permite ordenar eficazmente el pensamiento y mantener la coherencia en el análisis.

1.2 Partes interesadas e intereses

Lo fundamental de un sistema varía en función de las preocupaciones y funciones de las partes interesadas. Las estructuras de software, por tanto, también varían según los roles y preocupaciones de las partes interesadas.

El **cliente** de un sistema de software: está más interesado en **cuándo estará listo** el sistema y cuánto **costará** construirlo y utilizarlo.

Los **usuarios**: están más interesados en **lo que hace y cómo usarlo**.

Los **diseñadores y programadores** que construyen el sistema tienen sus propias preocupaciones: como saber si un **algoritmo cumplirá los requisitos** del sistema.

Los responsables de garantizar la seguridad de funcionamiento del sistema tienen otras preocupaciones.

1.2 Partes interesadas e intereses

Las preocupaciones abarcan una amplia **gama de aspectos** que pueden influir en un sistema, incluyendo factores de:

- Desarrollo, tecnológicos, empresariales, operativos, organizativos, políticos, económicos, legales, normativos, ecológicos y sociales.

Al igual que los requisitos de software, estas preocupaciones pueden clasificarse en **funcionales, no funcionales o restrictivas**. Se manifiestan a través de:

- Requisitos, atributos de calidad, propiedades emergentes y diversas restricciones.

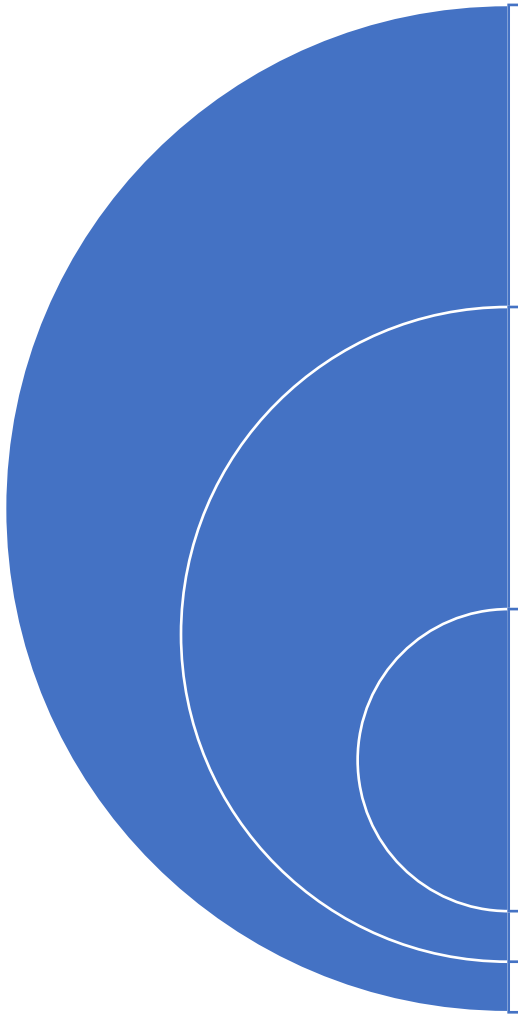
Estas preocupaciones moldean la arquitectura de software y los productos relacionados.

1.2 Partes interesadas e intereses

Ejemplos de problemas arquitectónicos

- Asequibilidad, agilidad, garantía, autonomía, disponibilidad, comportamiento, objetivos y estrategias empresariales, complejidad, conformidad con la normativa, concurrencia, control, coste, accesibilidad de los datos, desplegabilidad, disponibilidad, eficiencia energética, evolucionabilidad, extensibilidad, viabilidad, flexibilidad, funcionalidad, garantía de la información, comunicación entre procesos, interoperabilidad, limitaciones conocidas, mantenibilidad, modificabilidad, modularidad, apertura, rendimiento, privacidad, calidad del servicio, fiabilidad, utilización de recursos, reutilización, seguridad, escalabilidad, programación, seguridad, modos del sistema, estructura del software, integración de subsistemas, sostenibilidad, características del sistema, comprobabilidad, usabilidad, uso, experiencia del usuario.

1.3 Usos de la arquitectura



Proporcionar a quienes trabajan con él una comprensión compartida del sistema **para guiar su diseño y construcción**.

Una arquitectura también sirve como **concepción preliminar** del sistema de software que proporciona una base para analizar y evaluar alternativas.

Un tercer uso común es permitir la **ingeniería inversa** (o arquitectura inversa) ayudando a los que trabajan con ella a **entender un sistema** de software existente **antes de emprender su mantenimiento, mejora o modificación**. Para apoyar estos usos, la arquitectura debe estar documentada.

1.3 Usos de la arquitectura

La Ley de Conway

- Establece que **la arquitectura de un sistema** refleja **la estructura de comunicación** **de la organización** que lo desarrolla.

  - Esto puede ser una ventaja o desventaja según el caso.
  - Una **arquitectura bien planificada facilita la comunicación** dentro del equipo, mejora la **planificación y la reutilización** de componentes de software. Además, **permite diseñar familias de programas** identificando similitudes entre ellos y creando **componentes reutilizables y adaptables**.

2 Descripción de la arquitectura de software

En el tema anterior, se definió

- **Arquitectura de software** como los **conceptos o propiedades fundamentales de un sistema** de software en su entorno.

2. Descripción de la arquitectura de software

En **sistemas complejos**

- Desarrollados por equipos, las diversas partes interesadas pueden tener **visiones distintas** de lo esencial. Por ello, **contar con descripciones de arquitectura** (DA) tangibles **es crucial**.
- Estas **DA** proporcionan un **marco claro** para analizar, organizar y guiar el diseño e implementación del sistema, especialmente **en entornos cambiantes y con rotación** de personal.

Para **sistemas pequeños** o individuales

- Un **modelo mental** puede bastar, en contextos más complejos, las DA son esenciales para la comprensión y coordinación efectiva.

Una descripción de arquitectura (AD)

- Documenta la arquitectura de un sistema informático.
- **Se dirige** a las **partes interesadas** del sistema que **tienen inquietudes** sobre el sistema de software a las que responde la arquitectura.

2. Descripción de la arquitectura de software

Una descripción de arquitectura (**AD**)

- **Audiencia primaria** comprende:
  - Diseñadores, ingenieros y programadores cuyas preocupaciones se refieren a la construcción del sistema.
- Para estas partes interesadas, la AD sirve como un **plano para guiar** la construcción del sistema de software.
- Para otros, la AD es una **base para su trabajo**, por ejemplo, pruebas y aseguramiento de la calidad, certificación, despliegue, operación y mantenimiento y evolución futura.

2. Descripción de la arquitectura de software

Históricamente, las AD utilizaban **texto y diagramas informales** para transmitir la arquitectura.

Sin embargo, **la diversidad de las audiencias** interesadas y sus diferentes preocupaciones han dado lugar a una **diversidad de representaciones** de la arquitectura.

A menudo, **estas representaciones se especializan** en base a las prácticas existentes de las comunidades o disciplinas involucradas para abordar eficazmente esta variedad de partes interesadas y preocupaciones. Estas diversas representaciones se denominan **vistas** de arquitectura.

2.1 Vistas de Arquitectura y Puntos de Vista

Una vista de arquitectura

- **Representa uno o más aspectos** de una arquitectura para abordar una o más preocupaciones.
- Por ejemplo
  - **Una vista lógica:** muestra cómo el sistema satisfará los requisitos funcionales.
  - **Una vista de proceso:** muestra cómo el sistema utilizará la concurrencia.
  - **Una vista física:** muestra cómo se desplegará y distribuirá el sistema y
  - **Una vista de desarrollo:** muestra cómo el diseño de alto nivel se divide en unidades de implementación, las dependencias entre esas unidades y cómo se construirá la implementación.
- **Separar** las preocupaciones **por vistas** permite a las partes interesadas **centrarse en unos pocos aspectos** a la vez y ofrece un medio de gestionar la comprensibilidad y la complejidad general de la arquitectura.

2.1 Vistas de Arquitectura y Puntos de Vista

- **Los puntos de vista comunes** incluyen el **punto de vista del módulo**, utilizado para expresar la implementación de un sistema de software en términos de sus módulos y su organización.
- El **punto de vista del componente** y el conector, utilizado para expresar la organización a gran escala del **tiempo** de ejecución del software y las interacciones
- El **punto de vista lógico**, utilizado para expresar **conceptos fundamentales** del **dominio y la capacidad del software**
- El **punto de vista de los escenarios/casos de uso**, utilizado para expresar cómo **interactúan los usuarios con el sistema.**
- El **punto de vista de la información**, utilizado para **expresar los elementos de información** clave de un sistema y cómo se accede a ellos y se almacenan
- El **punto de vista del despliegue**, utilizado para expresar cómo se **configura y despliega** un sistema para su funcionamiento.
- Otros puntos de vista documentados incluyen **puntos de vista de disponibilidad, comportamiento, comunicaciones, gestión de excepciones, rendimiento, fiabilidad, seguridad y protección.**

Cada punto de vista

- Ofrece un **lenguaje específico** para abordar preocupaciones particulares, proporcionando a las partes interesadas un medio compartido de expresión.
- Estos puntos de vista pueden ser **reutilizados** por varias aplicaciones o sistemas similares dentro de una organización o comunidad.
- Incluso las representaciones genéricas como el Lenguaje Unificado de Modelado (UML) pueden adaptarse y especializarse según el sistema, su dominio o las organizaciones involucradas.

Punto de vista:

- **Formas de representación**
- Puede captar las **formas de trabajar** dentro de una disciplina o comunidad de práctica.
- Por ejemplo, un punto de **vista fiabilidad del software** recoge las prácticas existentes para identificar y analizar problemas de fiabilidad, formular problemas de fiabilidad, formular alternativas y sintetizar y representar soluciones. Al igual que los manuales de ingeniería, los genéricos y especializados proporcionan un medio para documentar o **reutilizables para los problemas recurrentes** del software.

Punto de vista:

- Clements et al. han introducido **tipos de puntos de vista** que establecen una **categorización** de los puntos de vista. Estas categorías son:
  - Módulo
  - Componente y
  - Conector, y
  - Tipos de vista de asignación

Las descripciones de arquitectura utilizan **múltiples vistas** para **representar diferentes estructuras** y abordar las preocupaciones de las partes interesadas.

Dos enfoques comunes son el enfoque sintético y el enfoque proyectivo.

En el **enfoque sintético**, los arquitectos crean **vistas del sistema y** las integran mediante reglas de correspondencia.

En el enfoque **proyectivo**, las vistas se **derivan de un modelo único**, lo que limita las posibles incoherencias pero puede reducir la expresividad.

2.2 Estilos y patrones de arquitectura

Inspirado en su uso en la larga historia de la **arquitectura de edificios**.

Un **estilo arquitectónico**

Es una **forma particular de construcción** que da lugar a los **rasgos característicos de un sistema de software**.

A menudo **expresa la organización** a **gran escala de un sistema** de software.

En cambio, un **patrón arquitectónico**

Expresa una **solución común a un problema recurrente** dentro del contexto de un sistema de software.

2.2 Estilos y patrones de arquitectura

Se han documentado varios estilos y patrones arquitectónicos:

- **Estructuras generales:** por ejemplo, en capas.
- **Sistemas distribuidos:** p. ej., cliente-servidor
- **Basados en métodos:** por ejemplo, orientados a objetos, basados en eventos, flujo de datos.
- **Interacción usuario-ordenador:** por ejemplo, modelo-vista-controlador.
- **Sistemas adaptativos:** por ejemplo, micronúcleo, arquitecturas de metanivel.
- **Máquinas virtuales:** por ejemplo, intérpretes, control de procesos basado en reglas, control de procesos.

2.2 Estilos y patrones de arquitectura

No existe una línea divisoria estricta entre **estilos arquitectónicos y patrones.**

Un **estilo arquitectónico**

Describe la **estructura** general de un sistema o subsistema y, por tanto, define las partes principales de un (sub)sistema y cómo interactúan.

Los **patrones arquitectónicos**

Existen en varios **rangos de escala** y pueden aplicarse **repetidamente** en una arquitectura de software.

Ambos proporcionan una solución a un problema informático concreto en un contexto determinado. De hecho, cualquier estilo arquitectónico puede describirse como un patrón arquitectónico.

2.2 Estilos y patrones de arquitectura

Los **patrones y estilos arquitectónicos**, así como los **puntos de vista** de arquitectura

Son **formas de expresar** aspectos específicos de las arquitecturas y diseños de software.

Utilizan un **vocabulario específico** para describir elementos de la vista, incluyendo tipos de elementos, relaciones, instancias y restricciones para combinarlos.

Estos mecanismos **codifican prácticas recomendadas** para facilitar la **reutilización** en el desarrollo de software.

2.3 Lenguajes de descripción de arquitecturas y arquitecturas Frameworks

Un lenguaje de descripción de arquitectura (ADL)

- Es un lenguaje para **expresar arquitecturas de software**.
- Los ADL surgieron de los lenguajes de interconexión de módulos para programación a **gran escala.**
- Algunos ADL se centran en un **único estilo** arquitectónico (como MetaH para sistemas de sistemas de aviónica en un estilo dirigido por eventos)
- Otros son de amplio espectro para enmarcar las preocupaciones en **toda la empresa** (comoArchiMate™).
- UML se ha utilizado con frecuencia como un ADL.

2.3 Lenguajes de descripción de arquitecturas y arquitecturas Frameworks

- Un lenguaje de descripción de arquitectura (ADL)
  - **ADLs** a menudo proporcionan capacidades **más allá de la descripción** para permitir el **análisis** de la arquitectura o la **generación de código.**
  - Un **marco de arquitectura captura las "convenciones**, principios y prácticas para la descripción de arquitecturas establecidas dentro de un dominio específico de aplicación y/o comunidad de interesados".
  - Los **marcos codifican prácticas recomendadas** en un ámbito específico y son implementados como un conjunto interconectado de puntos de vista o ADLs.
  - Algunos ejemplos son OMG Unified Architecture Framework (UAF®)y el Modelo de Referencia ISO para el Procesamiento Distribuido Abierto(RM-ODP).

2.4 La arquitectura como decisión significativa

El **diseño arquitectónico** es un **proceso creativo**.

Durante esta actividad, los **arquitectos toman muchas decisiones** que afectan la arquitectura, el proceso de desarrollo posterior y el sistema de software.

Muchos factores afectan a la toma de decisiones, en la **toma de decisiones**, incluidas las principales de software, sus **requisitos** y los **recursos** disponibles durante el durante el desarrollo y a lo largo del ciclo de vida.

El **impacto sobre atributos de calidad** y las **compensaciones** entre atributos de calidad son a menudo la base de las decisiones de diseño.

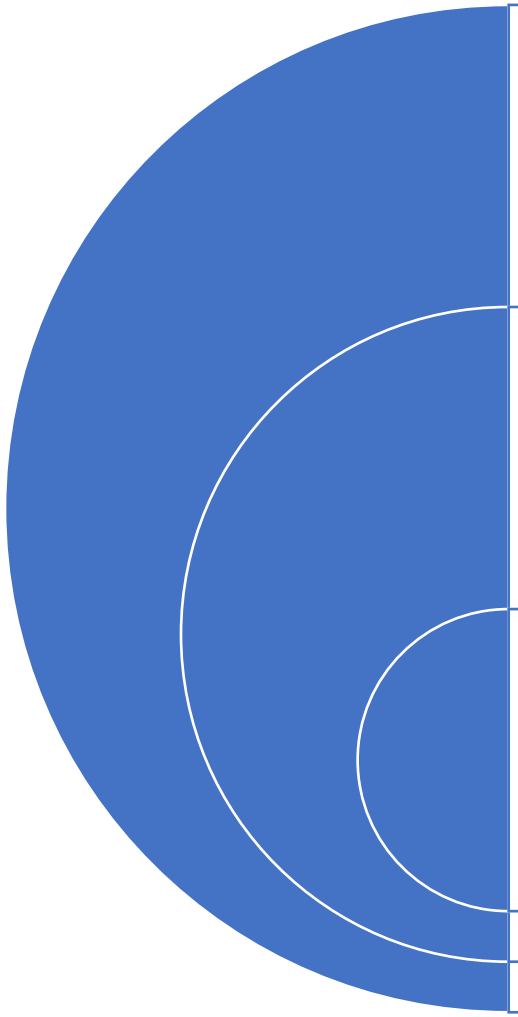
2.4 La arquitectura como decisión significativa

La **actividad de diseño arquitectónico** genera una **red de decisiones,** algunas de las cuales se basan en decisiones anteriores.

El análisis de decisiones es un método para evaluar la arquitectura.

Es **importante documentar** explícitamente las decisiones, junto con una explicación de los fundamentos detrás de cada decisión no trivial.

2.4 La arquitectura como decisión significativa



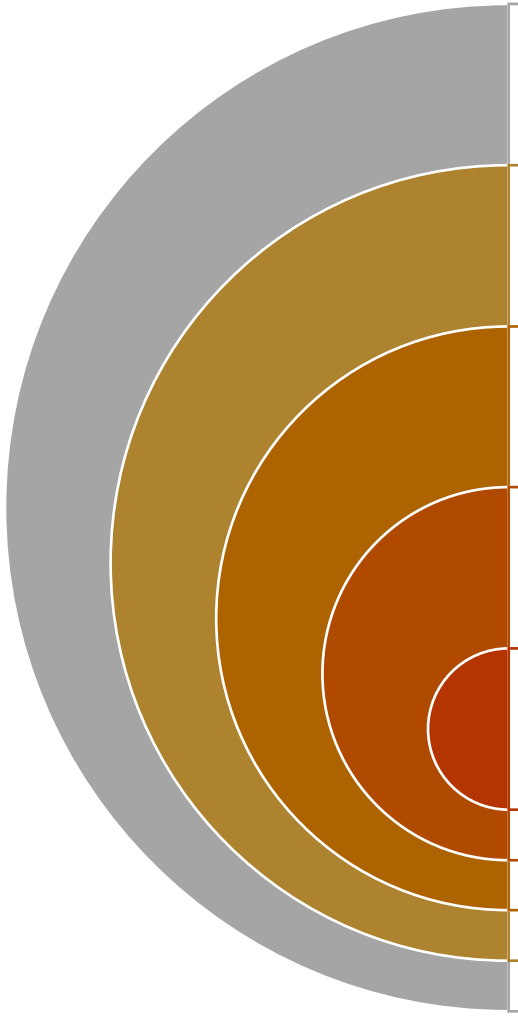
La justificación de la arquitectura explica por qué se tomó una decisión arquitectónica.

Incluye suposiciones previas, alternativas consideradas y criterios de selección.

Registrar decisiones rechazadas previene errores futuros y permite adaptarse a cambios en las condiciones relevantes.

3. Proceso de arquitectura de software

3.1 La arquitectura en su contexto



La arquitectura se desarrolla en **diversos contextos**.

En el **ciclo de vida tradicional**, el diseño arquitectónico se basa en los **requisitos del sistema.**

Algunos requisitos influyen en las decisiones arquitectónicas principales, mientras que **otros se posponen.**

En entornos de línea o **familia de productos,** la arquitectura se basa en un conjunto básico de **necesidades y requisitos.**

Esta arquitectura actúa como **punto de partida** para desarrollar **productos específicos** dentro de la línea de productos.

3.1 La arquitectura en su contexto

En los enfoques ágiles

- **No** hay una **fase separada** de diseño arquitectónico.
- La arquitectura puede estar **implícita en el código** mismo.
- En algunas prácticas ágiles, la arquitectura emerge durante el desarrollo del sistema, basada en las historias de usuario.

Este enfoque puede **funcionar** para **sistemas centrados en el usuario,** pero puede ser problemático para otros tipos de aplicaciones donde las propiedades arquitectónicas críticas no están articuladas por las historias de usuario.

3.1.1 Relación entre arquitectura y diseño

Diseño	Arquitectura
Se enfoca en un conjunto establecido de requisitos.	Da forma a los requisitos mediante negociación y análisis con las partes interesadas
Surge de la disciplina del diseño de software	Surge del diseño de software a medida que la disciplina madura
Aborda requisitos específicos del sistema.	Reconoce y aborda una gama más amplia de preocupaciones, algunas de las cuales pueden no convertirse en requisitos del sistema de software.

3.2 Diseño arquitectónico

- El diseño arquitectónico **aplica principios y métodos** para crear y documentar una arquitectura de software.
- Existen **múltiples métodos** de arquitectura para esta tarea.
- Implica **identificar** los principales **componentes, sus responsabilidades, propiedades e interfaces, así como sus relaciones e interacciones** con el entorno.
- Se centra en los **fundamentos del sistema** y otros aspectos, como los detalles internos de los componentes principales.

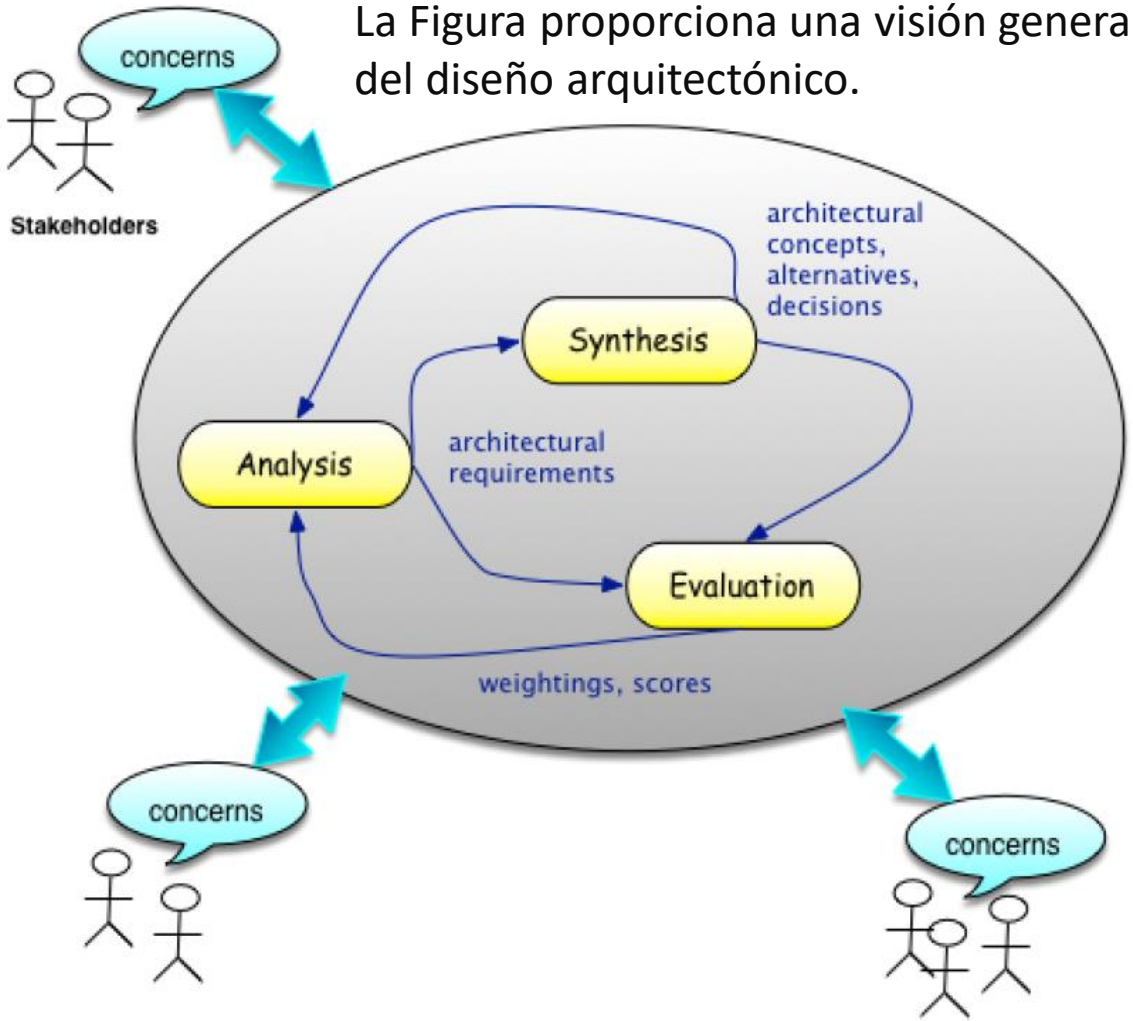
3.2 Diseño arquitectónico

Las preocupaciones típicas en el diseño arquitectónico son:	**Estilos de arquitectura** y paradigmas informáticos.
	Refinamiento a gran escala del sistema en componentes clave.
	Comunicación e interacción entre componentes.
	Asignación de preocupaciones y responsabilidades de diseño a los **componentes.**
	Interfaces entre componentes.
	Comprensión y análisis de propiedades como rendimiento, consumo de recursos y fiabilidad.
	Planteamientos a gran escala o para todo el sistema, incluyendo aspectos como seguridad cuando es relevante.

3.2 Diseño arquitectónico

- El diseño arquitectónico es un proceso iterativo que comprende tres actividades principales:
  - Análisis
  - Síntesis y
  - Evaluación.
- Estas actividades suelen realizarse simultáneamente a diferentes niveles de detalle.

La Figura proporciona una visión general del diseño arquitectónico.



3.2.1 Análisis de la arquitectura

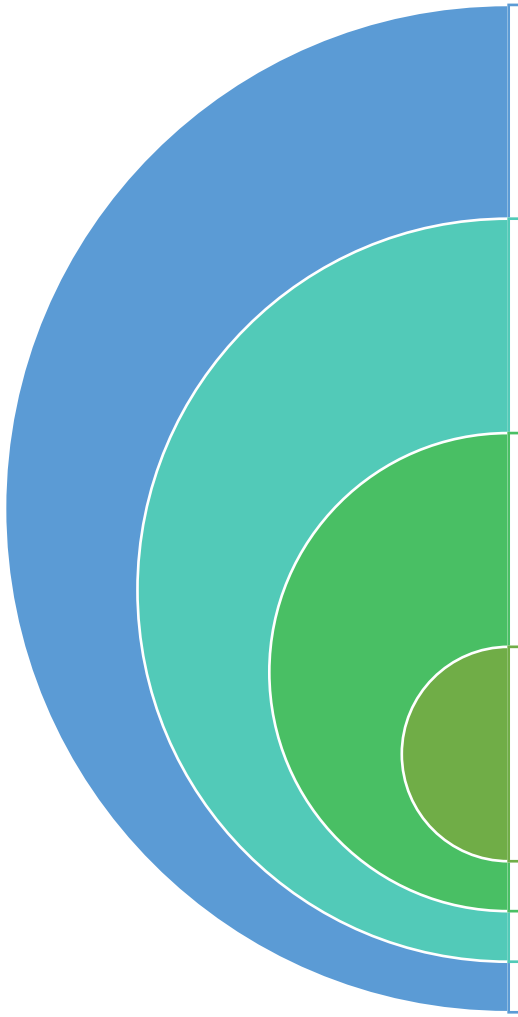
El análisis de arquitectura se enfoca en **identificar y formular requisitos** arquitectónicamente significativos (ASR) que influyen en la arquitectura del sistema.

Basado en preocupaciones identificadas y comprensión del contexto del software, incluyendo **requisitos conocidos, necesidades de partes interesadas y limitaciones** del entorno.

Los ASR reflejan los **problemas** de diseño que la arquitectura **debe resolver** y pueden requerir negociación para ajustar las necesidades y expectativas.

Produce decisiones iniciales para todo el sistema y principios generales del sistema derivados del contexto.

3.2.2 Síntesis de arquitecturas



La síntesis de arquitectura **crea soluciones** en respuesta al análisis de arquitectura.

Detalla **soluciones** a los problemas identificados por los ASR.

Interactúa entre estas soluciones para obtener resultados más detallados.

Estos resultados influyen en el análisis de arquitectura, generando ASR más detallados y **decisiones más específicas.**

La evaluación de la arquitectura

Verifica si las **soluciones cumplen** con los requisitos arquitectónicamente significativos (ASR) y determina dónde se necesita realizar ajustes.

3.3 Prácticas, métodos y tácticas de arquitectura

- Existen varios métodos de arquitectura documentados.

3.3 Prácticas, métodos y tácticas de arquitectura

Modelo en cascada:

- Este modelo sigue un enfoque secuencial lineal, en el que cada fase debe completarse antes de que pueda comenzar la siguiente. Incluye documentación para cada fase, como requisitos, diseño, pruebas y mantenimiento.

Modelo ágil:

- Este modelo hace hincapié en el desarrollo iterativo y la entrega continua, y la documentación se crea según sea necesario. Suele incluir historias de usuario, backlog del producto, planificación de sprints y notas de publicación.

Modelo en V:

- Este modelo es similar al de cascada, pero incluye la verificación y validación en cada etapa. La documentación incluye los requisitos, el diseño, los planes de pruebas y los resultados de las pruebas.

Modelo en espiral:

- Este modelo incluye múltiples iteraciones de creación de prototipos y pruebas, y la documentación se actualiza y perfecciona a lo largo del proceso.

Modelo RAD:

- Este modelo se centra en el desarrollo rápido y la creación de prototipos, con la documentación que se crea según sea necesario.

3.4 La arquitectura a lo grande

- El **diseño arquitectónico** es solo **una parte de la arquitectura** del software y se lleva a cabo en un **entorno que incluye otras arquitecturas, como la empresarial.**
- Se requiere que las **arquitecturas de aplicación** se **ajusten** a la **arquitectura empresarial**, y en sistemas de sistemas, cada sistema debe adaptarse a la arquitectura general del sistema.
- Estas relaciones se reflejan como requisitos arquitectónicamente significativos (ASR) en el software.
- Las actividades y principios de arquitectura de software también se aplican a la arquitectura de sistemas y empresarial.

3.4 La arquitectura a lo grande

- Weinreich y Buchgeher ampliaron el modelo de Hofmeister et al. para incluir estas actividades en el diseño arquitectónico.
  - **Aplicación de la arquitectura:** Supervisión e implementación para garantizar que las implementaciones sigan la arquitectura establecida.
  - **Mantenimiento de la arquitectura:** Gestión y expansión de la arquitectura después de su implementación.
  - **Gestión de la arquitectura:** Administración de arquitecturas interrelacionadas dentro de una organización.
  - **Gestión del conocimiento de la arquitectura:** Extracción, mantenimiento, compartición y aprovechamiento de activos reutilizables, como decisiones, lecciones aprendidas y documentación, en toda la organización.

4. Evaluación de la arquitectura de software

4.1 Bondad de la arquitectura

El **análisis de la arquitectura** ocurre durante **todo el proceso** de creación y mantenimiento.

La **evaluación de la arquitectura** generalmente se realiza **por terceros** en hitos específicos.

La **calidad de una arquitectura** de software se evalúa en varios aspectos.

Se deben considerar **atributos** como la robustez, la utilidad, la viabilidad y la comprensibilidad.

Cuestiones como la idoneidad **para el uso previsto** y la rentabilidad también son importantes.

La **evaluación** se basa en **requisitos o necesidades** y expectativas.

Una **"buena" arquitectura** debe equilibrar **diferentes preocupaciones y considerar las consecuencias** de sus interacciones.

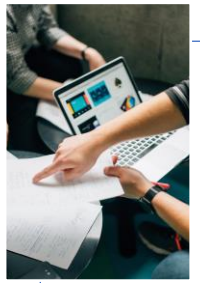
4.2 Razonamiento sobre arquitecturas

Cada **problema arquitectónico** requiere una **evaluación específica,** mejorando con descripciones sólidas de la arquitectura.

Las **Descripciones de Arquitectura** (AD) son útiles para este propósito.

La **funcionalidad y el comportamiento** se benefician de una arquitectura explícita.

Los aspectos especializados como **fiabilidad y seguridad** se basan en representaciones especializadas.



En **ausencia de documentación** completa, se confía en el conocimiento de los **participantes.**

Los **casos de uso** se emplean para verificar la **coherencia** con la arquitectura.

4.3 Revisiones de la arquitectura

Las revisiones de arquitectura son efectivas para evaluar el **estado** y la **calidad** de una arquitectura, así como para **identificar riesgos**.

Pueden ser **informales**, basadas en la experiencia, o más **estructuradas**, organizadas en torno a una lista de temas.

Parnas y Weiss propusieron un **enfoque** llamado **revisiones activas**, donde cada evaluación implica una actividad específica para obtener información.

Muchas organizaciones han **institucionalizado las revisiones de arquitectura**, utilizando marcos definidos para su realización y documentación.

4.4 Métricas de arquitectura

Una métrica de arquitectura

Es una medida cuantitativa de una característica de una arquitectura.

Varias **métricas de arquitectura** han sido definidas, muchas de las cuales **se originaron como métricas de diseño** o código.

Ejemplos de métricas

Incluyen dependencia de componentes	Ciclicidad	Complejidad ciclomática	Complejidad interna del módulo	Acoplamiento de módulos	Cohesión	Niveles de anidamiento y	Cumplimiento del uso de patrones	Estilos y API obligatorias.

4.4 Métricas de arquitectura

Una métrica de arquitectura

En paradigmas de desarrollo continuo como **DevOps**, han surgido otras métricas que no se centran directamente en la arquitectura, sino en la **capacidad de respuesta (**indicadores del estado de la arquitectura.**),** tales como:

- El tiempo de espera para los cambios
- La frecuencia de implementación
- El tiempo medio para restaurar el servicio y
- La tasa de cambio de fallas.

Actividad

- Crea un acta de reunión de trabajo con el cliente.
- *Elabora 10 preguntas a más para recabar requerimientos del sistema.*

¿Preguntas o dudas?

- ¿Qué me llevo de la clase?

Reforzamos lo aprendido

Referencias

- Bourque, P., & Fairley, R. E. (Eds.). (2014). SWEBOK: Guide to the software engineering body of knowledge (Version 3.0). IEEE Computer Society.

GRACIAS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca