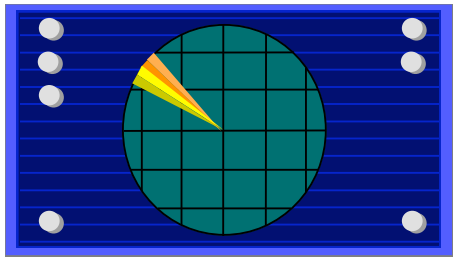
Modelado y Diseño de Arquitectura de Software

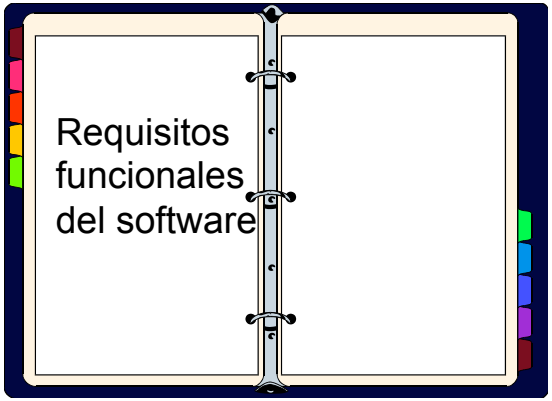
CONCEPTOS DE MODELADO

Fernando Barraza A. MS.c.

fernando.barraza@gmail.com

Desarrollo de sistemas de software





Si todo lo que importara fuese la funcionalidad, cualquier software monolítico serviría, ... ***pero otras cosas también importan***

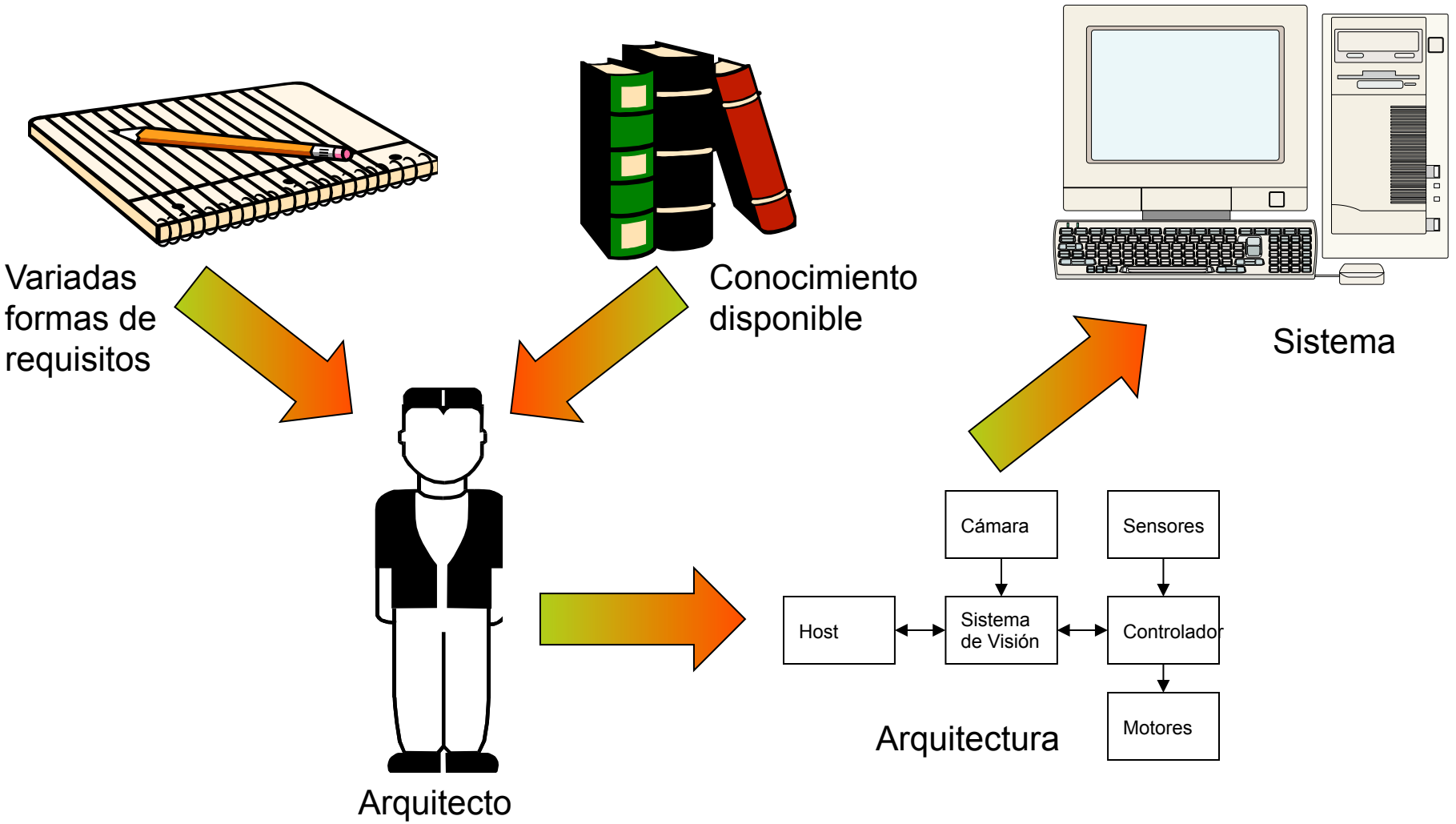
Los atributos de calidad del software y su caracterización son esenciales.

- Modificabilidad
- Interoperabilidad
- Disponibilidad
- Seguridad
- Predictabilidad
- Portabilidad
- ...

Manejadores de atributos de calidad → Arquitectura del software → Software

tiene estas cualidades

Los requisitos determinan el modelo



Implicaciones de no seguir un proceso conocido de modelado

- La arquitectura es una abstracción de un sistema.
- Los sistema pueden tener y tienen una estructura.
- Todo sistema tiene una arquitectura.
- Tener una arquitectura no es lo mismo que tener una arquitectura conocida por todos.

Si no se desarrolla la arquitectura explícitamente, se obtendrá una de todas formas, pero puede no gustarnos lo que obtenemos!

Arquitectura y Funcionalidad

- La funcionalidad es en gran medida ortogonal a los requisitos de calidad:
  - La funcionalidad es la capacidad del sistema de hacer lo que se pretendía que hiciese;
  - Los sistemas se descomponen en elementos para lograr variados propósitos, más allá de la funcionalidad:
    - Las opciones de arquitectura promueven ciertas cualidades al tiempo que implementan la funcionalidad deseada.

Consecuencias de las decisiones de AS sobre las Cualidades

- 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 una cualidad suele afectar las otras cualidades;
  - la arquitectura sólo puede permitir, no garantizar, que cualquier requisito de calidad se alcance.

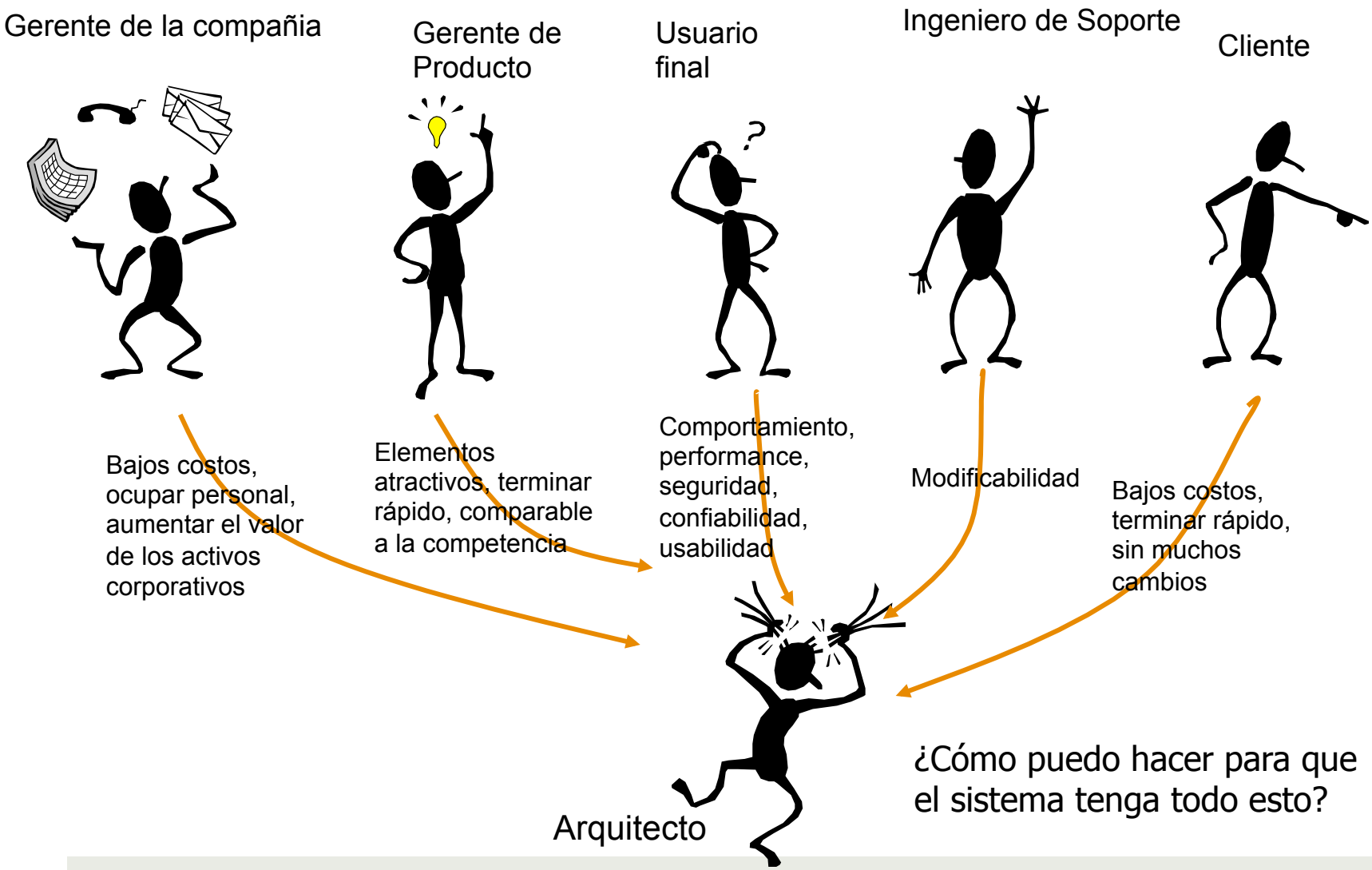
Desafíos

- ¿Qué significan con precisión atributos de calidad tales como modificabilidad, seguridad, performance y confiabilidad?
- ¿Cómo se estructura el sistema de modo que tenga 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?

Realidad sobre Arquitectura de Software

- Los requisitos de atributos de calidad son las principales guías para el diseño de la arquitectura.
- La medida en que un sistema alcance sus requisitos de atributos de calidad depende de las decisiones de arquitectura.
- El desarrollo requiere ser guiado por las decisiones de arquitectura.

Influencia de los Interesados



Interesados Involucrados

- 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.

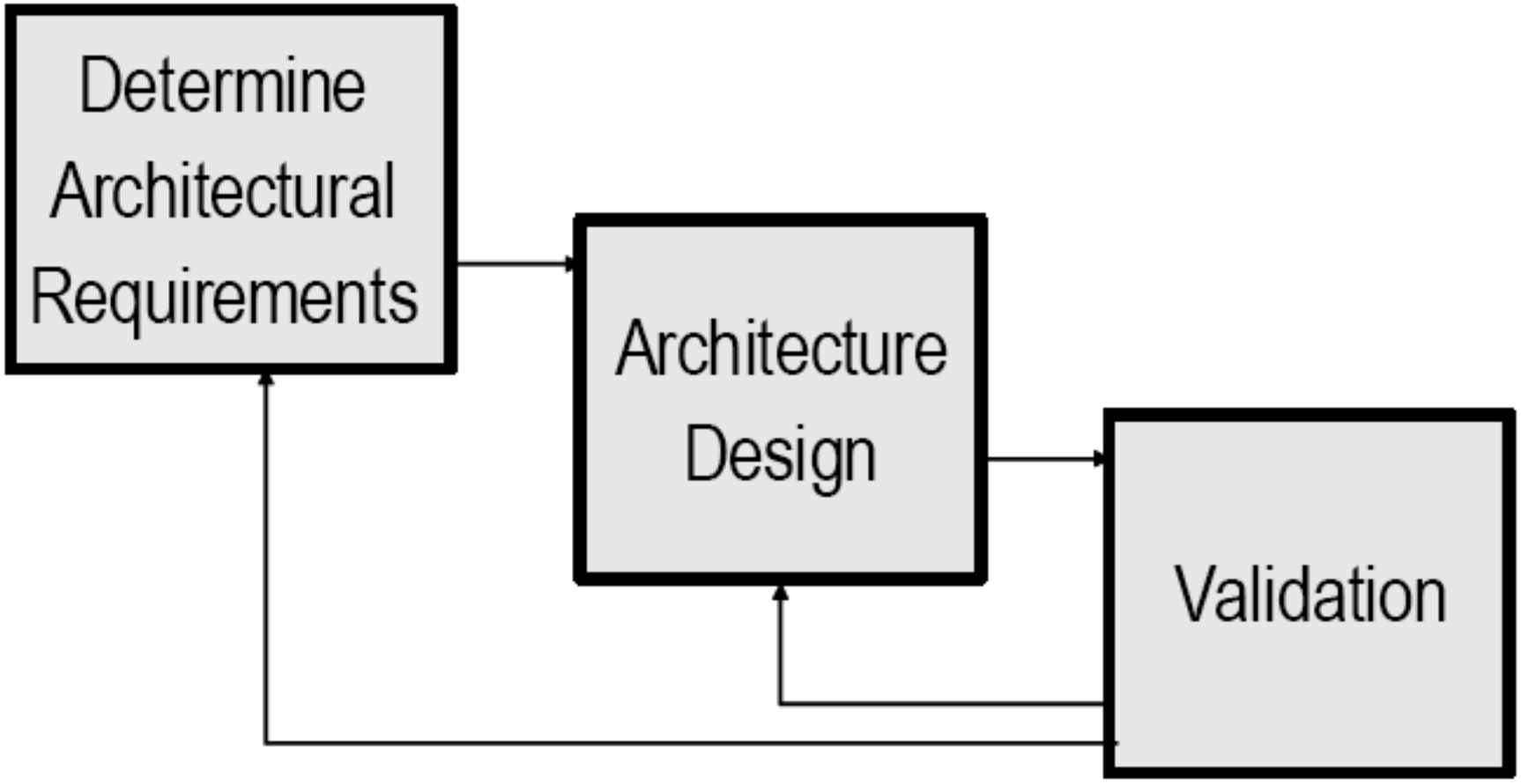
Desarrollo de Software Tradicional

- Los atributos de calidad rara vez se capturan como parte de la especificación de requisitos;
- generalmente son sólo vagamente comprendidos;
- frecuentemente pobremente articulados

Creación de una Arquitectura de Software

- Existen métodos y guías para la definición de la arquitectura, muchos de los cuales se focalizan en los requisitos funcionales.
- Es posible crear una arquitectura basada en las necesidades de atributos de calidad.

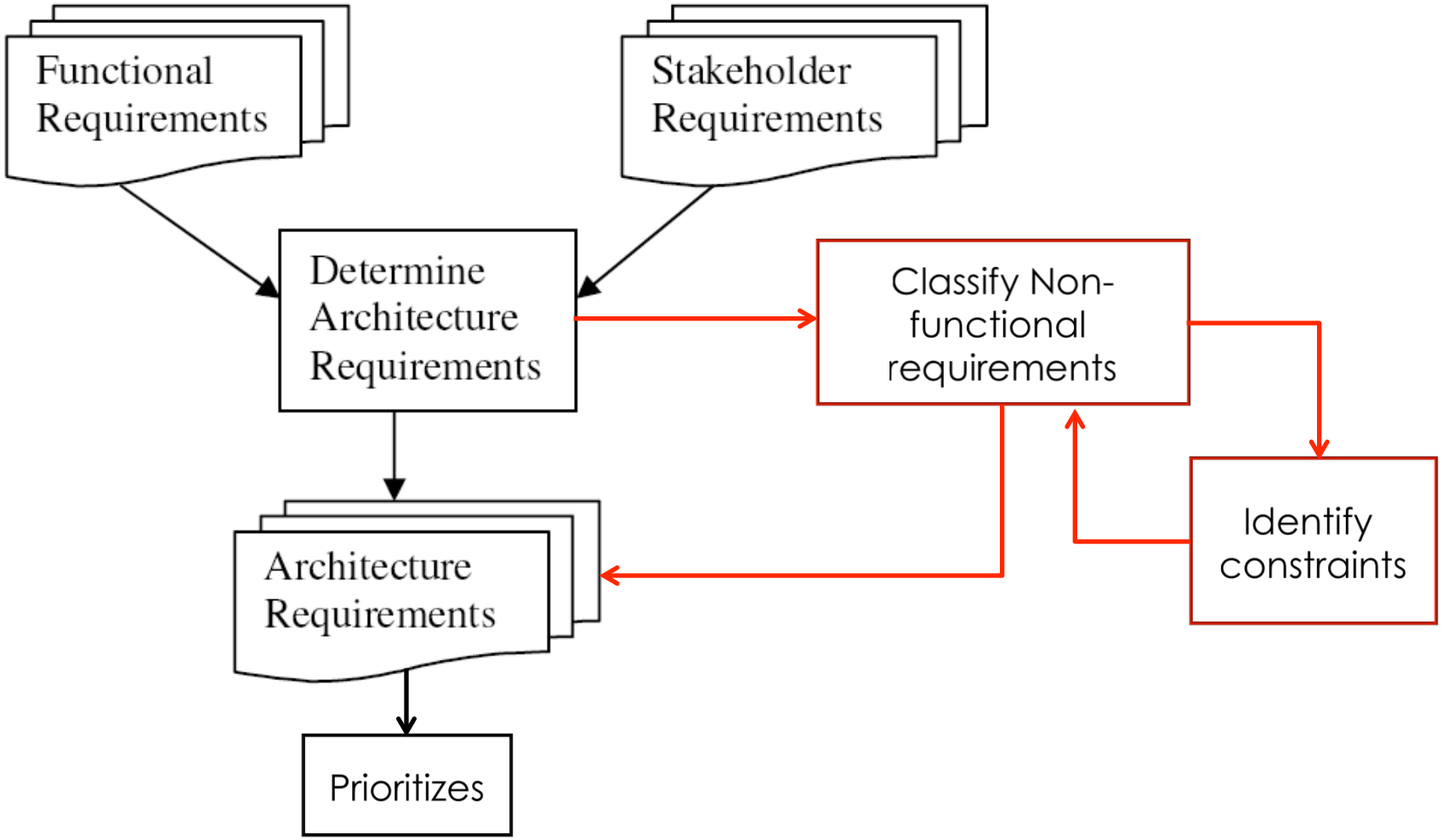
Esquema de proceso de modelado de AS



Etapas del proceso

- **Definir los requerimientos:** Involucra crear un modelo desde los requerimientos que guiarán el diseño de la arquitectura basado en los atributos de calidad esperados
- **Diseño de la Arquitectura :** Involucra definir la estructura y las responsabilidades de los componentes que comprenderán la Arquitectura de Software
- **Validación:** Significa "probar" la arquitectura, típicamente pasando a través del diseño contra los requerimientos actuales y cualquier posible requerimiento a futuro.

Identificar requerimientos



Requerimientos no funcionales

- Describen como el software debe comportarse, es decir como hacer algo, no que debe hacer
- Están relacionados con los requerimientos funcionales porque describen la forma que se espera se logren dichos requerimientos
- En algunos casos tienen restricciones de cómo hacerlo
- Se clasifican de acuerdo al atributo de calidad esperado del sistema

Ejemplo de requerimientos de AS

Quality Attribute	Architecture Requirement
Performance	Application performance must provide sub-four second response times for 90% of requests.
Security	All communications must be authenticated and encrypted using certificates.
Resource Management	The server component must run on a low end office-based server with 512MB memory.
Usability	The user interface component must run in an Internet browser to support remote users.
Availability	The system must run 24x7x365, with overall availability of 0.99.
Reliability	No message loss is allowed, and all message delivery outcomes must be known with 30 seconds
Scalability	The application must be able to handle a peak load of 500 concurrent users during the enrollment period.
Modifiability	The architecture must support a phased migration from the current Forth Generation Language (4GL) version to a .NET systems technology solution.

Restricciones (constraints)

- Las restricciones (*constraints*) imponen condiciones sobre la arquitectura que normalmente no son negociables.
- Limitan el rango de alternativas de decisión del arquitecto
  - Algunas veces hace la vida más fácil para el arquitecto, en otras lo complica.
- Se pueden clasificar según su naturaleza:
  - Negocio, Desarrollo, Tiempo, Costo, etc.

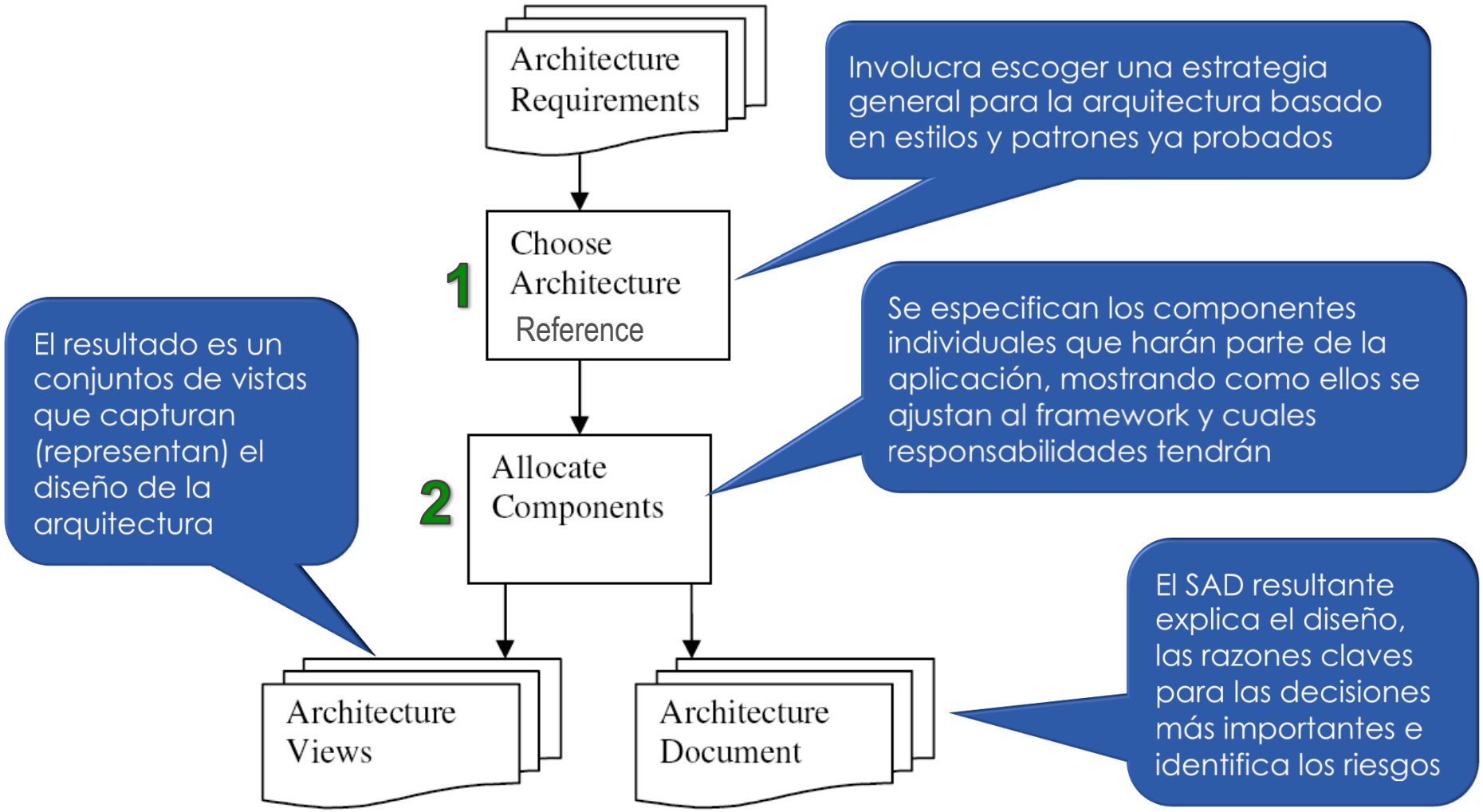
Ejemplos de restricciones

Constraint	Architecture Requirement
Business	The technology must run as a plug-in for MS BizTalk, as we want to sell this to Microsoft.
Development	The system must be written in Java so that we can use existing development staff.
Schedule	The first version of this product must be delivered within six months.
Business	We want to work closely with and get more development funding from *MegaHugeTech Corp*, so we need to use their technology in our application.

Priorización de requerimientos

- **Alta:** La aplicación debe soportar el requerimiento. Estos requerimientos guían el diseño de la arquitectura
- **Media:** Requerimientos que necesitan ser soportados en algún momento o etapa del proyecto pero no necesariamente en esta siguiente versión.
- **Baja:** Se conoce como parte de la "wish-list". Se pueden implementar cuando sea posible hacerlo.

Diseño de la Arquitectura de Software

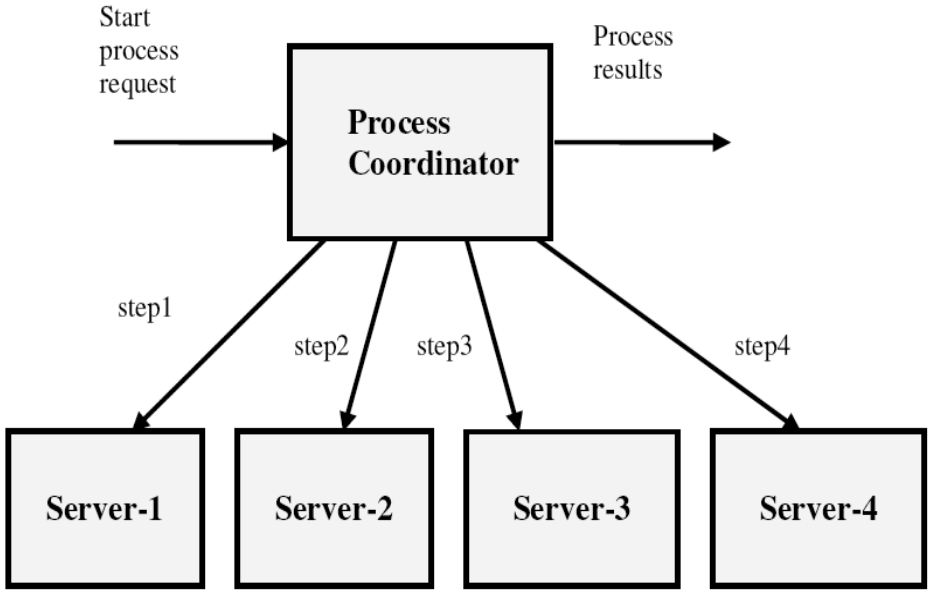


1. Escogencia de la Arquitectura de Referencia

- Discutir los posibles estilos y patrones más apropiados que den el soporte requerido para alcanzar los atributos de calidad deseados
  - ES LA TAREA MÁS CRÍTICA EN TODO EL PROCESO DE AS !!
- Basarse en Arquitecturas de Referencia reconocidas por tanto por la academia como por la industria
  - Implementaciones conocidas, de amplia difusión y uso
  - Buena documentación
- Reconocer el tamaño de la aplicación objetivo
  - Aplicaciones pequeñas → Pocos patrones requeridos
  - Aplicaciones grandes → Mezcla de varios patrones

Ejemplo de mapeo de patrones a atributos de calidad

Patrón *Process Coordinator*



Elementos esenciales

- **Encapsulación del proceso:** El coordinador contiene la lógica requerida para alcanzar el objetivo del proceso de negocio, siendo un solo punto de definición hace más fácil entender y modificar.
- **Bajo acoplamiento:** Los componentes de servidores no se conscientes de su rol en el proceso ni su orden de ejecución
- **Comunicación flexible:** La comunicación entre el coordinador y los servidores puede ser sincrónica o asincrónica

Atributos de calidad para el patrón Coordinador (1)

Quality Attribute	Issues
Availability	The coordinator is a single point of failure. Hence it needs to be replicated to create a high availability solution.
Failure handling	Failure handling is complex, as it can occur at any stage in the business process coordination. Failure of a later step in the process may require earlier steps to be undone using compensating transactions. Handling failures needs careful design to ensure the data maintained by the servers remains consistent.
Modifiability	Process modifiability is enhanced because the process definition is encapsulated in the coordinator process. Servers can change their implementation without affecting the coordinator or other servers, as long as their external service definition doesn't change.

Atributos de calidad para el patrón Coordinador (2)

Performance	To achieve high performance, the coordinator must be able to handle multiple concurrent requests and manage the state of each as they progress through the process. Also, the performance of any process will be limited by the slowest step, namely the slowest server in the process.
Scalability	The coordinator can be replicated to scale the application both up and out.

2. Asignación de componentes

- Su objetivo es definir los componentes principales que comprenderán el diseño
  - La arquitectura de referencia define los patrones de comunicación en general para los componentes
- Se busca además:
  - Identificar como los componentes se ajustan a los patrones
  - Identificar las interfaces y los servicios que cada componente soporta para así
    - Validar la asignación de responsabilidades de los componentes
    - Identificar dependencias entre ellos
  - Identificar las partes de la arquitectura candidatas a distribuirse en varios servidores

Guías para diseño de componentes

- Por ser los componentes el más alto nivel de abstracción en el diseño de la AS, existe similitudes en su diseño con las técnicas de diseño orientado a objetos:

  - Minimizar dependencias entre componentes evitando propagar los cambios entre muchos componentes y por ende sus pruebas.

  - Diseñar componentes que encapsulen un alta cohesión del conjunto de responsabilidades. La cohesión es una medida de que tan bien las partes de un componente encajan entre si.

Guías para diseño de componentes (2)

- Aísle las dependencias con tecnologías Middleware y cualquier COTS. Esto facilita los cambios en el diseño de la AS. Por supuesto se incurren en más esfuerzo para construir y algo de penalidad en desempeño.

- Utilice la descomposición para estructurar componentes jerárquicamente. El componente más externo define la interfaz pública disponible. Internamente los llamados a esa interfaz son delegados a otros componentes localmente definidos cuyas interfaces no son visibles externamente.

Guías para diseño de componentes (3)

- Reduzca al mínimo las llamadas entre componentes, ya que pueden resultar costosas si los componentes se distribuyen.

  Trate de agregar secuencias de llamadas entre componentes en una sola llamada que pueda realizar el procesamiento necesario en una sola solicitud. Esto crea métodos de grano grueso o servicios en las interfaces que hacen más trabajo por cada requerimiento-invocación.

Validación

- Durante el proceso de creación de la arquitectura, el objetivo de la fase de validación consiste en aumentar la confianza del equipo de diseño con respecto a que la arquitectura es adecuada para cumplir con los requerimientos del sistema.
- Aunque se puede estar actuando sobre un sistema existente o nuevo al final el resultado del modelado es un diseño de AS por lo que el proceso de validación puede ser el mismo para ambos casos.
- Se puede escoger entre dos técnicas: Pruebas manuales o Prototipos.

Técnicas de validación

- El objetivo de ambas técnicas es el de identificar posibles deficiencias y debilidades en el diseño para que puedan ser mejoradas antes de la implementación.

1. Prueba manual: Involucra la prueba de la AS usando escenarios

2. Prototipo: Involucra la construcción de un prototipo que crea un arquetipo de la aplicación deseada, de esta forma su capacidad para satisfacer las necesidades se pueden evaluar con más detalle a través de prototipos.

Prueba manual por escenarios

- Los escenarios son una técnica desarrollada en el SEI para desentrañar las cuestiones relativas a una arquitectura a través de pruebas y evaluación manual.
- Los escenarios están relacionados con las preocupaciones arquitectónicas tales como los atributos de calidad, y tienen como objetivo poner de relieve las consecuencias de las decisiones arquitectónicas que se encapsulan en el diseño.
- Los escenarios pueden ser concebidos para hacer frente a cualquier requisito de calidad de interés en una aplicación dada.

El método SEI ATAM

- Este método describe los escenarios y su generación en gran detalle.

Ejemplo:

"Un escenario de "disponibilidad" muestra que los mensajes se pueden perder si un servidor falla antes de que los mensajes han sido entregados. La implicación aquí es que los mensajes no se han persistido en el disco, por razones de desempeño probablemente. La pérdida de los mensajes en algunos contextos de aplicación puede ser aceptable. Si no es así, este escenario pone de relieve un problema, que puede obligar al diseño de la adopción mensajería persistente para evitar la pérdida de mensajes"

Ejemplo de escenarios ATAM

Quality Attribute	Stimulus	Response
Availability	The network connection to the message consumers fails.	Messages are stored on the MOM server until the connection is restored. Messages will only be lost if the server fails before the connection comes back up.
Modifiability	A new set of data analysis components must be made available in the application.	The application needs to be rebuilt with the new libraries, and the all configuration files must be updated on every desktop to make the new components visible in the GUI toolbox.
Security	No requests are received on a user session for ten minutes.	The system treats this session as potentially insecure and invalidates the security credentials associated with the session. The user must logon again to connect to the application.

Prototipos de Arquitectura

- Los prototipos son versiones reducidas o restringidas de la aplicación deseada, creadas específicamente para poner a prueba algunos los aspectos del diseño de alto riesgo o que puedan ser mal entendidos.
- Los prototipos se utilizan normalmente con dos objetivos:
  - Prueba de concepto: ¿Puede la arquitectura como esta diseñada construirse en una forma que pueda satisfacer los requerimientos?
  - Prueba de tecnología: ¿La tecnología (middleware, aplicaciones integradas, librerías, etc.) seleccionadas para implementar la aplicación se comportan como se espera?

Créditos

- **Software Architecture in Practice, Second Edition**. Len Bass, Paul Clements, Rick Kazman
- **Essential Architecture.** Ian Gorton.
- **Documenting Software Architectures, Views and Beyond. Second Edition.** Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford.
- **Curso de Arquitectura de Software, Universidad de Chile.** Cecilia Bastarrica, Daniel Perovich.