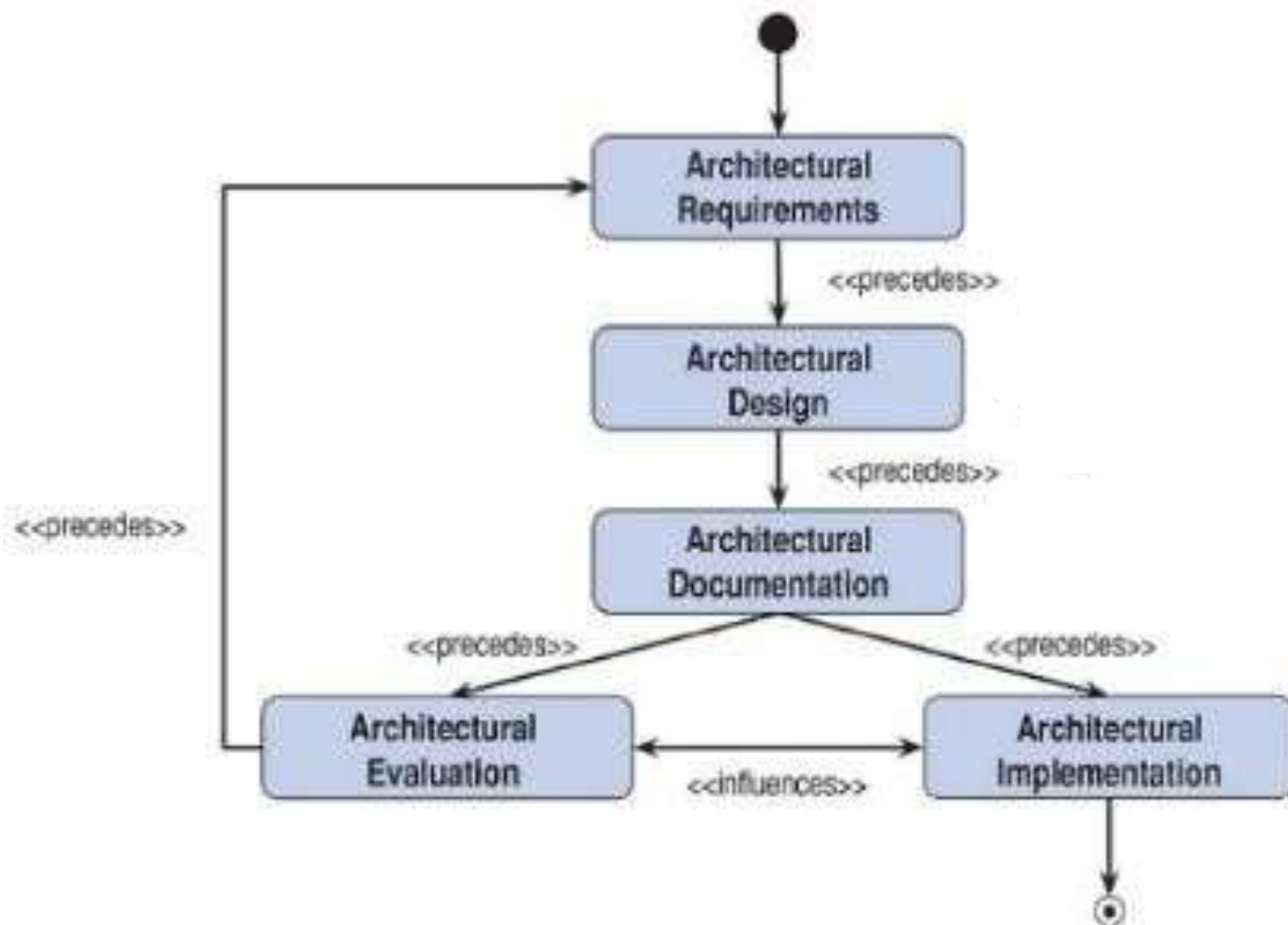


# Atributos de Calidad y decisiones para el diseño de la Arquitectura

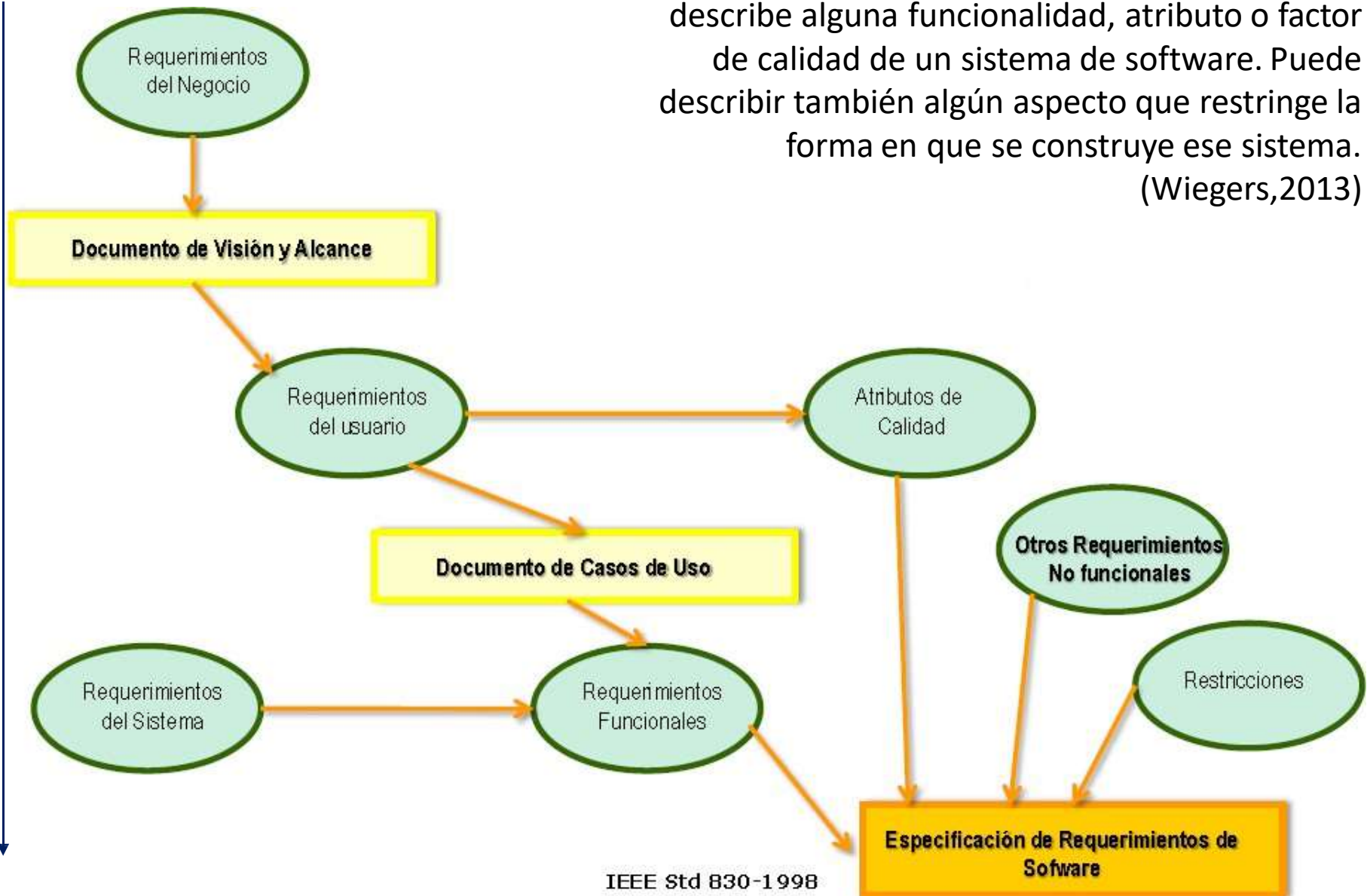
# Arquitectura de Software

- La arquitectura es la organización fundamental de un sistema y sus componentes, las relaciones entre ellos y con el entorno, así como los principios que guían su diseño y evolución. (IEEE Std 1471-2000)
  - Cada sistema de software tiene una arquitectura así esté documentada o no. (Bass , 2003)
- Importancia de la arquitectura (Bass , 2003):
  - La arquitectura es el vehiculo común de comunicación con los stakeholders
  - La arquitectura manifiesta la decisiones tempranas de diseño





Requerimiento es una especificación que describe alguna funcionalidad, atributo o factor de calidad de un sistema de software. Puede describir también algún aspecto que restringe la forma en que se construye ese sistema. (Wiegers,2013)



IEEE Std 830-1998

# ACTIVIDADES

- Ejemplos de cómo llegan los requerimientos al área de TI.
- Ejemplos de Requerimientos del Negocio
- Ejemplos de requerimientos del Usuario
- Ejemplos de RF
- Ejemplos de Restricciones
- Ejemplos de RNF

# Atributos de Calidad

- Cualificaciones sobre funcionalidades o sobre el sistema
- “Propiedad medible y comprobable de que tan bien un sistema satisface las necesidades de los interesados”
- Ejemplos:
  - Sobre funcionalidad: Qué tan rápido es la función, o qué tan segura es.
  - Sobre sistema: Qué tan rápido es desplegar el sistema, o agregar componentes.

# Atributos de Calidad (2)

- Las categorías mas mencionadas en la literatura:
  - Disponibilidad
  - Seguridad
  - Desempeño
  - Facilidad de Prueba
  - Modificabilidad
  - Usabilidad
  - Interoperabilidad



Hay relación entre lo funcional y los atributos de calidad, más estrecha de lo que se podría creer



¿ Qué tan fácil se puede aprender?

¿ Qué tan rápido aparecerá?

¿ Con qué frecuencia fallará?

**DESEMPEÑO**



**USABILIDAD**



**DISPONIBILIDAD**



# ¿ Son QA requerimientos funcionales ?

- Sí, pero :
  - el término no funcional sugiere una falsa partición.
  - QA no se puede describir de forma independiente de la funcionalidad
- Problemas:
  - 1) No es posible probar un atributo de calidad
  - 2) Se discute a que dominio de interés (concern) corresponde un atributo
  - 3) Cada comunidad de interés ha desarrollado su propio vocabulario → recordar a los ciegos y el elefante

Escenarios

# Taxonomías

- En general hay dos grandes grupos:
  - Los que describen características en run-time
  - Los que describen características en construcción

# Modelo de McCall

- 3 ejes o puntos de vista
- 11 factores
- 23 criterios
- 41 métricas



Factores de calidad de McCall (Fuente: Roger Pressman, 2005)

Puntos De Vista	Factor	Criterios
OPERACION DEL PRODUCTO	Facilidad de uso	Facilidad de operación Facilidad de comunicación Facilidad de aprendizaje Formación
	Integridad	Control de accesos Facilidad de auditoría Seguridad
	Corrección	Compleitud Consistencia Trazabilidad o rastreabilidad
	Fiabilidad	Precisión Consistencia. Tolerancia a fallos Modularidad Exactitud
	Eficiencia	Eficiencia en ejecución Eficiencia en almacenamiento

REVISION DEL PRODUCTO	Facilidad de mantenimiento	Modularidad. Simplicidad. Consistencia. Concisión Auto descripción
	Facilidad de prueba	Modularidad. Simplicidad. Auto descripción. Instrumentación.
	Flexibilidad	Auto descripción. Capacidad de expansión Generalidad Modularidad.

TRANSICION DEL PRODUCTO	Reusabilidad	Auto descripción. Generalidad. Modularidad. Independencia entre sistema y software Independencia del hardware
	Interoperabilidad	Modularidad. Compatibilidad de comunicaciones Compatibilidad de datos Estandarización en los datos
	Portabilidad	Auto descripción. Modularidad. Independencia entre sistema y software. Independencia del hardware.

Métrica \ Factor	Corrección	Fiabilidad	Eficacia	Integridad	Facilidad de mant.	Flexibilidad	Fac. de prueba	Portabilidad	Reusabilidad	Interoperabil.	Fac. de uso
Facilidad de auditoría				x			x				
Exactitud		x									
Normaliz de las comunic.										x	
Complejidad	x										
Complejidad		x				x	x				
Consición			x		x	x					
Consistencia	x	x			x	x					
Estandariz en los datos										x	
Tolerancia de errores		x									
Eficiencia en la ejecución			x								
Facilidad de expansión						x					
Generalidad						x		x	x	x	
Indep. del hardware								x	x		
Instrumentación				x	x		x				
Modularidad		x			x	x	x	x	x	x	
Facilidad de operación			x								x
Seguridad				x							
Auto-Documentación					x	x	x	x	x		
Simplicidad		x			x	x	x				
Indep. del sistema								x	x		
Facilidad de traza	x										

Generalmente estas métricas definidas por MacCall solo pueden ser medidas en forma subjetiva.

Las métricas pueden estar listas de comprobaciones para obtener el grado de los atributos específicos del software.

El esquema de graduación propuesto por McCall va en una escala de 0 (bajo) a 10 (alto).

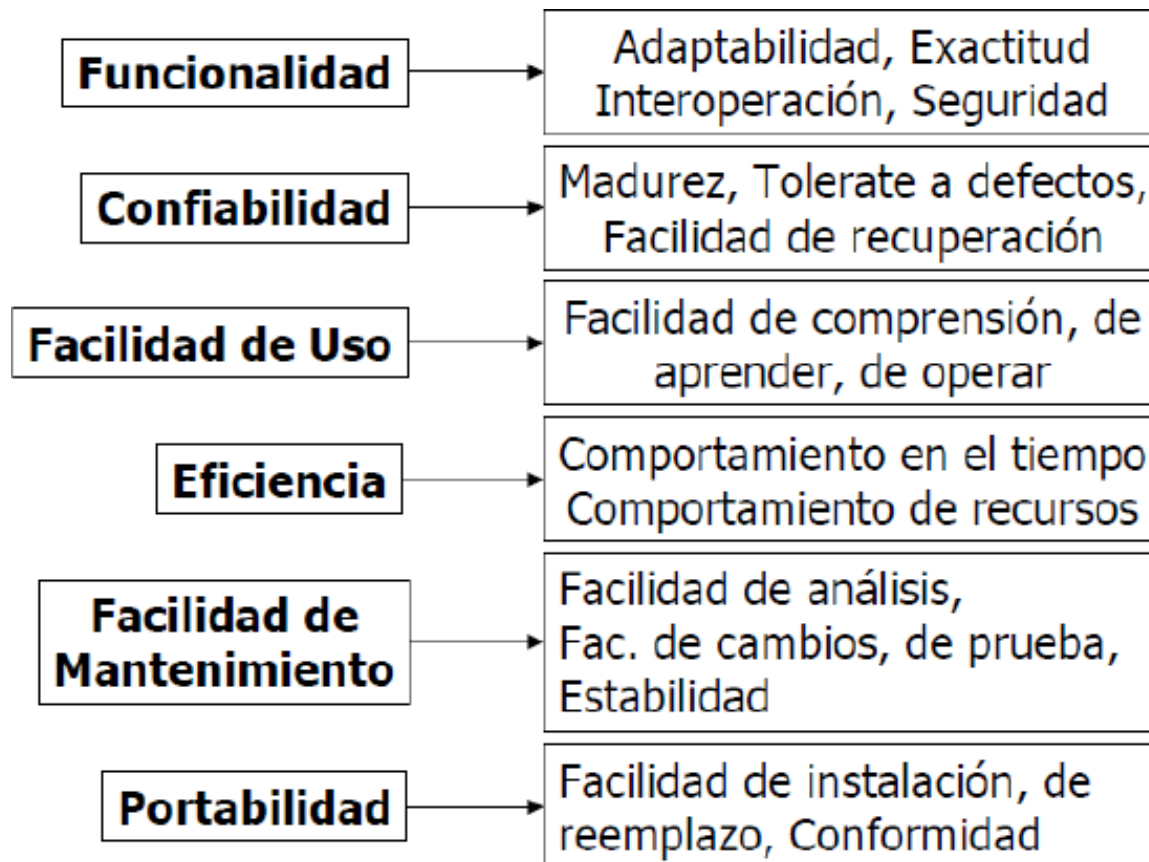
# Modelo FURPS

- Functionality, Usability, Reliability, Performance, Supportability)
- HP (1987)

ATRIBUTOS	SUBATRIBUTOS
<b>FUNCIONALIDAD</b>	Conjunto de características Capacidades Generalidad Seguridad
<b>FACILIDAD DE USO</b>	Factores humanos Aesthetics Consistencia Documentación
<b>FIABILIDAD</b>	Frecuencia/severidad de falla Recuperabilidad Predictabilidad Precisión Tiempo promedio de falla
<b>RENDIMIENTO</b>	Velocidad Eficiencia Consumo de recursos Throughput Tiempo de respuesta
<b>CAPACIDAD DE SOPORTE</b>	Capacidad de prueba Extensabilidad Adaptabilidad Mantenibilidad Compatibilidad Configurabilidad Capacidad de Servicio Capacidad de Instalación Capacidad de Localización



# Modelo ISO 9126



- La arquitectura de software determina muchos de los atributos de calidad de un sistema (Bass , 2003)
  - Si la arquitectura no ha sido diseñada para soportar ciertos atributos de calidad, será difícil agregar soporte únicamente con diseño detallado e implementación.
  - La realización de muchos atributos de calidad no depende de un módulo solo.
  - La descomposición de un sistema afecta a sus atributos de calidad.
- Dicho de otra manera: muchos atributos de calidad son "arquitecturalmente sensitivos"
  - La arquitectura de Software es crítica para su realización.
  - Los atributos de calidad deben ser diseñados y evaluados a nivel de arquitectura (Bass, 2003)

# No todos los QAs son arquitecturales

- La Arquitectura, por sí misma, no puede cumplir con los atributos de calidad (Bass, 2003)
  - La Arquitectura proporciona una base, no una garantía
  - El diseño detallado y la implementación también afectan
- Existen muchos aspectos de los atributos de calidad que no son arquitecturalmente sensitivos : algunos atributos son más sensitivos que otros.

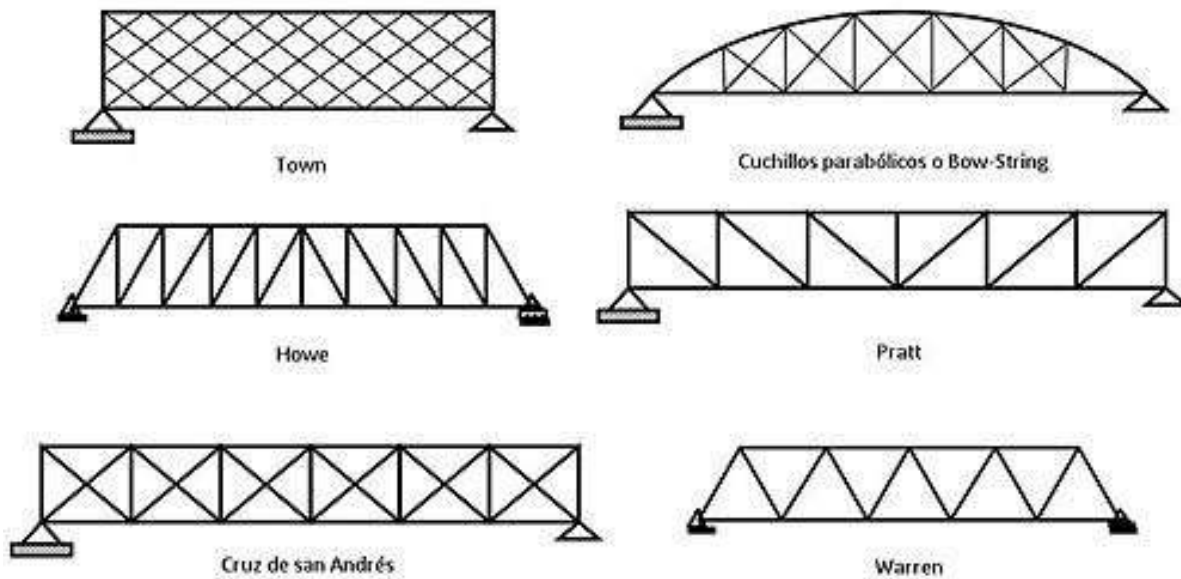
# Funcionalidad vs. QAs sin S.A.

- El mapeo entre la funcionalidad del sistema sobre las estructuras de software determina el soporte de la arquitectura para dichos atributos de calidad
- Ahora bien, la funcionalidad por si misma no es arquitecturalmente sensitiva.
  - La Funcionalidad es ortogonal a la estructura
  - Cualquier numero de posibles estructuras pueden concebirse para implementar la funcionalidad.
  - Si el cumplimiento del logro de la funcionalidad es el único requerimiento, el sistema podría existir como un componente simple y monolítico .

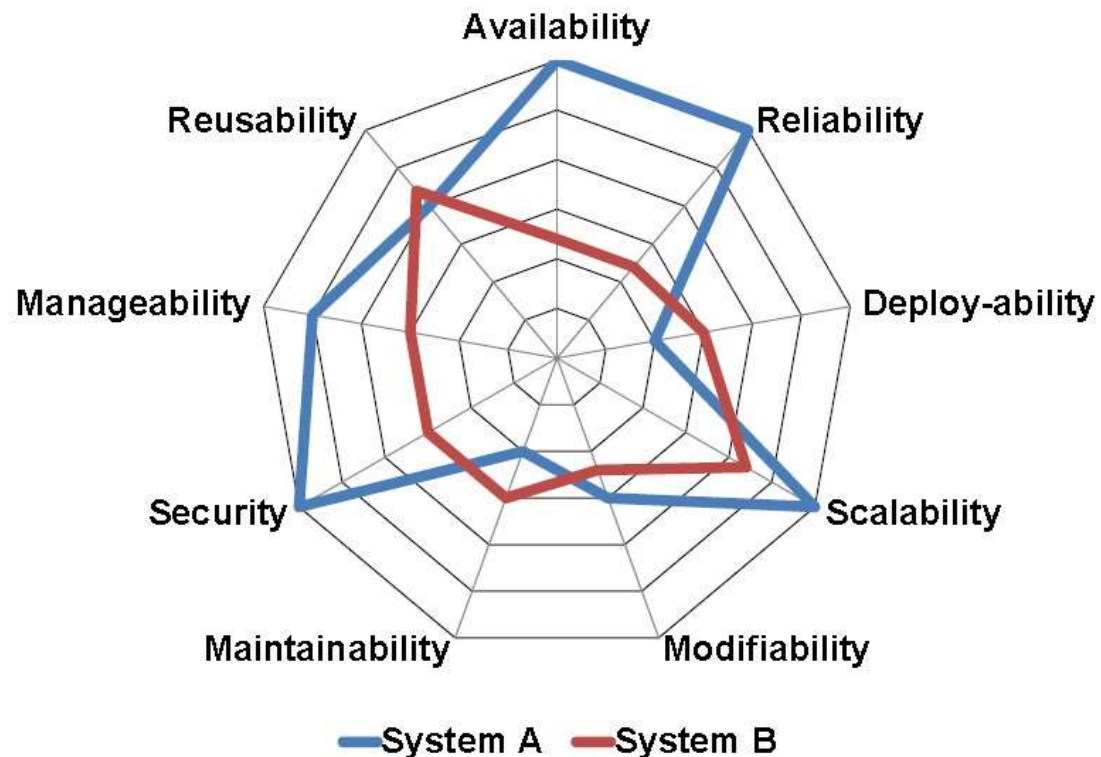
(Bass, 2003)

- ¿ la funcionalidad es lo único que importa ? ← **NO**
  - Dos sistemas pueden tener estructura diferente y proporcionar la misma función

Ejemplo: un puente ...



- ¿que más importa?
  - modificabilidad, rendimiento y muchos otros
- ¿cuáles son los efectos de las decisiones arquitectónicas en los atributos de calidad? : Cada estructura promueve diferentes calidades
  - diseñado para la modificabilidad
  - diseñado para la seguridad
  - diseñado para un rendimiento ...



- Si el sistema requiere performance:
  - Manejar el comportamiento de los elementos en el tiempo
  - Uso de recursos compartidos
  - Frecuencia y volumen de comunicación entre los elementos
  
- Si el sistema requiere disponibilidad
  - Manejar el comportamiento de los elementos cuando ocurre una falla
  - Como el sistema responde a la falla
  
- Si el sistema requiere usabilidad
  - Aislar los detalles de la GUI y de aquellos elementos responsable de la experiencia de usuario del resto del sistema.
  
- Si el sistema requiere interoperabilidad
  - Manejar los elementos responsables de las interacciones externas de manera que se puedan controlar.

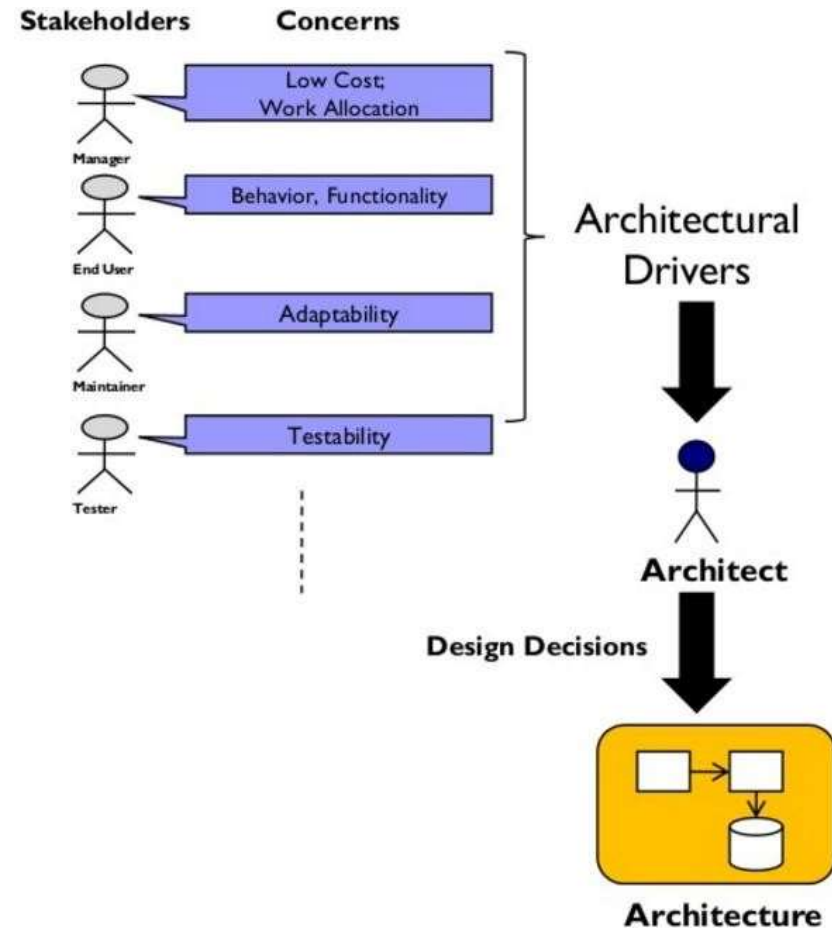
“Good design  
goes to heaven;  
bad design goes  
everywhere.”

Alfred Genshorn



# Drivers Arquitecturales

- **Requisitos** que **dan forma** a la arquitectura
- En principio TODOS los requerimientos son Drivers, pero dado el plazo, deben **priorizarse** (\*\*\*)
- La arquitectura es fundamental para la realización de evaluaciones de calidad : no se puede diseñar un sistema y luego volver atrás y agregar la calidad



# No debería ser ...

## ***“El sistema debe ser modificable”***

- No se puede diseñar un sistema que es modificable para todos los cambios posibles
- Cada sistema se puede modificar con respecto a un conjunto de cambios y no con respecto a alguna otra
- El problema REAL de la cuestión es: ¿cómo podemos reducir al mínimo el costo de estos cambios?
- Entonces agradeceremos que sean específicos :
  - ¿ qué tipo de cambios?
  - ¿ la forma de medir el costo?

# ¿ Por qué los Requerimientos no son suficientes?

- Los requerimientos de calidad generalmente son pobremente definidos, están a muy alto nivel o simplemente no existen.
- En consecuencia, los requerimientos no comunican lo suficiente
  - Generan malentendidos o diferentes interpretaciones
- *Los Escenarios* proporcionan una manera simple y poderosa para cubrir la brecha entre los requerimientos y la arquitectura
  - Concretan los requerimientos, comunican a todos los stakeholders
  - Resaltan los aspectos más críticos del sistema



# Escenarios

- Un escenario es una oración corta que describe la interacción de uno de los stakeholders con el sistema. (Clements, 2002)
- Un escenario arquitectónico es una descripción nítida y concisa de una situación que es probable que se enfrente en el entorno de producción del sistema, junto con la definición de la respuesta requerida del sistema. (Rozanski y Woods, 2005)



# Un escenario NO ES un Caso de uso

- Aunque algunas veces se tratan como sinónimos, un escenario NO ES un Caso de Uso.
- Un escenario es UNA instancia de un caso de uso (Kruchten, 1995; Jacobson, 1995) ← pero no siempre (o no todos)
  - Un caso de uso describe un conjunto de secuencias de acciones
  - Un escenario es la instanciación concreta de un caso de uso.
- Los casos de uso son más formales (Jacobson, 1995)
- Los casos de Uso describen el sistema estrictamente desde un punto de vista de caja negra, mientras que los escenarios son vinculados a los elementos internos del sistema (Jacobson, 1995)
- Los escenarios son iniciados por el sistema o por el transcurrir del tiempo.

# Utilidad de lo escenarios

- Insumo para el proceso de diseño de la arquitectura (Kruchten, 1995)
- Evaluación de la arquitectura
- Comunicación con los stakeholders
- Identificación de requerimientos faltantes
- Guía para pruebas

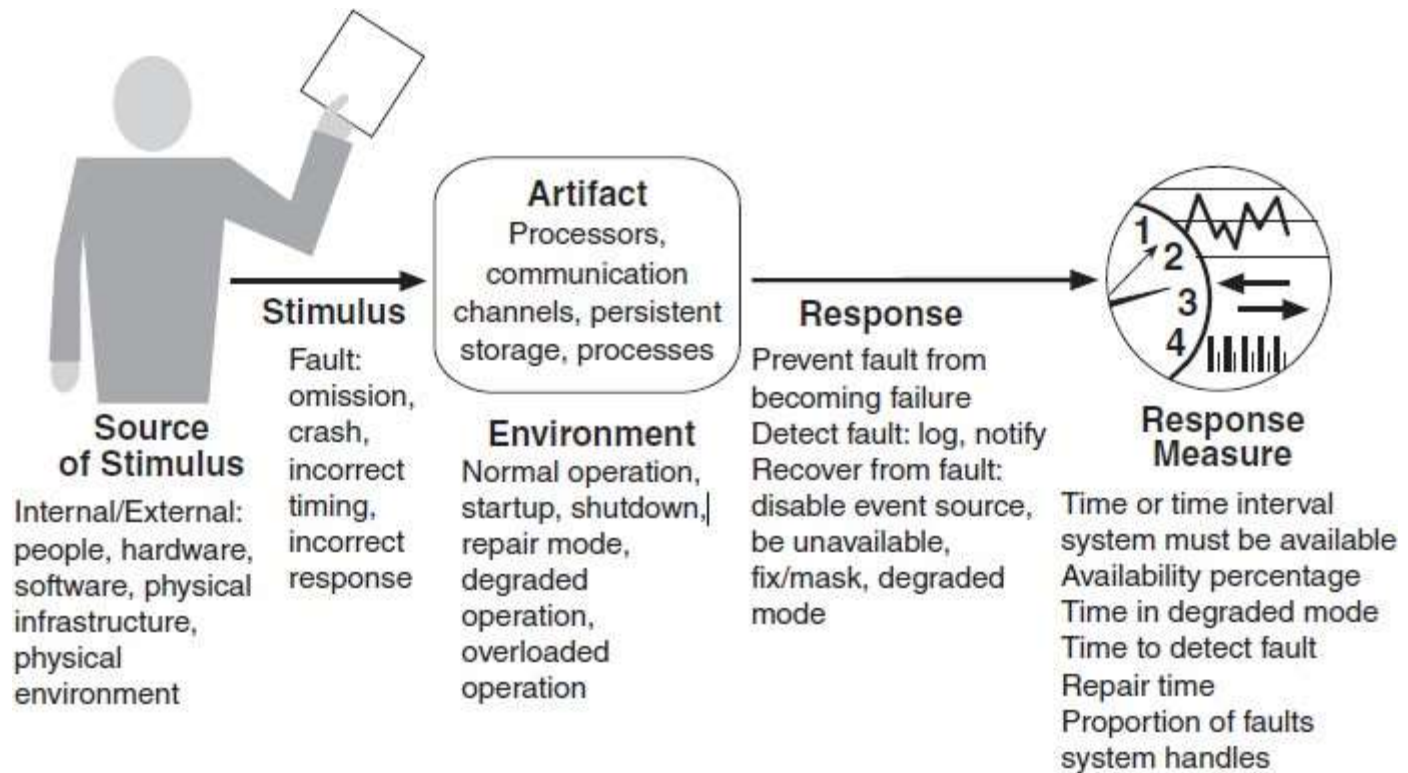
(Rozanski and Woods, 2005)

# Tipos de escenarios

- Rozanski y Woods (2005):
  - Escenarios funcionales
  - Escenarios de Calidad del Sistema
- Clements (2002):
  - Escenarios de Casos de Uso
    - Interacción del usuario con el sistema
    - Pueden usarse para capturar ciertos atributos de calidad
  - Escenarios de crecimiento
    - Para anticipar cambios en el sistema
  - Escenarios exploratorios
    - Para evaluar what-if



# Elementos del Escenarios



Las partes innecesarias pueden omitirse

# Ejemplo: Escenario de Performance

- Fuente
  - Fuentes internas o externas
- Estimulo
  - Arribo de eventos periódicos;
  - Arribo de eventos esporádicos,
  - Arribo de eventos estocásticos
- Artefacto
  - Sistema
- Ambiente
  - Modo Normal; Modo overload
- Respuesta
  - Procesar estímulo
  - Cambiar el nivel de servicio
- Medida de la respuesta
  - Latencia;
  - Throughput
  - Jitter
  - miss rate
  - Pérdida de datos

(Bass *et al.*, 2003)

# Ejemplo: Escenario de Disponibilidad

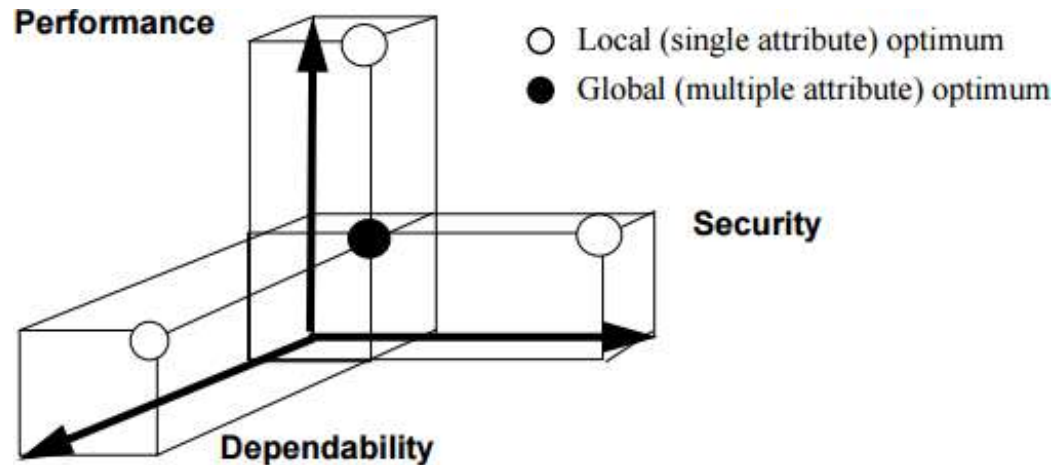
- Fuente
  - Interna o externa
- Estímulo
  - Falla: omisión, caída, timeout, respuesta incorrecta
- Artefacto
  - Recurso que debería estar disponible: procesador, canal de comunicación channel, almacenamiento, proceso
- Ambiente
  - Modo Normal; modo degradado
- Respuesta
  - Detectar el evento y ejecutar una acción:
    - Registrar el evento
    - Notificar a las personas adecuadas
    - Desactivar la fuente de eventos
    - Colocarse en modo no disponible por un lapso de tiempo
    - Continuar operando en modo normal o en modo degradado
- Medida de la Respuesta
  - Intervalo de tiempo para que el sistema esté disponible
  - Tiempo de disponibilidad;
  - Intervalo de tiempo en el cual el sistema opera en modo degradado
  - Tiempo de reparación

# Ejemplo: Escenario de Modificabilidad

- Fuente
  - Usuario final, desarrollador, administrador del sistema
- Estímulo
  - Deseo de agregar, borrar, modificar funcionalidad, atributo de calidad o capacidad
- Artefacto
  - Interface de usuario
  - Plataforma
  - Entorno
  - etc
- Ambiente
  - Runtime
  - compile time, build time, design time
- Respuesta
  - Unicación de los lugares a ser modificados en la arquitectura
  - Hacer la modificación sin afectar otra funcionalidad
  - Probar la modificación
  - Desplegar la modificación
- Medida de la Respuesta
  - Costo en términos del número de elementos afectados / esfuerzo / dinero;
  - Lo anterior se extiende a otras funciones y atributos de calidad afectados

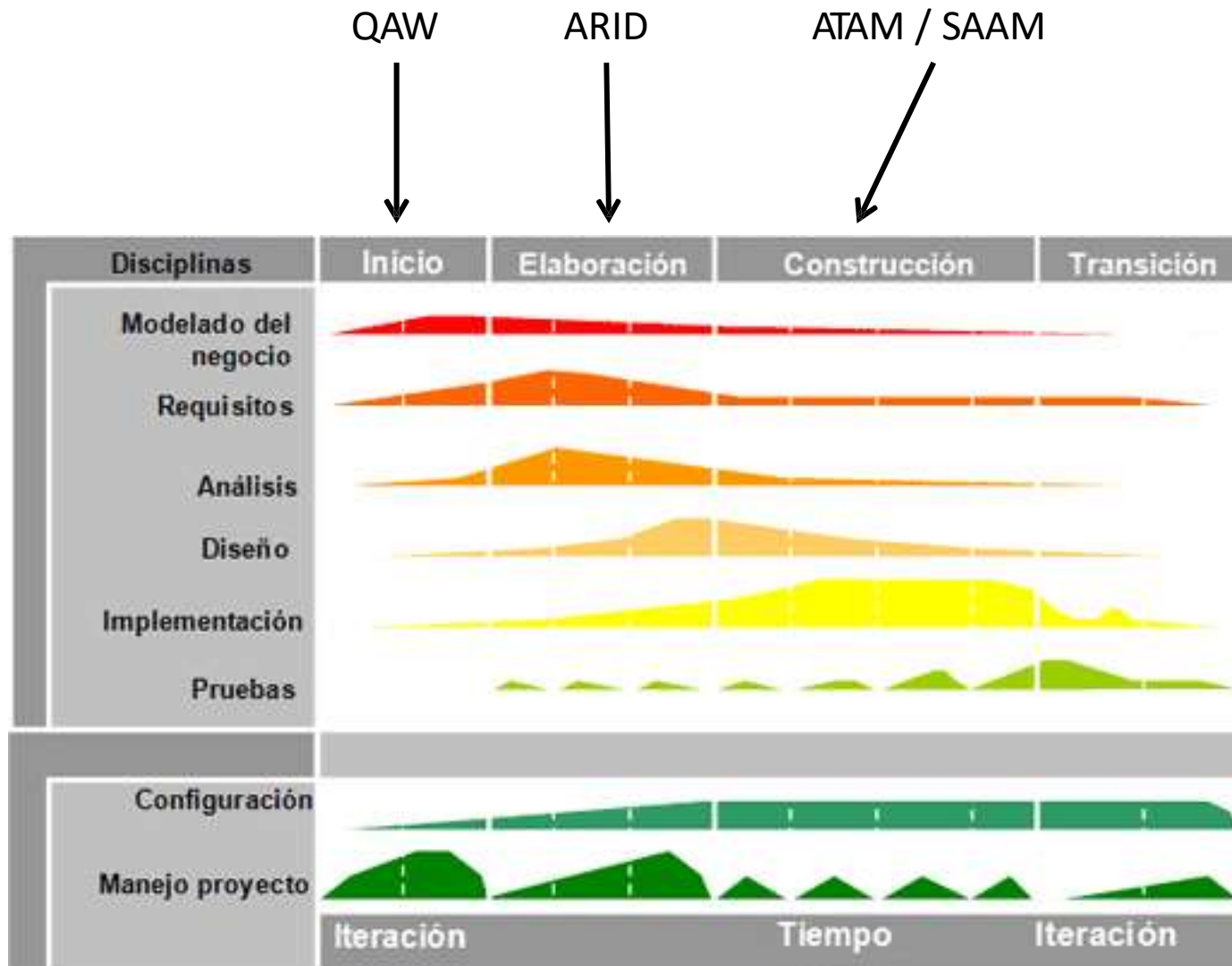
# Calidad del Software

La calidad del software es el grado en el que el software posee una combinación deseada de atributos



# Validar y Mejorar la arquitectura

- ✓ Quality Attributes Workshop (QAW)
- ✓ Architecture Tradeoff Analysis Method (ATAM)
- ✓ Software Architecture Analysis Method (SAAM)
- ✓ Active Reviews for Intermediate Designs (ARID)



Paso	QAW	ATAM	SAAM	ARID
1	QAW presentation and introductions	Present the ATAM	Develop scenarios	Identify the reviewers
2	Business and mission presentation	Present the business drivers	Describe architecture(s)	Prepare the design briefing
3	Architectural plan presentation	Present the architecture	Classify and prioritize the scenarios	Prepare the seed scenarios
4	Identification of architectural drivers	Identify the architectural approaches	Individually evaluate indirect scenarios.	Prepare the material
5	Scenario brainstorming	Generate the quality attribute utility tree	Assess scenarios interactions (Similar QAW Step 6)	Present ARID
6	Scenario consolidation	Analyze the architectural approaches		Present the design
7	Scenario prioritization	Brainstorm and prioritize scenarios		Brainstorm and prioritize scenarios
8	Scenario refinement	Analyze architectural approaches		Apply the scenarios
9		Present the results		Summarize





Se puede ver una arquitectura como el resultado de aplicar un conjunto de decisiones de diseño.

A continuación, una categorización sistemática de estas decisiones para que un arquitecto pueda centrar la atención en las dimensiones de diseño que puedan ser más problemáticas.

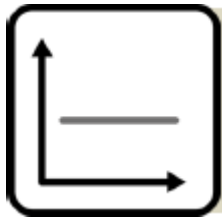
# Workload

<http://searchdatacenter.techtarget.com/definition/workload>

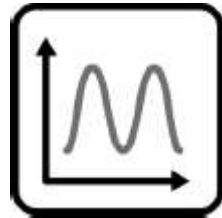
La carga de trabajo es la cantidad de procesamiento que el computador puede hacer en un momento dado.

La carga de trabajo se compone de una cierta cantidad de lógica de aplicaciones en ejecución y por lo general algún número de usuarios conectados para interactuar con las aplicaciones.

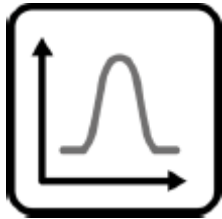
Se debe identificar el workload



Estático



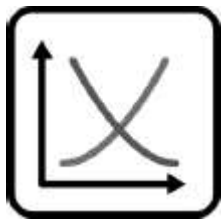
Periódico



Pico de Una Vez



Aleatoria e Impredecible



Crecimiento/Reducción constante

USER STORY	HOW OFTEN	HOW MUCH
1. User logs in	once	Trivial
2. User sets initial configuration	rarely	3-4 forms, a few 100 bytes
3. User checks status	about $\frac{1}{2}$ the time (1 second updates)	300 bytes
4. User gets statistics	once per day	5K bytes
5. User displays performance charts	about $\frac{1}{2}$ the time (1 second updates)	300 bytes
6. User logs out	once	Trivial

# Allocation of Responsibilities

- Identificar las responsabilidades más importantes:
  - ✓ Funcionamiento básico del sistema
  - ✓ Infraestructura
  - ✓ Cumplimiento de Atributos de Calidad
- Determinar cómo dichas responsabilidades se asignan:
  - ✓ Elementos que no son Runtime
  - ✓ Elementos de Runtime

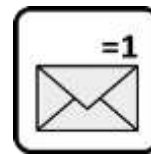
Se puede usar :




- Descomposición funcional
- Modelamiento de objetos del mundo real
- Agrupamiento basado en los modos de operación
- Agrupamiento basado en requerimientos de calidad similares



# Coordination Model

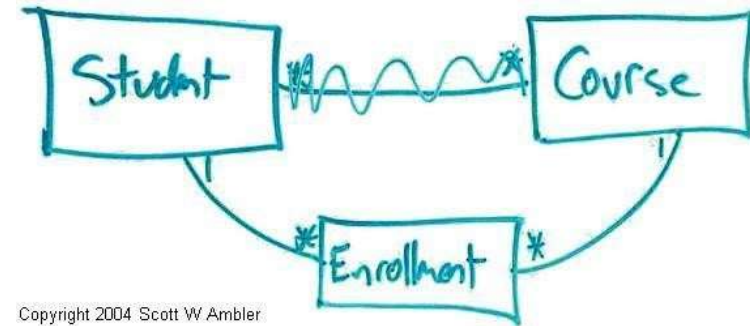
- Identificar elementos del sistema que DEBEN/NO DEBEN coordinar
- Determinar las propiedades de la coordinación:
  - ✓ Timeliness → puntualidad/oportunidad
  - ✓ Accuracy → precisión
  - ✓ Completeness → completitud
  - ✓ Correcteness → exactitud
  - ✓ Consistency → consistencia
- Seleccionar los mecanismos de coordinación, por ejemplo:
  - ✓ Sin Estado / Con Estado
  - ✓ Síncrono / asíncrono
  - ✓ Entrega garantizada / Entrega no garantizada



Consistency		La consistencia requiere que no haya dos o más requisitos de una especificación se contradicen entre sí
Completeness		La noción de integridad está esencialmente relacionada y vinculada a la finalidad y el objetivo de un sistema . Es decir, la integridad de una especificación es relativo a la satisfacción de un determinado conjunto de objetivos
Correcteness		La corrección de una especificación de requisitos describe la correspondencia de esa especificación con las necesidades reales de los usuarios previstos de la misma manera que la corrección de una pieza de software se refiere al acuerdo de la parte de software con su especificación .

# Data Model

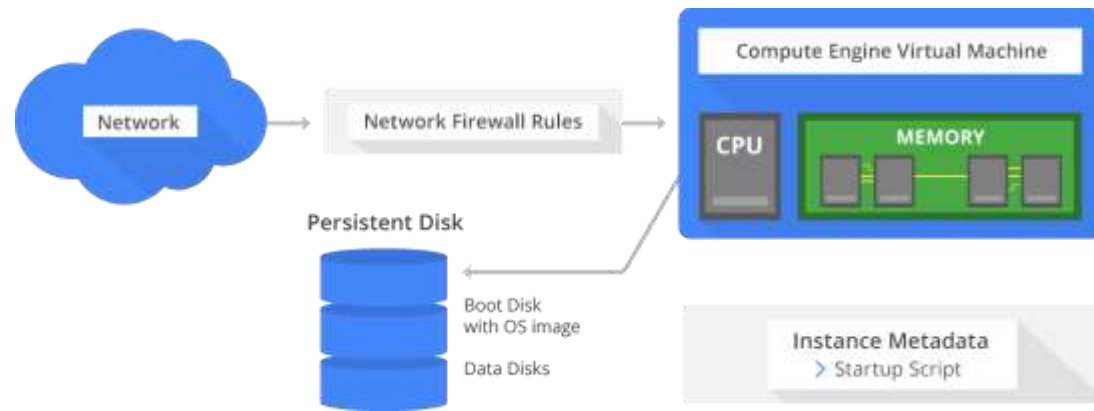
- Seleccionar las mayores abstracciones de los datos, sus operaciones y propiedades:
  - ✓ Como se crean
  - ✓ Como se inicializan
  - ✓ Como se acceden
  - ✓ Como se manipulan
  - ✓ Como se transfieren
  - ✓ Como se destruyen
- Determinar la METADATA que permita interpretar los datos
- Seleccionar la tecnología con la que se organizarán los datos





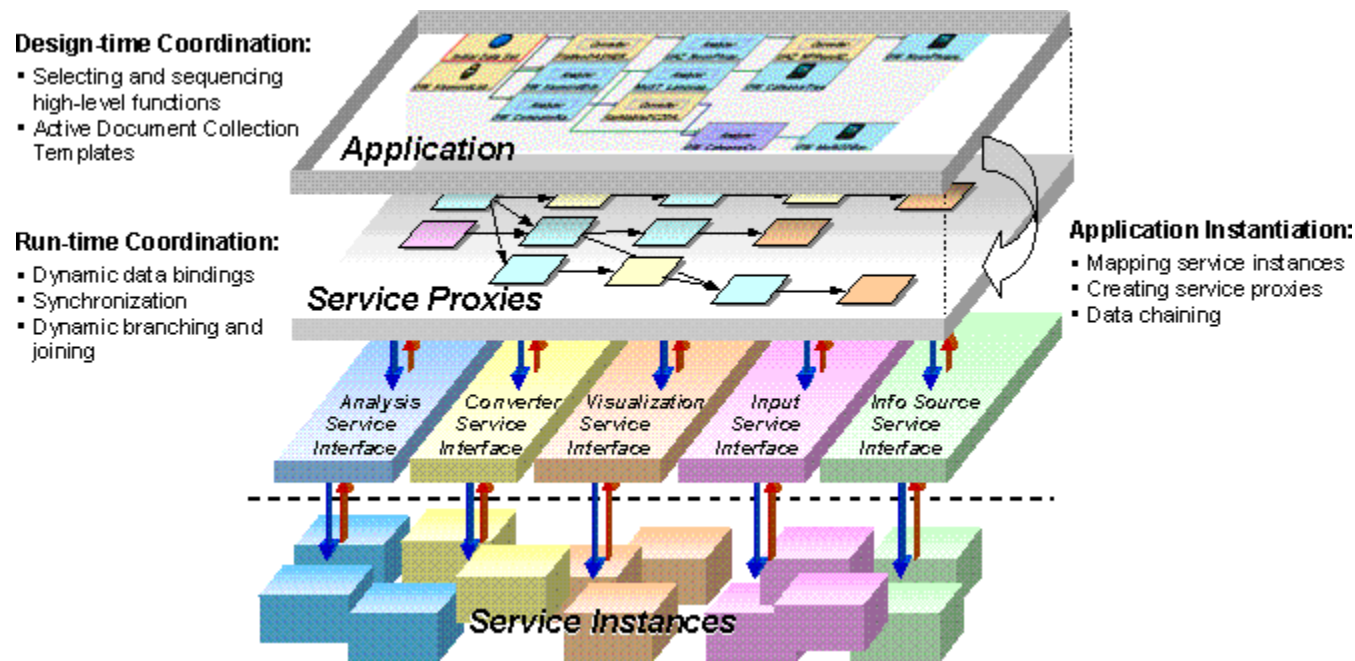
# Management of Resources

- Balancear el uso de **recursos compartidos**:
  - ✓ Hardware : CPU, memoria, red, etc
  - ✓ Software : Threads , pools , buffers
- Determinar los recursos a ser manejados : ver sus límites
- Determinar las estrategias cuando haya contención de recursos
- Determinar los elementos a ser afectados cuando haya saturación de recursos.



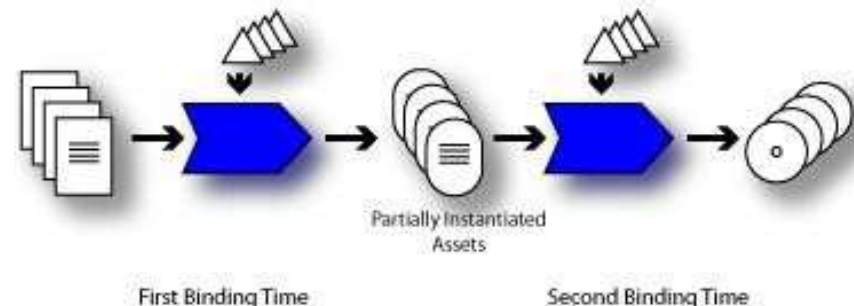
# Mapping among architectural elements

- Mapear módulos a unidades de ejecución
- Mapear elementos de software con elementos del entorno
- Por ejemplo:
  - ✓ Asignación de elementos de run-time a los procesadores.



# Binding time decisions

- Introducen un rango de variación (variabilidad)
  - Las decisiones de las láminas anteriores deberían considerar la variabilidad
  - Considerar los costos de implementar la decisión y los costos asociados por modificarla una vez tomada
- 
- ✓ **Source reuse time.** Decisions bound when reusing a configurable source artifact
  - ✓ **Development time.** Decisions bound during architecture, design, and coding
  - ✓ **Static code instantiation time.** Decisions bound during assembly of code just prior to a build
  - ✓ **Build time.** Decisions bound during compilation or related processing
  - ✓ **Package time.** Decisions bound while assembling binary & executable collections
  - ✓ **Customer customizations.** Decisions bound during custom coding at customer site
  - ✓ **Install time.** Decisions bound during the installation of the software product
  - ✓ **Startup time.** Decisions bound during system startup
  - ✓ **Runtime.** Decisions bound when the system is executing



# Choice of technology

- Si la tecnología es seleccionada por otros, se convierte en una restricción para la arquitectura.
- En caso contrario, el arquitecto debe seleccionar un conjunto de tecnologías posibles que permitan realizar decisiones en cada una de las categorías anteriores:
  - ✓ Analizar las Tecnologías disponibles que sean adecuadas
  - ✓ Herramientas de desarrollo disponibles
  - ✓ Familiaridad (interna) y soporte (externo)
  - ✓ Efectos de seleccionar una tecnología
  - ✓ Compatibilidad con el stack de software existente.



# HIGH LEVEL ARCHITECTURE

SENSORS

IQRF

GATEWAYS

INTERNET

DATA CENTER

