

# Diseño de la arquitectura

Elizabeth Suescún Monsalve, PhD  
esuescu1@eafit.edu.co

# Contenido

- Definición e importancia
- Decisiones en el diseño arquitectónico
- Vistas arquitectónicas

# ¿Qué es arquitectura de software?

- La arquitectura de software se ocupa de definir y detallar las estructuras, sus elementos y las relaciones de esos elementos entre sí.
- Consiste en la toma de decisiones tempranas que pueden ser difíciles de cambiar más adelante.
  - Base de datos
  - Infraestructura
  - Estilos y patrones de arquitectura
  - Tecnologías a usar

# Algunas definiciones

## Clements:

La Arquitectura de Software es a grandes rasgos, una **vista del sistema** que incluye los **componentes** principales del mismo, la **conducta** de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes **interactúan** y se **coordinan** para alcanzar la misión del sistema.

## IEEE 1471-2000:

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución.

## IEEE 610.12.1990:

**Ingeniería de Software** es la aplicación de una estrategia sistemática, disciplinada y cuantificable al desarrollo, aplicación y mantenimiento del software; esto es, la aplicación de la ingeniería al software.

<http://www.sei.cmu.edu/architecture/definitions.html>

# Puntos principales

- Una arquitectura de software es una parte fundamental de un sistema de software.
- Un sistema de software está situado en un entorno y su arquitectura de software toma en consideración el entorno en el que debe operar.
- Una descripción de la arquitectura documenta la arquitectura y comunica a las partes interesadas cómo la arquitectura satisface las necesidades del sistema.
- Las vistas de la arquitectura se crean a partir de la descripción de la arquitectura, y cada vista cubre una o más preocupaciones de arquitectura de las partes interesadas

# Qué NO es arquitectura

- Una normativa madura
- Igual en la academia y en la industria
- Diseño de software con UML
- Ocurre en algún punto entre la elicitación de requisitos y la especificación de casos de uso, o entre éstos y el diseño
- Naturalmente vinculada a metodología (RUP)
- Naturalmente relacionada con modelado Orientado a Objetos
- Hay vínculo “natural” entre requisitos (casos de uso) y clases. Las herramientas arquitectónicas generan el código de la aplicación



# Qué SI es...

- Vista estructural de alto nivel
- Define **estilo o combinación de estilos para una solución**
- Se concentra en **requisitos no funcionales**
- Los requisitos funcionales se satisfacen mediante modelado y diseño de aplicación
- **Esencial para éxito o fracaso de un proyecto**



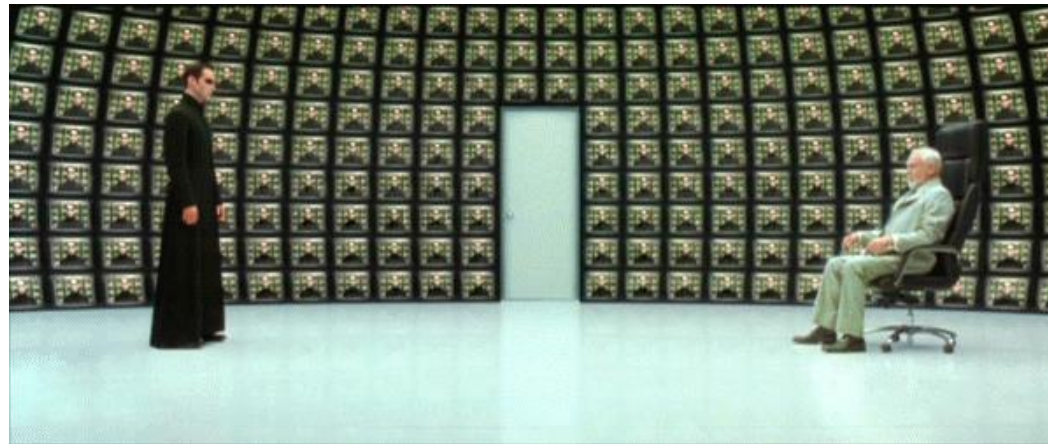
# ¿Por qué es importante?

- La arquitectura de software es la base de un sistema de software.
- Desarrollo exitoso y eventualmente mantenimiento del sistema.
- Todo sistema tiene una arquitectura, así no sea formal o no se documente.
- Beneficios:
  - Se cumplen con los requisitos
  - Predecir las habilidades del sistema de software
  - Se produce un modelo reusable.
  - Se imponen restricciones de implementación.



# ¿Quién es el arquitecto?

“A software architect is responsible for **creating or selecting the most appropriate architecture for a system** (or Systems), such that it suits the **business needs**, satisfies user **requirements**, and achieves the desired result under given **constraints**. This article describes the myriad responsibilities of a software architect, and attempts to identify human personality traits that naturally aid a person in such position...”

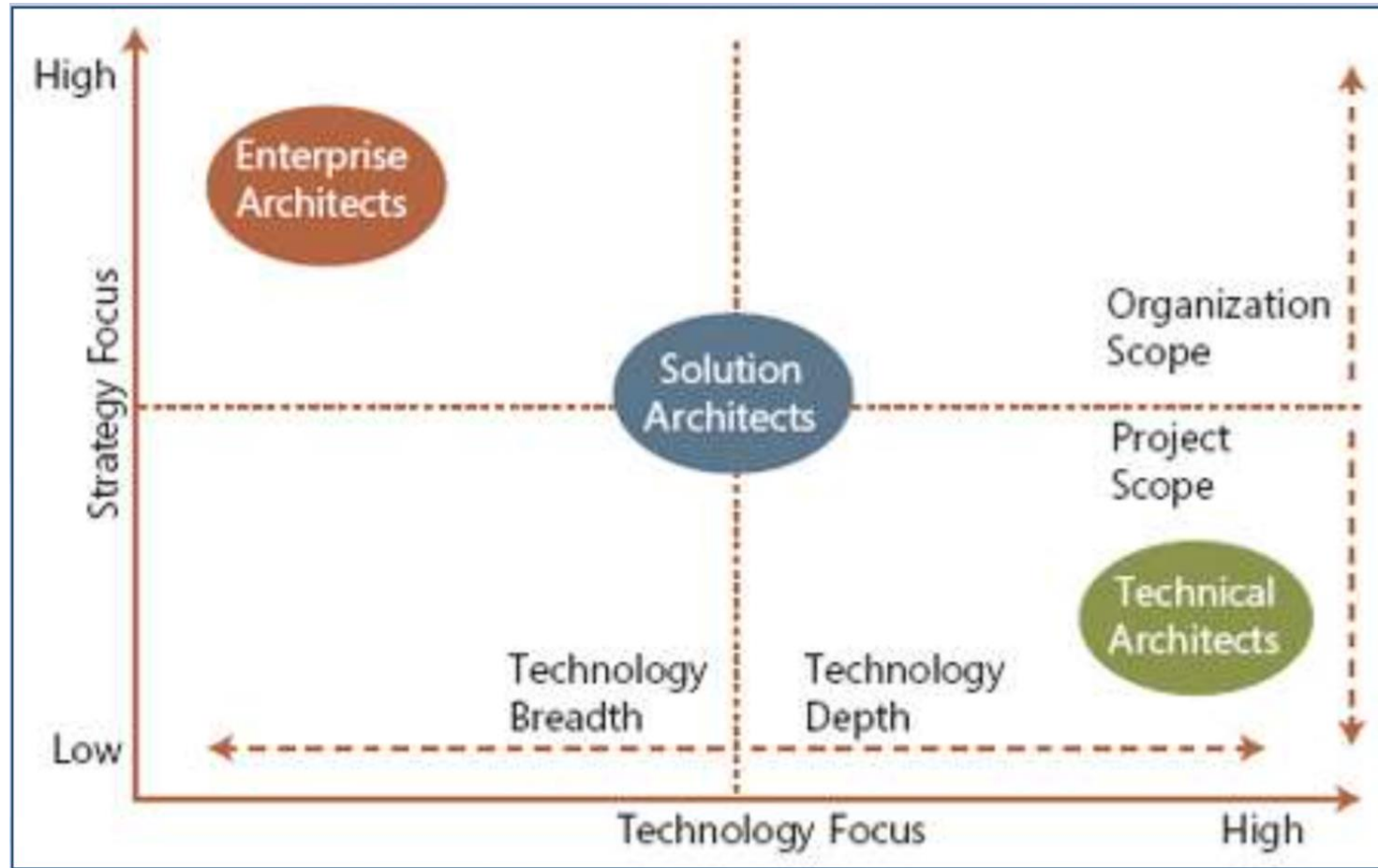


<http://www.softwarearchitectures.com/responsibilities.html>

# ¿Quién es el arquitecto?

- Líderes técnicos.
- Los arquitectos de software combinan su experiencia, conocimientos y habilidades, tanto técnicas como no técnicas, para cumplir con tales deberes.
- Se espera que los arquitectos de software tengan un conocimiento firme del diseño de arquitecturas de software, patrones de arquitectura y mejores prácticas.
- Deben ser capaces de mitigar los riesgos y evaluar las soluciones de modo que puedan seleccionar la adecuada para resolver un problema en particular.
- Son muy competentes en los lenguajes, herramientas, marcos y bases de datos que se utilizan en sus sistemas de software.
- Los arquitectos de software tienen la amplitud de conocimientos para estar al tanto de las múltiples soluciones a un problema y comprender las compensaciones entre ellas.

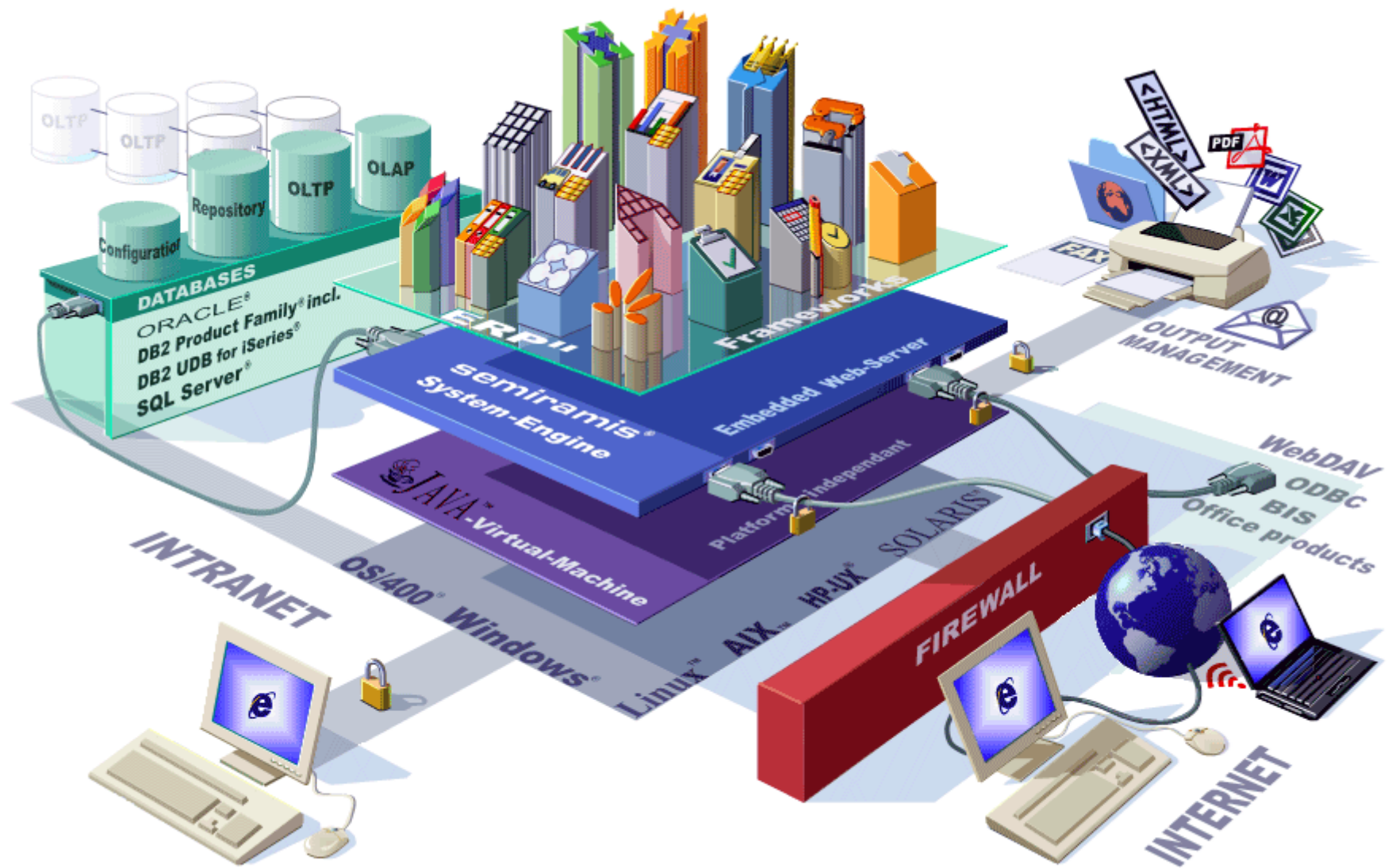
# Tipos de Arquitectos y arquitecturas



# Tipos de arquitectos

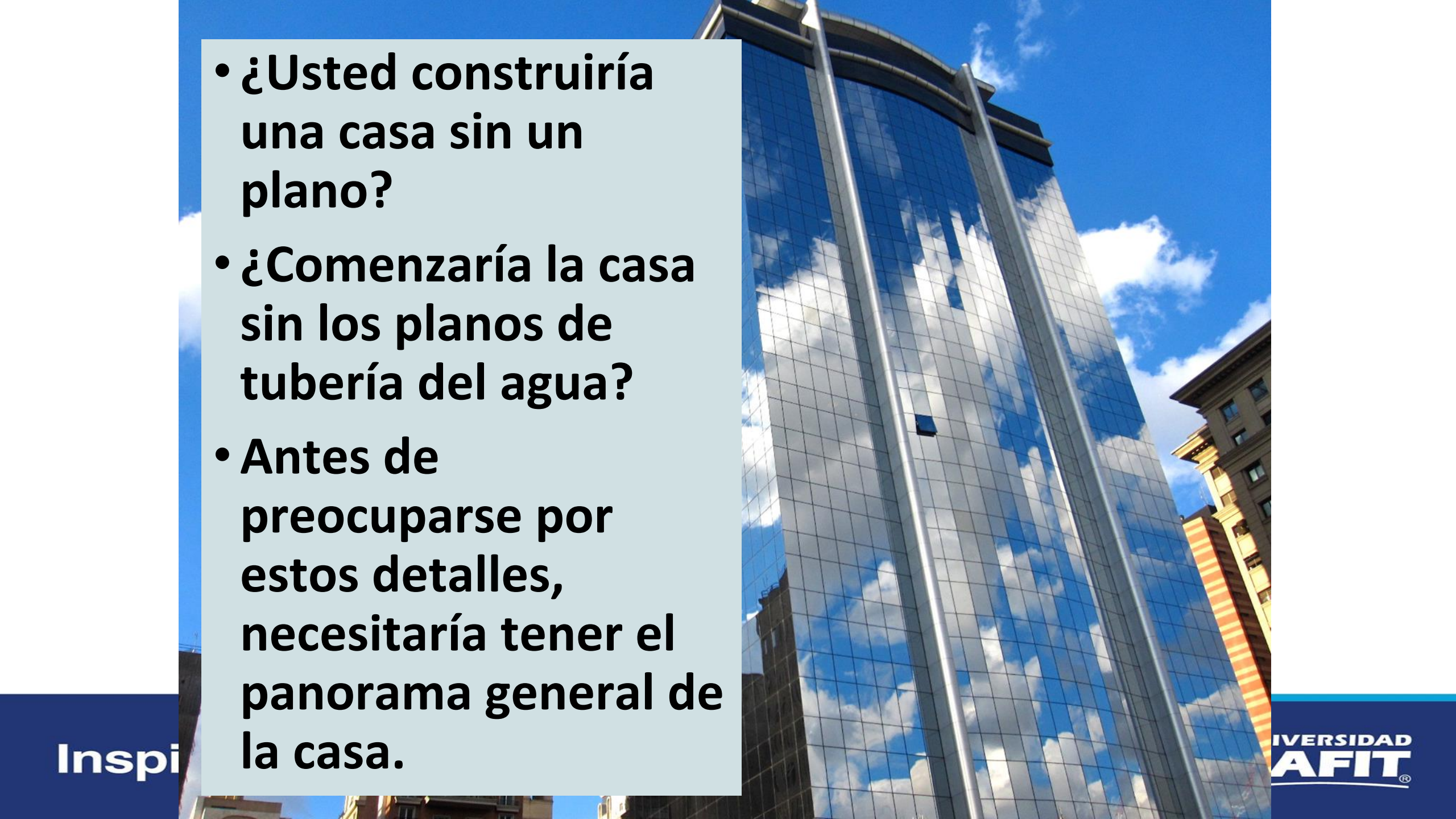
El rol de un arquitecto de software puede variar de una organización a otra.

- **Arquitecto empresarial:** garantiza que los objetivos comerciales y estratégicos de una organización estén sincronizados con las soluciones técnicas.
- **Arquitecto de soluciones:** seleccionan las tecnologías más adecuadas para el problema que debe resolverse.
- **Arquitecto de aplicaciones:** pueden recomendar soluciones o tecnologías para una aplicación y evaluar enfoques alternativos a los problemas.
- **Arquitecto de datos / Arquitecto de información:** responsable de diseñar, implementar y administrar la arquitectura de datos de una organización
- **Arquitecto de infraestructura:** involucrado en los componentes de infraestructura (servidores, elementos de red, sistemas almacenamiento, etc):
- **Arquitecto de seguridad:** realizan evaluaciones de seguridad y pruebas de vulnerabilidad.
- **Arquitecto de nube:** responsable de la estrategia e iniciativas de computación en la nube de una organización



Inspira Crea Transforma



- 
- **¿Usted construiría una casa sin un plano?**
  - **¿Comenzaría la casa sin los planos de tubería del agua?**
  - **Antes de preocuparse por estos detalles, necesitaría tener el panorama general de la casa.**

# Una mirada rápida...II



## ¿Cuáles son los pasos?

- Obtener el **modelo de datos**, se obtiene una o más **representaciones**, se analizan alternativas de **estilo o patrones arquitectónicos** para llegar a **los requisitos y atributos de calidad**. Una vez seleccionada la alternativa, se elabora la arquitectura con el empleo de un método de diseño.

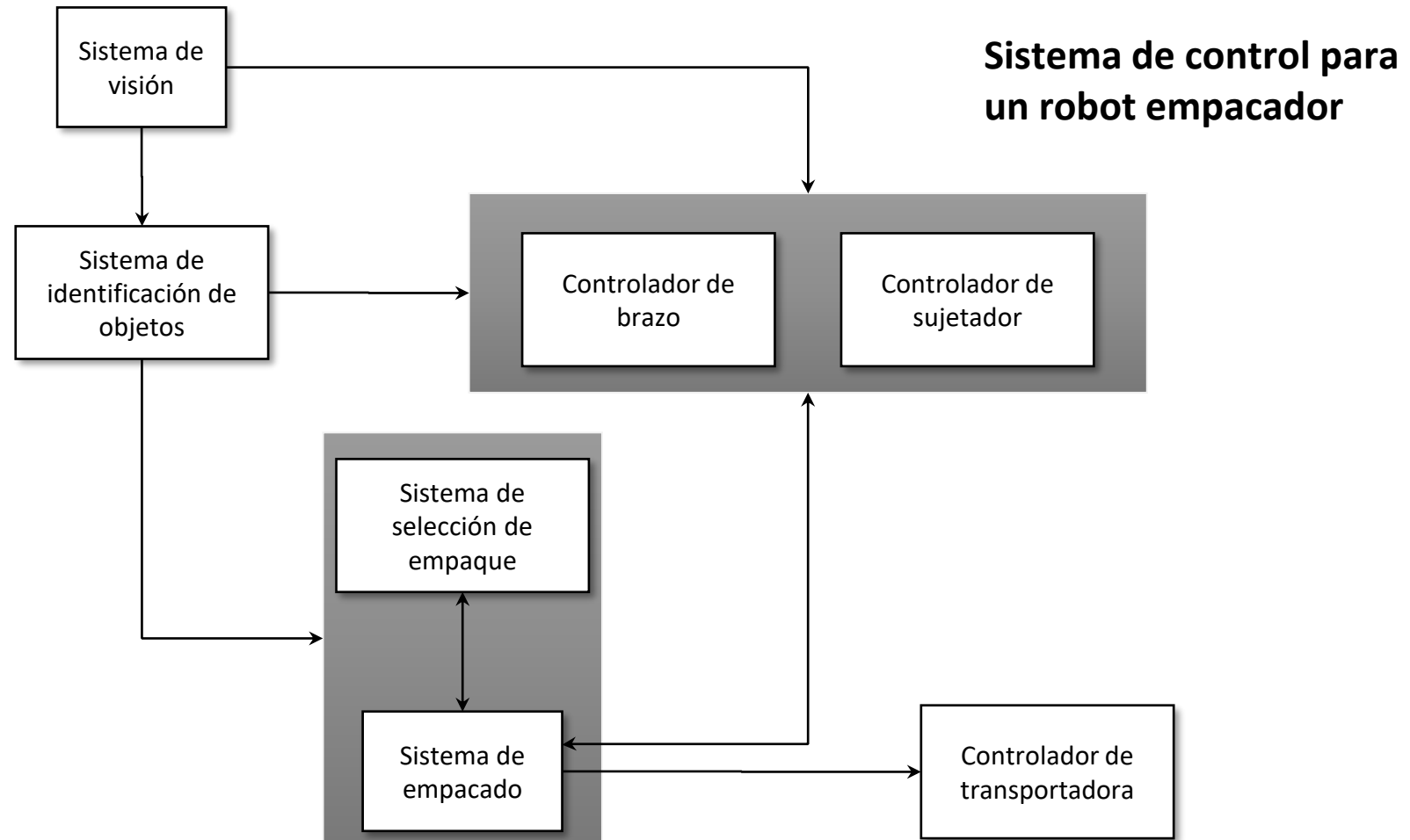
# Estereotipos utilizados en el diseño

- Se usa:
  - **Diagramas de bloques simples.**
  - Cada recuadro en el diagrama representa un **componente**.
  - Los recuadros dentro de recuadros indican que el **componente se dividió en subcomponentes**.
  - Las **flechas** significan que los datos y/o señales de control pasan de un componente a otro en la dirección de las flechas.





# Ejemplo de arquitectura



# ¿Para qué sirve?

DID YOU  
KNOW?



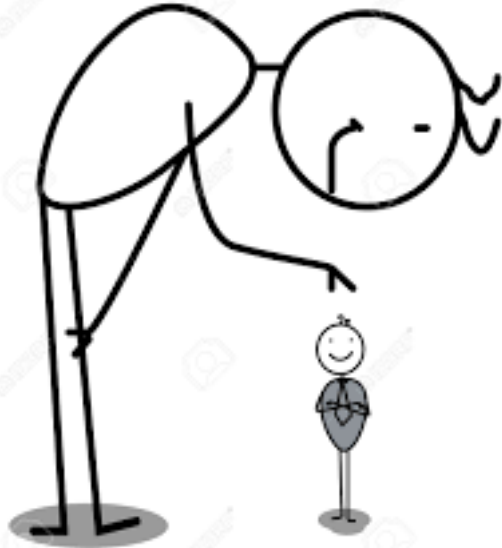
- Plan de diseño para **negociación de requisitos de sistema**.
- Medio para establecer **discusiones** con clientes, desarrolladores y administradores.
- Dos formas de usar un modelo arquitectónico:
  - Como una forma de facilitar la discusión acerca del **diseño del sistema**.
  - Como una forma de **documentar una arquitectura** que se haya diseñado.

# Niveles de abstracción de una arquitectura...



- **Arquitecturas en pequeño**
  - Se interesa por la arquitectura de **programas individuales**.
  - En este nivel nos preocupamos por la forma en que el programa individual se separa en **componentes**.

# Niveles de abstracción de una arquitectura...II



## Arquitecturas en grande

Se interesa por la arquitectura de **sistemas empresariales** complejos que incluyen otros sistemas, programas y componentes de programa.

Tarea: leer sobre arquitecturas de sistemas distribuidos

# Ventajas de documentar una arquitectura...I



La arquitectura de software es importante porque afecta el **desempeño** y la **potencia**, así como la **capacidad de distribución** y **mantenimiento** de un sistema (Bosch, 2000).

Los **requisitos no funcionales** dependen de la arquitectura del sistema, es decir, la forma cómo esos **componentes se organizan y comunican**.

# Ventajas de documentar una arquitectura...II

**Importante:** en muchos sistemas, los requisitos no funcionales **RNF** están también **influenciados** por componentes, pero no hay duda de que la **arquitectura del sistema es la influencia dominante.**



# Ventajas de documentar una arquitectura...III

Ventajas de diseñar y documentar de manera explícita la arquitectura de software:

## 1. Comunicación con los participantes.

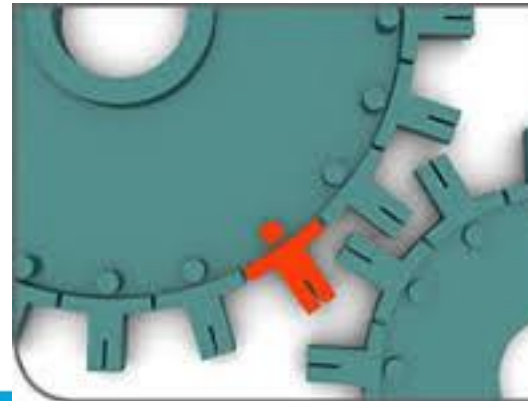
- La arquitectura es una **presentación de alto nivel del sistema.**
- La arquitectura diseñada puede usarse como un enfoque para discusión de un amplio número de **participantes.**



# Ventajas de documentar una arquitectura...IV

## 2. Análisis del sistema,

- En etapas tempranas del desarrollo requiere de cierto **análisis**.
- Las decisiones de diseño arquitectónico tienen un efecto profundo sobre si el sistema puede o no cubrir requisitos críticos como:
  - **Rendimiento**
  - **Fiabilidad**
  - **Mantenibilidad**





# Ventajas de documentar una arquitectura...V

## 3. Reutilización a gran escala,

- Un modelo de una arquitectura de sistema es una **descripción corta y manejable de cómo se organiza un sistema y cómo interoperan sus componentes.**
- Por lo general es la misma para sistemas con **requisitos similares** y, por lo tanto, puede soportar la reutilización de software a gran escala.
  - Por ejemplo: Arquitecturas de línea de productos, mirar ejemplo capítulo 16.



# Acerca de la documentación, tener en cuenta...



- **A veces es necesario documentar la arquitectura** haciendo uso de una notación semántica bien definida para su descripción.
- Algunas personas consideran que la documentación exhaustiva y detallada ni es útil ni vale realmente la pena el costo de su desarrollo.





# Decisiones en el diseño arquitectónico...

# Decisiones en el diseño arquitectónico...I

- Es el **proceso creativo** en el cual se diseña una organización del sistema que **cubrirá los requisitos funcionales y no funcionales**.
- Es mejor pensar en el diseño arquitectónico como **un conjunto de decisiones a tomar** en vez de una secuencia de actividades.



# Decisiones en el diseño arquitectónico...II



- Los arquitectos del sistema deben considerar las siguientes preguntas fundamentales:

1. ¿Existe alguna **arquitectura de aplicación genérica** que actúe como plantilla para el sistema que se está diseñando?
2. ¿Cómo **se distribuirá el sistema** a través de algunos núcleos o procesadores?
3. ¿Qué **patrones o estilos arquitectónicos** pueden usarse?
4. ¿Cuál será el enfoque fundamental usado para **estructurar** el sistema?

# Decisiones en el diseño arquitectónico...III



5. ¿Cómo los **componentes estructurales** en el sistema se separarán en **subcomponentes**?
6. ¿Qué estrategia se usará para **controlar la operación de los componentes** en el sistema?
7. ¿Cuál organización arquitectónica es mejor para entregar los **requisitos no funcionales** del sistema?
8. ¿Cómo se **evaluará** el diseño arquitectónico?
9. ¿Cómo se documentará la arquitectura del sistema?

Preguntas 4-6 tratan de resolver las dudas acerca del estilo o patrón a utilizar.

# Formato para la descripción de una decisión arquitectónica

- Aspectos del diseño
- Resolución
- Categoría
- Suposiciones
- Restricciones
- Alternativa
- Argumentos
- Implicaciones
- Decisiones relacionadas
- Preocupaciones relacionadas
- Productos finales
- Notas



# Géneros de arquitectura basados en software (Grady Booch, 2008)

- Inteligencia artificial
- Comerciales y no lucrativos
- Comunicaciones
- Contenido de autor
- Dispositivos
- Entretenimiento y deportes
- Financieros
- Juegos
- Gobierno
- Industrial
- Legal
- Médicos
- Militares
- Sistemas operativos
- Plataformas
- Científicos
- Herramientas
- Transporte
- Utilidades



# Requisitos no Funcionales Interesantes para Arquitectura

# Escalabilidad

- La habilidad para soportar la calidad de servicio requerida conforme la carga aumenta.
- habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.
- Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman, 2002)

Ej. Sistema de profesores de planta y luego profesores visitantes

Tipos:

Vertical, horizontal.

# Requisitos de sistema NO funcionales...I

- **Rendimiento:**

- Diseño de arquitectura específica con operaciones organizadas en componentes y **desplegados en toda la computadora en vez de distribuidos por la red**. Usar componentes grandes en vez de pequeños y de grano fino para disminuir el número de comunicaciones entre componentes.



# Requisitos de sistema NO funcionales...

- Seguridad:
  - Estructura en capas para la arquitectura, **con los activos más críticos protegidos en capas más internas** y con **alto grado de validación** de seguridad aplicado a dichas capas.



# Requisitos de sistema NO funcionales...II

- **Mantenibilidad:**

Diseñar usando **componentes auto contenidos de grano fino** que puedan **cambiarse con facilidad**. Los productores de datos tienen que separarse de los consumidores y hay que evitar compartir estructuras de datos.



# Requisitos de sistema NO funcionales...II

- **Disponibilidad:**

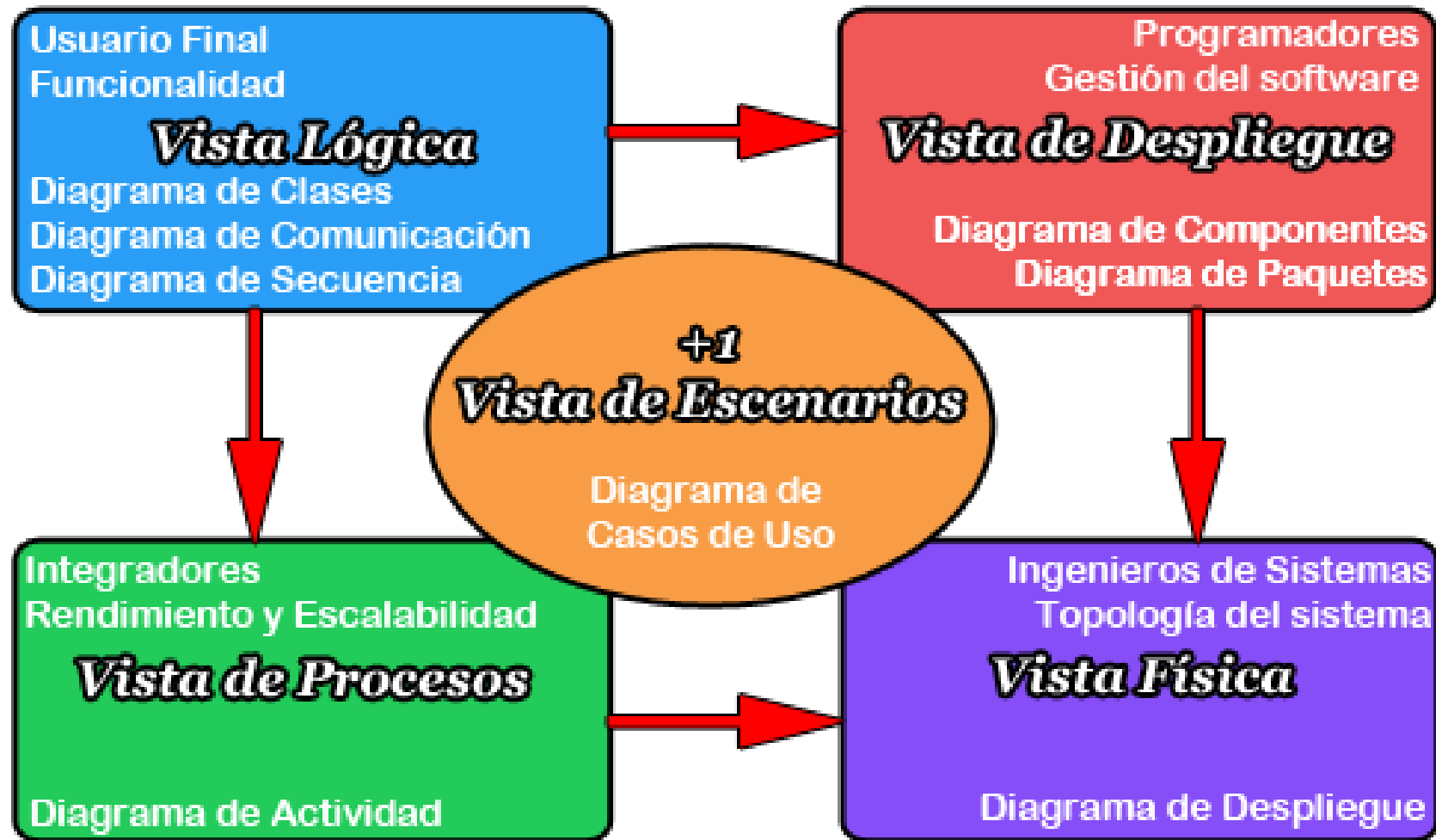
Incluir **componentes redundantes**, de manera que sea posible sustituir y actualizar componentes sin detener el sistema. [ver capítulo 13 Ian Sommerville].



# Vistas arquitectónicas...I

- ¿Qué **vistas o perspectivas** son útiles al diseñar y documentar una arquitectura del sistema?
- ¿Qué **notaciones** deben usarse para describir modelos arquitectónicos?







# Vistas arquitectónicas...II

**Modelo de vista 4+1 (Krutchen, 1995):** debe existir cuatro vistas fundamentales que se relacionan usando casos de uso o escenarios. Las vistas que él sugiere son:

## 1. Una vista lógica

Que indique las **abstracciones clave** en el sistema como **objetos o clases** de objeto.



# Vistas arquitectónicas...IV

## 4. Una vista física

Expone **el hardware y el software** del sistema y cómo los componentes de software se distribuyen a través de los procesadores en el sistema.

- **Una vista conceptual**

Por ejemplo, **la arquitectura de un robot de empacado.**



# Vistas arquitectónicas...III

## 2. Una vista de proceso

Vista útil para hacer juicios acerca de las **características no funcionales**, como el **rendimiento y la disponibilidad**.

## 3. Una vista de desarrollo

Muestra cómo se **descompone el software** en elementos que se implementan mediante un solo desarrollador o equipo de desarrollo.



# Estructura o estructuras y tipos

Código

Procesos, Hilos

Asignación de archivos en disco

Máquinas

Componentes

**Cada estructura presenta una perspectiva desde la cual se puede  
razonar acerca de la arquitectura**



# ¿Usar UML?



- Hay diversas opiniones:
  - Se aplica en una versión holgada e informal (Lange et al., 2006).
    - **Es incorrecto**, los objetos están más cerca de la implementación.
  - El UML es demasiado **formal**, se requieren notaciones mucho más rápidas.
    - **Seguirá siendo la notación de uso común.**
  - LDA (ADL) Lenguajes de descripción arquitectónicas (**Bass** et al., 2003).
    - Son muy complejos de entender y aplicar.

# Marcos ágiles

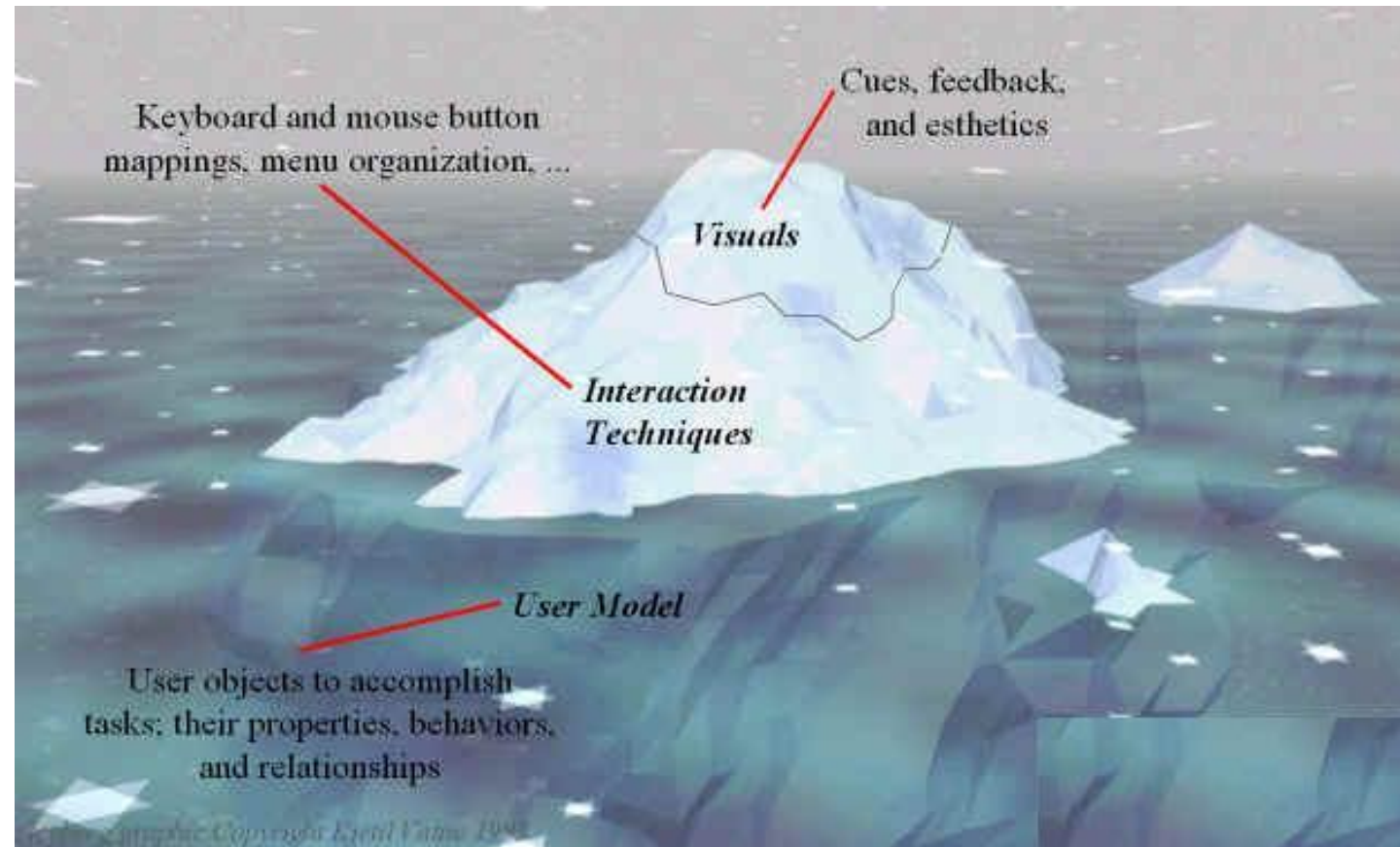
- No se usa la documentación detallada de diseño.
- Desarrollarla es un desperdicio de tiempo y dinero.
- Ian Sommerville está de acuerdo y considera que para la mayoría de sistemas, **no vale la pena desarrollar una descripción arquitectónica** detallada desde las 4 perspectivas.

**Es necesario las vistas que sean necesarias y útiles para la comunicación.**





# Usabilidad y Arquitectura





# Escalabilidad en Netflix

**¿Qué ocurre en AWS antes de que le des al play?**

**Vamos a la lectura:** <https://www.xataka.com/streaming/la-compleja-infraestructura-detras-de-netflix-que-pasa-cuando-le-das-al-play>

# Referencias

- Sommerville, Ian. *Ingeniería del software*. Pearson Educación, Novena Edición, 2011.
- Notas de clase profesor Juan Bernardo Quintero y Lenin Lozano
- Carlos Billy Reynoso. Introducción a la Arquitectura de Software. Universidad de Buenos Aires <http://www.willydev.net/descargas/prev/IntroArq.pdf>
- Paul Reed. Reference Architecture: The best of best practices. IBM
- <http://www.ibm.com/developerworks/rational/library/2774.html>
- Amit Unde, Becoming an Architect in a System Integrator. Microsoft Corporation.
- <http://msdn.microsoft.com/en-us/architecture/cc505970.aspx>

# Referencias

- Software's architecture handbook.
- Pattern-Oriented Software Architecture, A System of Patterns, Volume 1, Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, Wiley & Sons, 1996, ISBN 0 471 95869
- Patterns-Oriented Software Architecture, Volume 2 : Concurrent and Networked Objects, Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann, 2000).
- Vitalie Temnenco. TOGAF or not TOGAF: Extending Enterprise Architecture beyond RUP. IBM
- <http://www.ibm.com/developerworks/rational/library/jan07/temnenco/index.html>
- Adrián Lasso. Arquitectura de Software. Microsoft Corporation.

# Referencias

- <http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art110.asp>
- Michael Platt. Microsoft Architecture Overview. Microsoft Corporation.
- <http://msdn.microsoft.com/architecture/overview/default.aspx?pull=/library/en-us/dnea/html/eaarchover.asp>.
- Carlos Reynoso y Nicolás Kicillof. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Universidad de Buenos Aires.  
[http://www.microsoft.com/spanish/msdn/arquitectura/roadmap\\_arq/lenguaje.asp](http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/lenguaje.asp)
- Desarrollo de Software Basado en Componentes (Luis F. Iribarne Martinez, U. de Almerias, Tesis Doctoral, Capítulo 1 )
- Interoperabilidad e Integración, Forum de Desarrolladores Corporativos, Madrid, Diciembre del 2002.
- [http://www.unilibre.edu.co/revistaavances/avances\\_10/r10\\_art1.pdf](http://www.unilibre.edu.co/revistaavances/avances_10/r10_art1.pdf)

<https://www.youtube.com/watch?v=V1dluXlu8P4>