FICH UNL
Facultad de Ingeniería y Ciencias Hídricas
UNIVERSIDAD NACIONAL DEL LITORAL

Ingeniería de software II

Diseño arquitectónico

Diseño arquitectónico

- Sistema simple

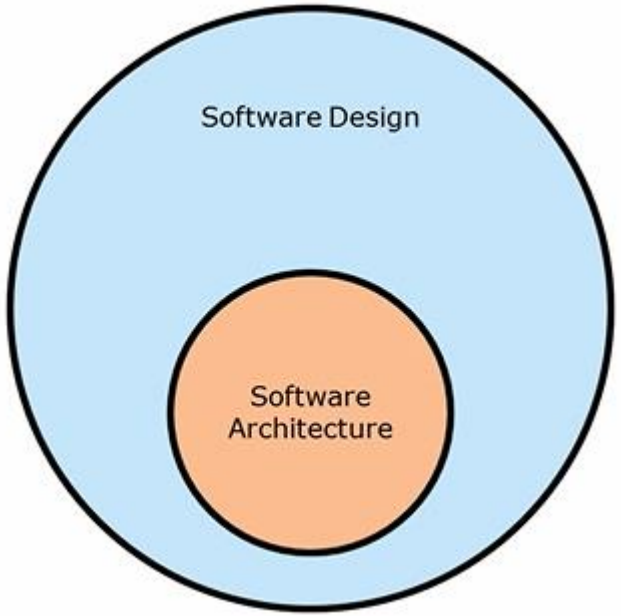
- Sistema complejo

Modelar los detalles relevantes
para minimizar el riesgo de fallas

Arquitectura de software

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

-- Clements, Bass, Kazman 2012



Los sistemas complejos están compuestos por subsistemas que interactúan bajo el control de un diseño de sistema

Arquitectura de software

- Arquitectura
  - es un conjunto de estructuras de software.
  - es una abstracción.
  - todo sistema tiene una arquitectura.
    - Debe ser conocida por todos.
    - No todas las arquitecturas son buenas arquitecturas.
  - Componentes:
    - Los subsistemas que componen el sistema
    - las interfaces
    - las reglas de interacción entre ellos
  - Ventajas:
    - Comunicación entre stakeholders
    - Decisiones tempranas de diseño
    - Reutilización

Arquitectura

- ¿Por qué la arquitectura es importante?
  - Inhibe o habilita los atributos de calidad de un sistema
    - Predicciones tempranas
  - Permite razonar y administrar los cambios a medida que el sistema evoluciona.
  - Su documentación facilita la comunicación entre los stakholders.

Arquitectura

- ¿Por qué la arquitectura es importante?
  - Decisiones de diseño tempranas y fundamentales.
  - Restricciones de la implementación.
  - Base para un prototipado que evoluciona.
  - Razonamiento sobre costos y cronograma.
  - Modelo transferible
  - Modelo reutilizable
  - Ensamblado de componentes.
  - Creatividad de desarrolladores
    - Reduce la complejidad del sistema y de su diseño.
  - Posibilita el entrenamiento de un nuevo miembro del equipo.

Diseño arquitectónico

- La medida en que un sistema alcanza sus requisitos de calidad depende de las decisiones de arquitectura:
  - la arquitectura es crítica para alcanzar los atributos de calidad;
  - las cualidades del producto deben diseñarse como parte de la arquitectura;
  - un cambio en la estructura que mejora un atributo de calidad suele afectar los otros atributos;
  - la arquitectura sólo puede permitir, no garantizar, que cualquier requisito de calidad se alcance.

Arquitectura

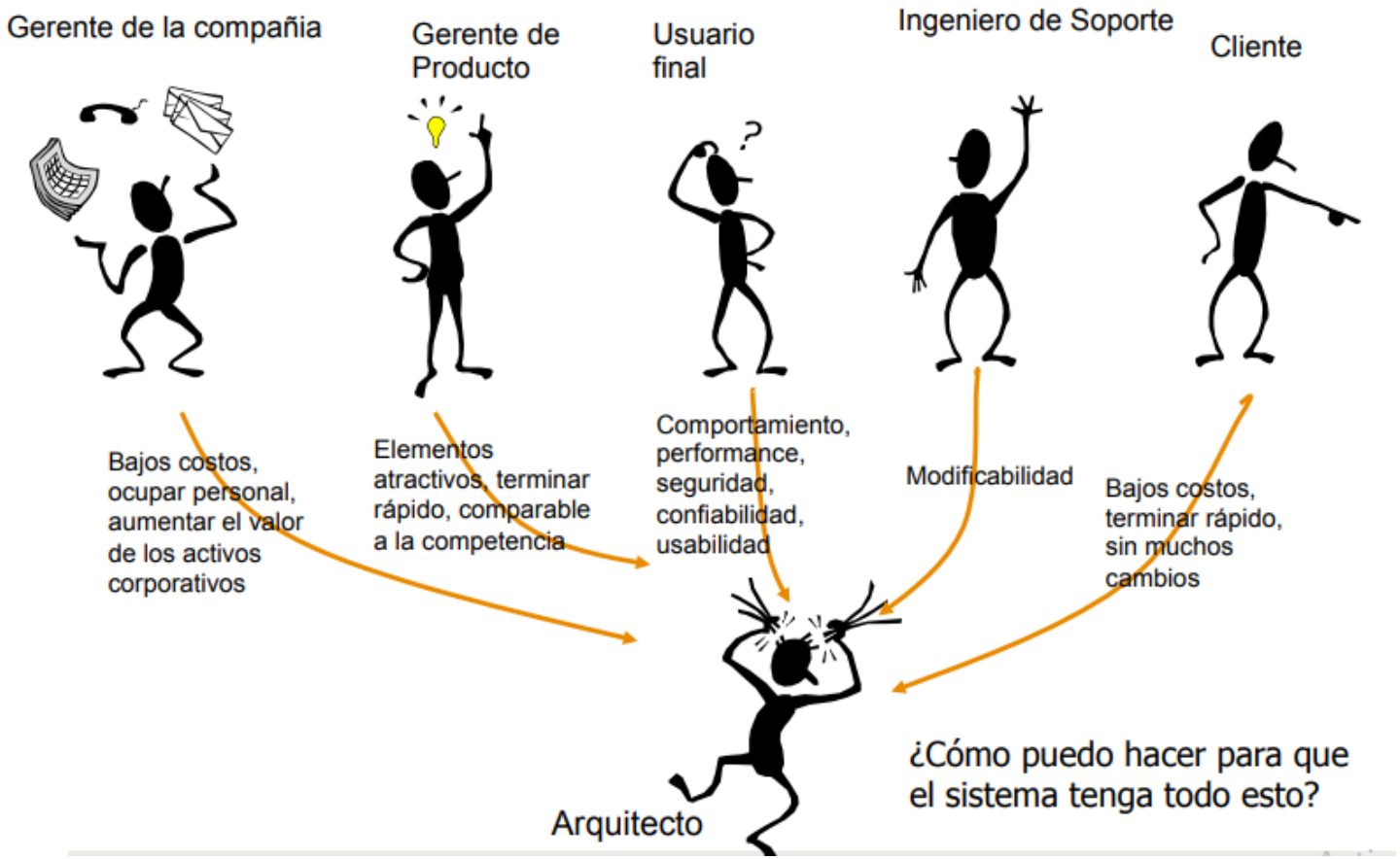
- Desafíos:
  - ¿Qué significan con precisión atributos de calidad?
  - ¿Cómo se estructura el sistema para obtener estas cualidades deseadas?
  - ¿Se puede analizar el sistema para determinar si tiene estas cualidades?
  - ¿Cuán temprano puede realizarse este análisis?
  - ¿Cómo se sabe si una arquitectura de software es apropiada para un sistema sin tener que construir el sistema primero?

Arquitectura de software

- Performance
- Seguridad
- Disponibilidad
- Mantenibilidad
- Flexibilidad
- Portabilidad
- Reutilización
- Legibilidad
- Corrección
- Usabilidad
- Eficiencia
- Confiabilidad
- Adaptabilidad
- Facilidad de prueba
- Interoperabilidad

El estilo y estructura particular elegido para una aplicación dependen fuertemente de los requerimientos no funcionales.

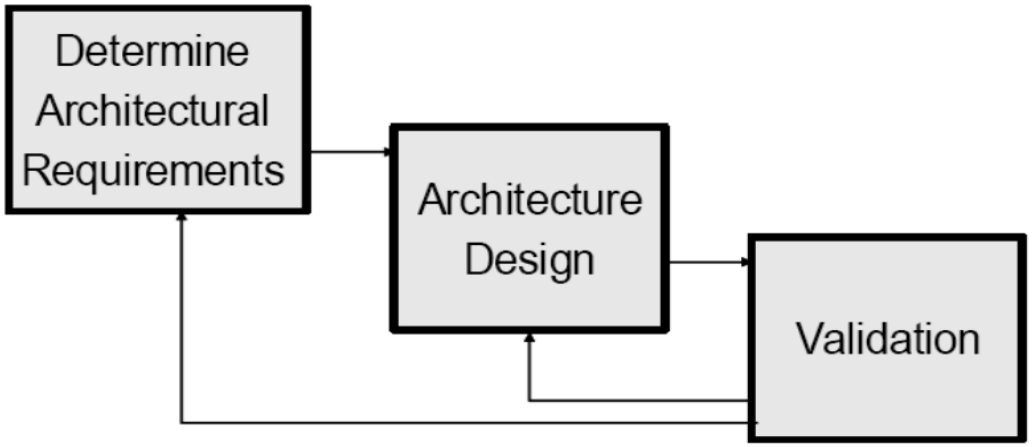
Interesados



Interesados

- Los objetivos organizacionales y las propiedades del sistema requeridas por el negocio raramente se comprenden y menos aún se articulan completamente.
- Los requisitos de calidad del cliente casi nunca se documentan, lo cual resulta en:
  - Objetivos que no se alcanzan;
  - Conflicto inevitable entre los interesados.
  - Los arquitectos deben identificar e involucrar activamente a los interesados de modo de:
    - comprender las restricciones reales del sistema;
    - administrar las expectativas de los interesados;
    - negociar las prioridades del sistema;
    - tomar decisiones de compromiso.

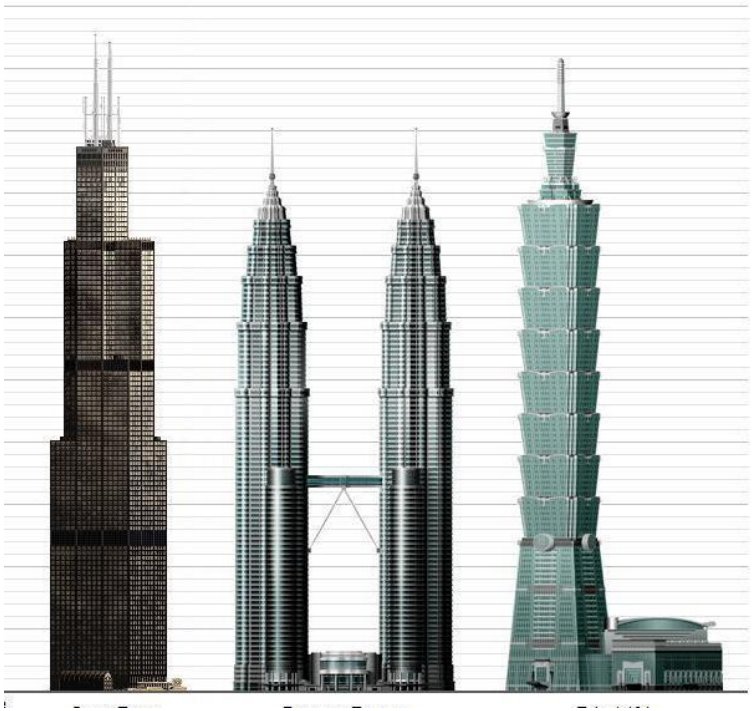
Proceso de modelado arquitectónico



- Definir los requerimientos: Crear un modelo desde los requerimientos basado en los atributos de calidad esperados
- Diseño de la Arquitectura : Definir la estructura y las responsabilidades de los componentes
- Validación: Probar la arquitectura con los requerimientos actuales y cualquier posible requerimiento a futuro.

¿Qué tan Fácil es Modificarla?

- Sears
  EEUU
  527 metros
- Petronas
  Malasia
  452 metros
- Taipei 101
  China
  508 metros



Patrones arquitectónicos

- Patrón arquitectónico:
  - colección de decisiones de diseño arquitectónico
    - con nombre específico
    - aplicables a problemas de diseño recurrentes
    - parametrizadas
  - para diferentes contextos de desarrollo de software.

  - Decisiones de diseño efectivas para organizar ciertas clases de sistemas de software.
    - “configurables”
    - necesitan ser instanciadas con los componentes y conectores particulares a una aplicación.

Patrones arquitectónicos

- Propósito
  - Compartir una solución probada, ampliamente aplicable a un problema particular de diseño.
    - Se presenta en una forma estándar
    - Fácilmente reutilizado.

- Establece las relaciones entre:
  - Un Contexto
  - Un Problema (atributos de calidad)
  - Una Solución (con abstracción apropiada):
    - Un conjunto de elementos
    - Un conjunto de mecanismos de interacción o conectores
    - Diseño topológico de los componentes
    - Un conjunto de restricciones semánticas

Patrones arquitectónicos

- Patrón en capas
  - Contexto:
    - Sistema complejo que requiere el desarrollo de partes de manera independiente.
    - Requieren la separación clara y bien documentada de los aspectos involucrados de manera que los módulos puedan ser desarrollados y mantenidos independientemente.
  - Problema:
    - **El software requiere ser segmentado en módulos a desarrollar por separado con mínima interacción entre las partes**.
    - Soporta portabilidad, modificabilidad y reutilización.
  - Solución:
    - Divide el software en unidades denominadas capas.
    - Cada capa agrupa a módulos que ofrecen un conjunto cohesivo de servicios.
    - La relación entre las capas debe ser unidireccional (allowed-to-use).

Patrones arquitectónicos

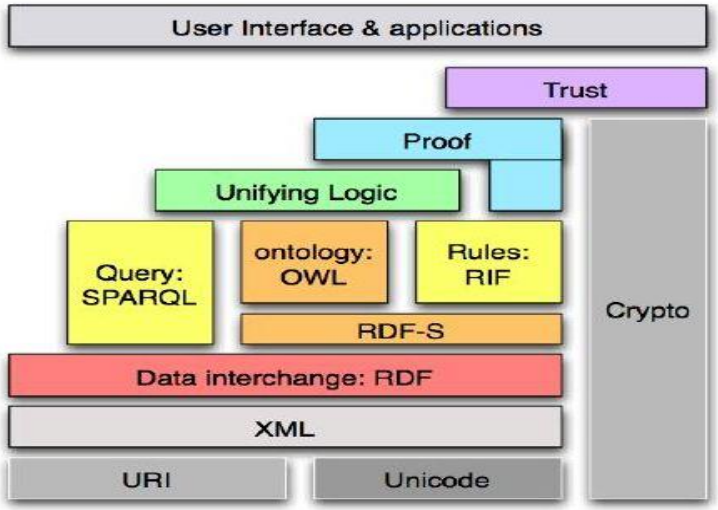
- Patrón en capas
  - Modelo estricto: una capa sólo utiliza servicios de la inmediata inferior.
  - Modelo relajado: se pueden “saltar” capas.
  - Definición de protocolos mediante los que interactúan las capas
  - Ventajas
    - Facilita la comprensión
    - Facilita mantenimiento
    - Facilita reutilización
    - Facilita portabilidad
  - Desventajas
    - No siempre es fácil estructurar en capas ni identificar los niveles de abstracción a partir de los Requerimientos.

Patrones arquitectónicos

- Patrón en capas
  - Ejemplo: Web de datos

Compatibilidad descendente

- Agentes con el conocimiento propio de una capa deberian también interpretar y usar información escrita en niveles inferiores (máximo provecho)



Patrones arquitectónicos

- Patrón en capas
  - Ejemplo: Web de datos

Compatibilidad descendente

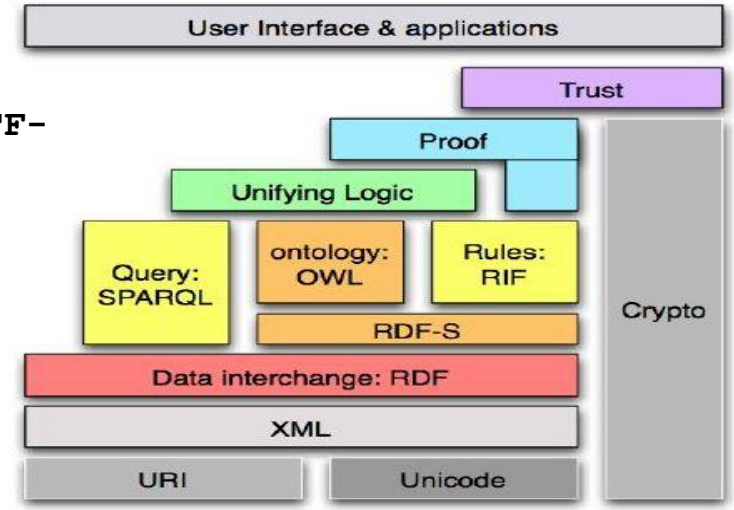
- Agentes con el conocimiento propio de una capa deberian también interpretar y usar información escrita en niveles inferiores (máximo provecho)
  - Capa XML
    - Base sintáctica.

```
<?xml version="1.0" encoding="UTF-8"?>

<correo>

    <para>Pepe</para>
    <remitente>Lola</remitente>
    <titulo>Hola</titulo>
    <mensaje>¿Todo bien?</mensaje>

</correo>
```



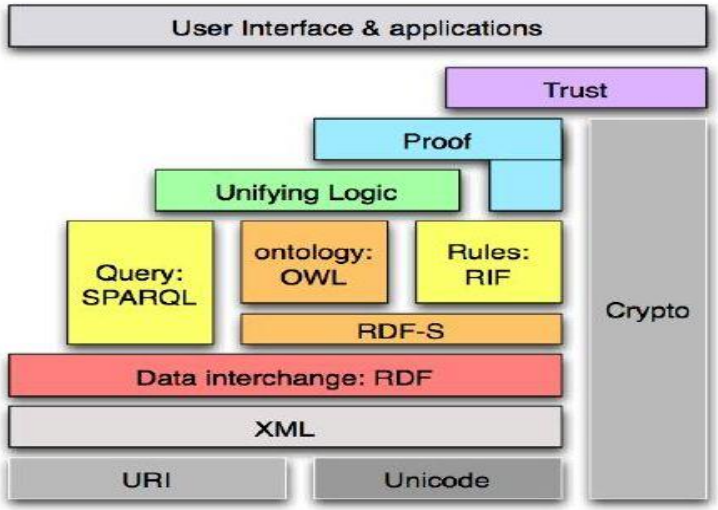
Patrones arquitectónicos

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:rdfs=http://www.w3.org/2000/01/rdf-
schema#
xml:base == "http://www.turismo.com/alojamientos#">
        <rdfs:Class rdf:ID="alojamiento" /rdfs:class>
        <rdfs:Class rdf:ID="hotel">
                <rdfs:subClassOf
rdf:resource="#alojamiento"/>
        </rdfs:Class>
</rdf:RDF>
```

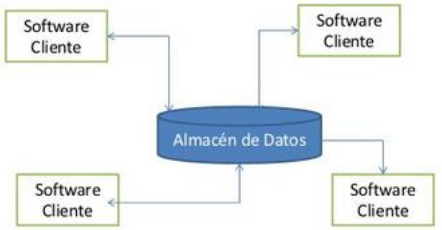
- Patrón en capas
  - Ejemplo: Web de datos

Compatibilidad descendente

- Agentes con el conocimiento propio de una capa deberian también interpretar y usar información escrita en niveles inferiores (máximo provecho)
  - Capa XML
    - Base sintáctica.
  - Capa RDF-Schema
    - Provee primitivas de modelado para la organzacion de recursos en jerarquias: clases, propiedades, relaciones de subclases y subpropiedades y restricciones de dominio y rango.
    - Esta basado en RDF
    - Lenguaje primitivo para la definicion de ontolgias

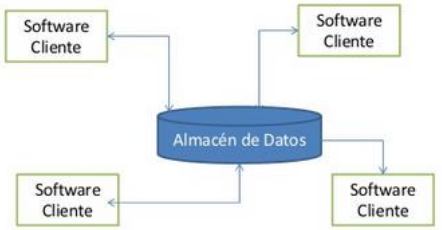


Patrones arquitectónicos



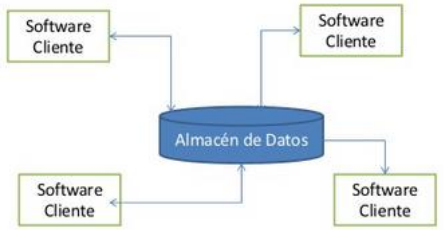
- Patrón centrado en datos
  - Contexto:
    - Varios componentes necesitan compartir y manipular grandes cantidades de datos.
    - Esos datos no pertenecen exclusivamente a ninguno de los componentes.
  - Problema:
    - ¿Cómo pueden los sistemas almacenar y manipular datos persistentes que son accedidos por múltiples componentes independientes?
  - Solución:
    - La interacción está dominada por el intercambio de datos persistentes entre múltiples entidades que acceden a esos datos y al menos un almacenamiento de datos compartido.
    - El tipo de conector es lectura y escritura.

Patrones arquitectónicos



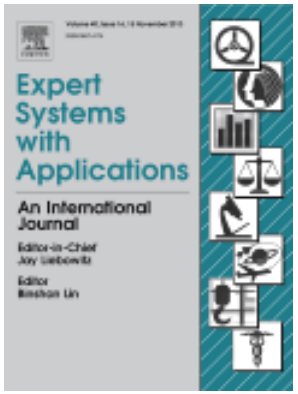
- Patrón centrado en datos
  - Ventajas:
    - Promueve la capacidad de integración.
    - Es posible cambiar componentes existentes y agregar nuevos componentes cliente a la arquitectura, los componentes clientes operan en forma independiente.

Patrones arquitectónicos



- Patrón centrado en datos
  - Relaciones:
    - relación attachment determina a qué componente se conecta con qué almacenamiento de datos.
  - Restricciones:
    - Los componentes que acceden a los datos interactúan con el almacenamiento de datos
  - Debilidad:
    - El almacenamiento de datos puede ser un cuello de botella.
    - Productores y consumidores de datos pueden estar estrechamente acoplados.

Patrones arquitectónicos



- Patrón centrado en datos
  - Ejemplo:

SNRD
Sistema Nacional
de Repositorios Digitales

OPEN ACCESS

Tipo de recurso: revista

Autor:

Título de la revista: Expert Systems with applicat

Edición: 2da

Palabras claves: Artificial Inteligence, expert systems,
rule based razoning

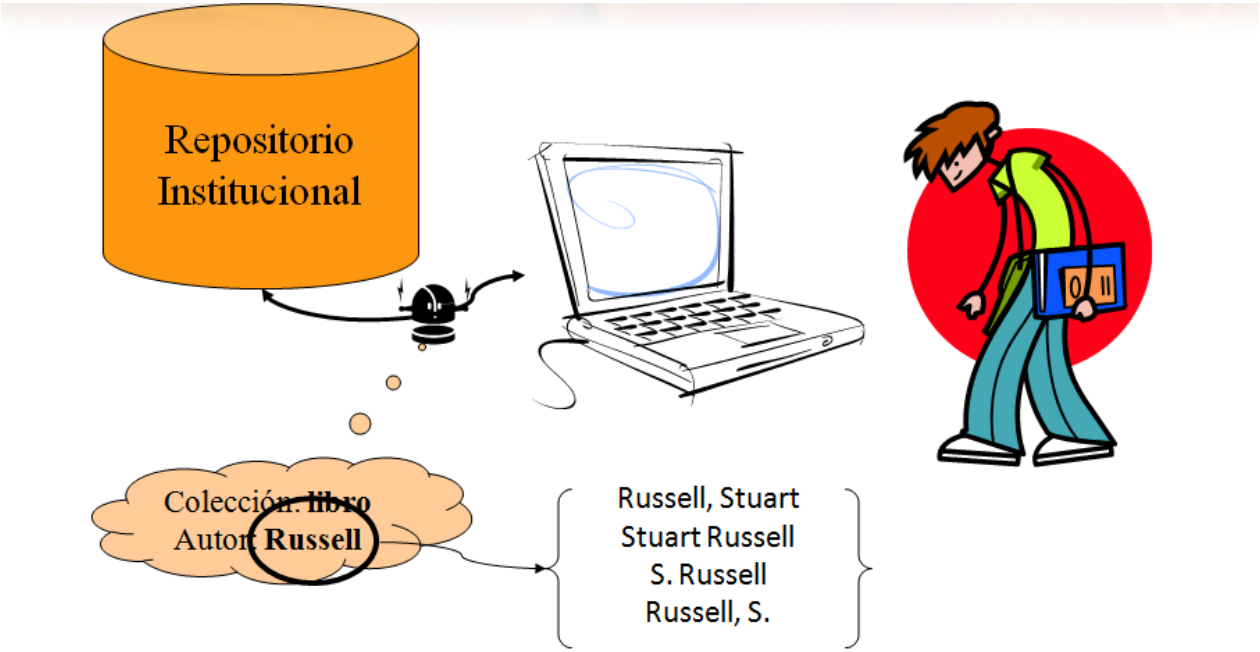
....

Patrones arquitectónicos

- Patrón centrado en datos
  - Ejemplo:

SNRD
Sistema Nacional
de Repositorios Digitales

OPEN ACCESS



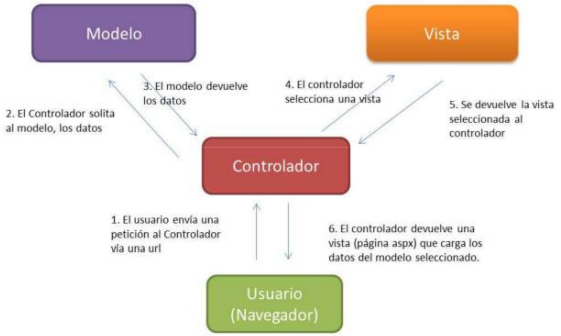
Patrones arquitectónicos

- Patrón centrado en datos
  - Ejemplo:

SNRD
Sistema Nacional
de Repositorios Digitales

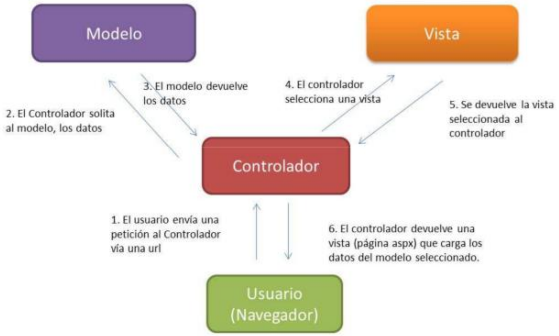
OPEN ACCESS

Patrones arquitectónicos



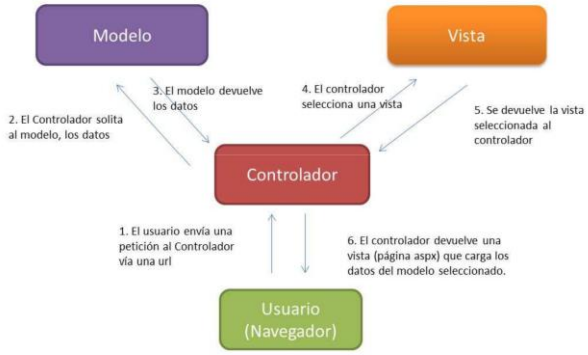
- Patrón modelo-vista-controlador
  - Contexto:
    - Software para IU es la porción modificada con mayor frecuencia de una aplicación interactiva.
    - Es recomendable mantener las modificaciones de la IU separada del resto del sistema.
  - Problema:
    - ¿Cómo puede mantenerse separada la funcionalidad de IU de la funcionalidad de la aplicación y seguir siendo sensible al ingreso del usuario, o a los cambios en la aplicación de datos subyacente?
  - Solución: El patrón MVC separa la funcionalidad de la aplicación en:
    - Un **modelo**, que contiene los **datos de la aplicación**
    - Una **vista** que muestra parte de los datos subyacentes y que **interactúa con el usuario**
    - Un **controlador**, que hace de intermediario entre el modelo y la vista y **administra los cambios en el estado y sus notificaciones.**

Patrones arquitectónicos



- Patrón modelo-vista-controlador
  - Contexto:
    - Software para IU es la porción modificada con mayor frecuencia de una aplicación interactiva.
    - Es recomendable mantener las modificaciones de la IU separada del resto del sistema.
  - Relaciones:
    - La relación *notifies* conecta instancias del modelo, vista y controlador notificando elementos de cambios de estado relevantes.
  - Restricciones:
    - Debe haber por lo menos una instancia de modelo, una de vista y una de controlador.
  - Debilidad:
    - La complejidad puede no valer la pena para interfaces de usuario simples.

Patrones arquitectónicos

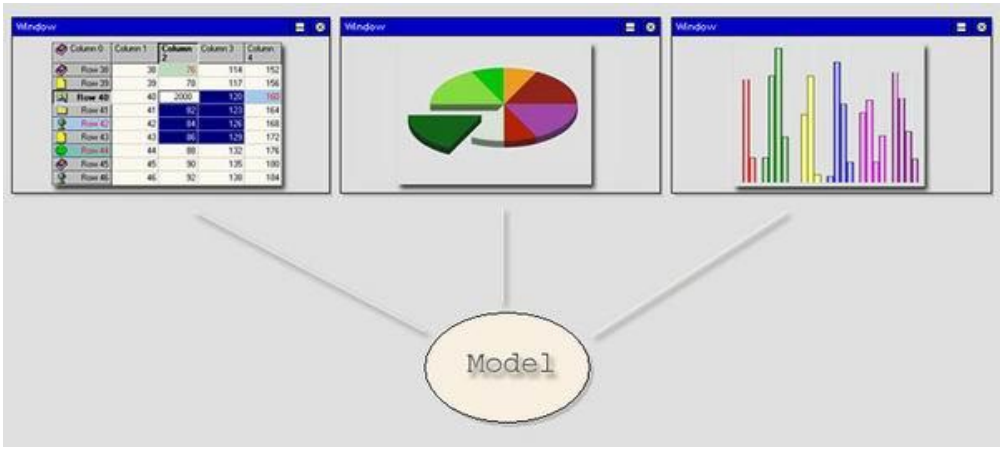


- Patrón modelo-vista-controlad
  - Ventajas:
    - Se presenta la misma información de distintas formas.
    - Las vistas y comportamiento de una aplicación deben reflejar las manipulaciones de los datos de forma inmediata.
    - Debería ser fácil cambiar la interfaz de usuario (incluso en tiempo de ejecución).
    - Permitir diferentes estándares de interfaz de usuario o portarla a otros entornos no debería afectar al código de la aplicación.

Patrones arquitectónicos

- Patrón modelo-vista-controlador
  - Ejemplo:
    - Los datos de una hoja de cálculo pueden mostrarse de en formato tabular, con un gráfico de barras, con uno de torta.
    - Los datos son el modelo.
    - Si cambia el modelo, las vistas deberían actualizarse en consonancia.
    - El usuario manipula el modelo a través de las vistas.

      (en realidad, a través de los controladores)



Patrones arquitectónicos

- Otras arquitecturas:
  - Pipeline
  - Orientada a servicios
  - Dirigida por eventos
  - Cliente servidor

Evaluación de Arquitecturas

- Cambiar la arquitectura de un producto ya construido requiere mucho esfuerzo
- Es importante evaluar la arquitectura antes de implementarla completamente
  - Verificar los requisitos de calidad establecidos
- Evaluaciones a posteriori resultan útiles como forma de aprendizaje y estudio de posibilidades de mejora
  - por ej. para una nueva versión del producto
- Software Engineering Institute (SEI) propone:
  - Architecture Tradeoff Analysis Method (ATAM)
  - Software Architecture Analysis Method (SAAM)

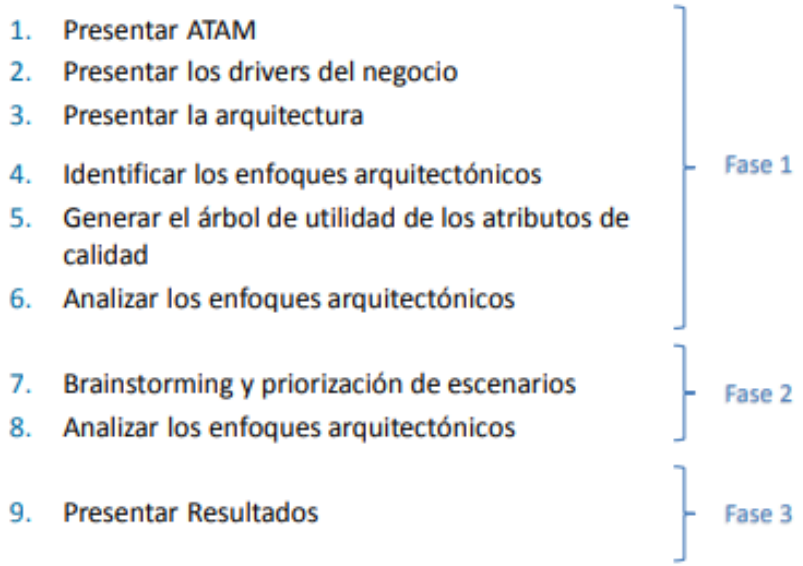
Evaluación de Arquitecturas

- Architecture Tradeoff Analysis Method
  - Establece en qué medida una arquitectura particular satisface las metas de calidad
  - Provee ideas de cómo esas metas de calidad interactúan entre ellas, com,o realizan concesiones.

Fase	Descripción	Participantes	Duración Típica
0 – Partnership y Preparación	Logística, planificación, reclutamiento de stakeholders, formación de equipos.	Líder del equipo de evaluación y tomadores de decisiones claves del proyecto	Procede informalmente por demanda, quizás a lo largo de unas pocas semanas
1 – Evaluación	Pasos del 1 a 6	El equipo de evaluación y los tomadores de decisiones claves del proyecto	1 o 2 días seguidos por un intervalo de 2 o 3 semanas
2 – Evaluación	Pasos del 7 a 9	El equipo de evaluación, los tomadores de decisiones claves del proyecto y los stakeholders	2 días
3 – Seguimiento	Generación y entrega de reporte, y mejora de procesos	El equipo de evaluación y el cliente de la evaluación	1 semana

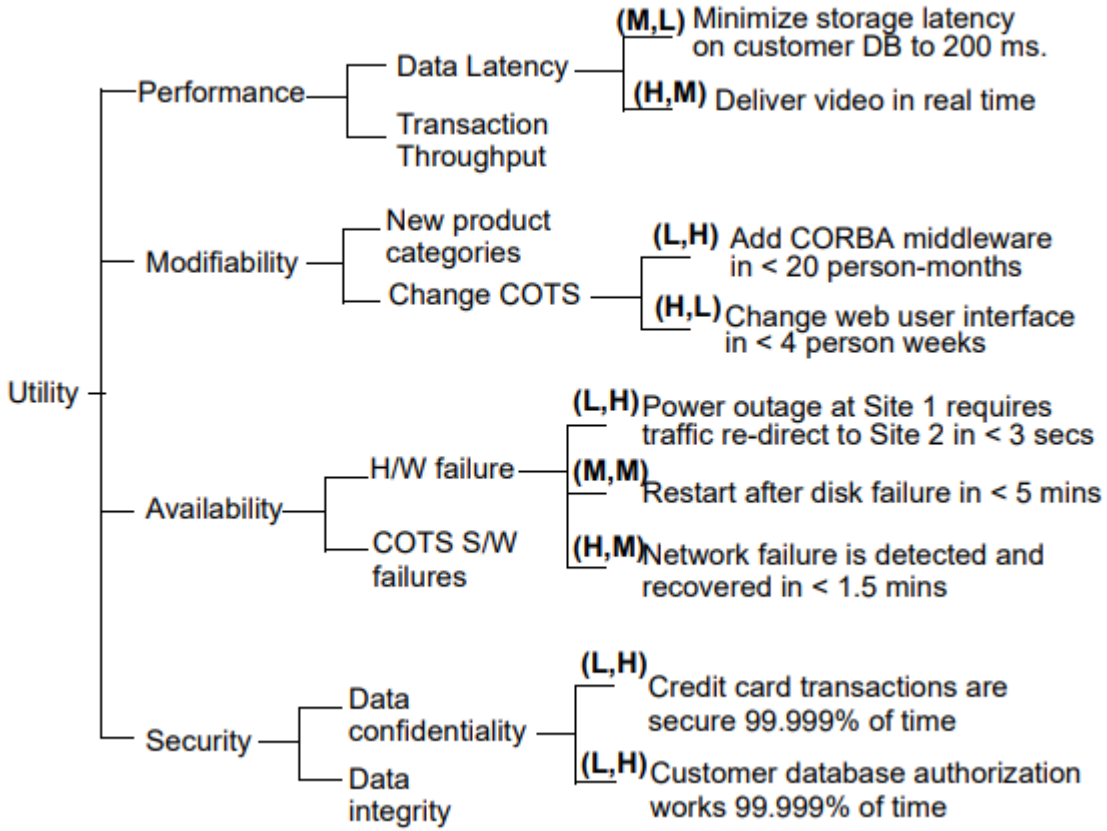
Evaluación de Arquitecturas

- Architecture Tradeoff Analysis Method
  - Establece en qué medida una arquitectura particular satisface las metas de calidad
  - Provee ideas de cómo esas metas de calidad interactúan entre ellas, com,o realizan concesiones.



ATAM – Fase 1 – Pasos

5. Generar árbol de utilidad de los atributos de calidad



ATAM – Fase 1 – Pasos

6. Analizar propuestas arquitectónicas

Escenario 1.

El servidor sufre una falla y deja de funcionar, el sistema debería estar en funcionamiento en un servidor de contingencia en al menos 3 minutos.

- **R1:** Si el servidor donde se encuentra el sistema sufre una falla dejando de funcionar sería un riesgo si no se dispone de un servidor de contingencia con el mismo ambiente del servidor principal.