MEDICIÓN DEL TAMAÑO DEL SOFTWARE EN APLICACIONES SOA CON PUNTOS DE FUNCIÓN COSMIC

Mirella Pérez Falcón

CONTENIDO

Conceptos básicos de SOA

- **Arquitectura de servicios independientes entre sí** que se acoplan para soportar los diferentes requerimientos de los usuarios y los procesos de negocio.
- Servicios **débilmente acoplados y altamente interoperables.**
- **Amplio rango de tecnologías y protocolos** (ej.: REST, RPC, DCOM, CORBA, Web Services, etc.).
- OASIS define SOA como lo siguiente:

  Un paradigma para organizar y utilizar capacidades distribuidas bajo el control de diferentes propietarios y dominios. Manera uniforme de ofrecer, descubrir, interactuar y usar dichas capacidades y producir los efectos deseados de manera consistente y medible.

Principios de SOA

Arquitectura en **capas de abstracción compuesta por servicios Web**: reutilizables, compartidos e interoperables.

Principios según el *diseño y definición de SOA centrándose en el comportamiento* :

Reusabilidad
Contrato de servicios
Servicios de Acoplamiento Suelto
Abstracción de servicios
Empaquetamiento del servicio
Autonomía de servicios (o encapsulación)
Sin estado
Descubrimiento de servicios

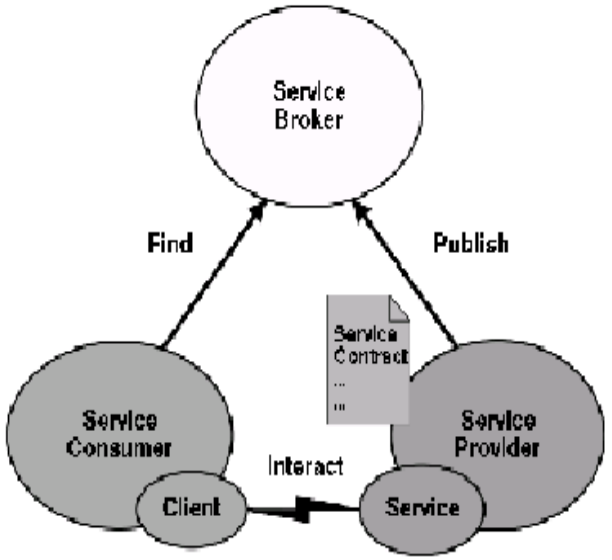
Componentes de la arquitectura SOA

En el nivel más alto de SOA encontramos tres **componentes:**

- *El servicio*
- *El directorio*
- *El cliente*

Colaboraciones entre componentes:

- *Localización de servicios*
- *Publicación de servicios*
- *Comunicación entere servicios y clientes*



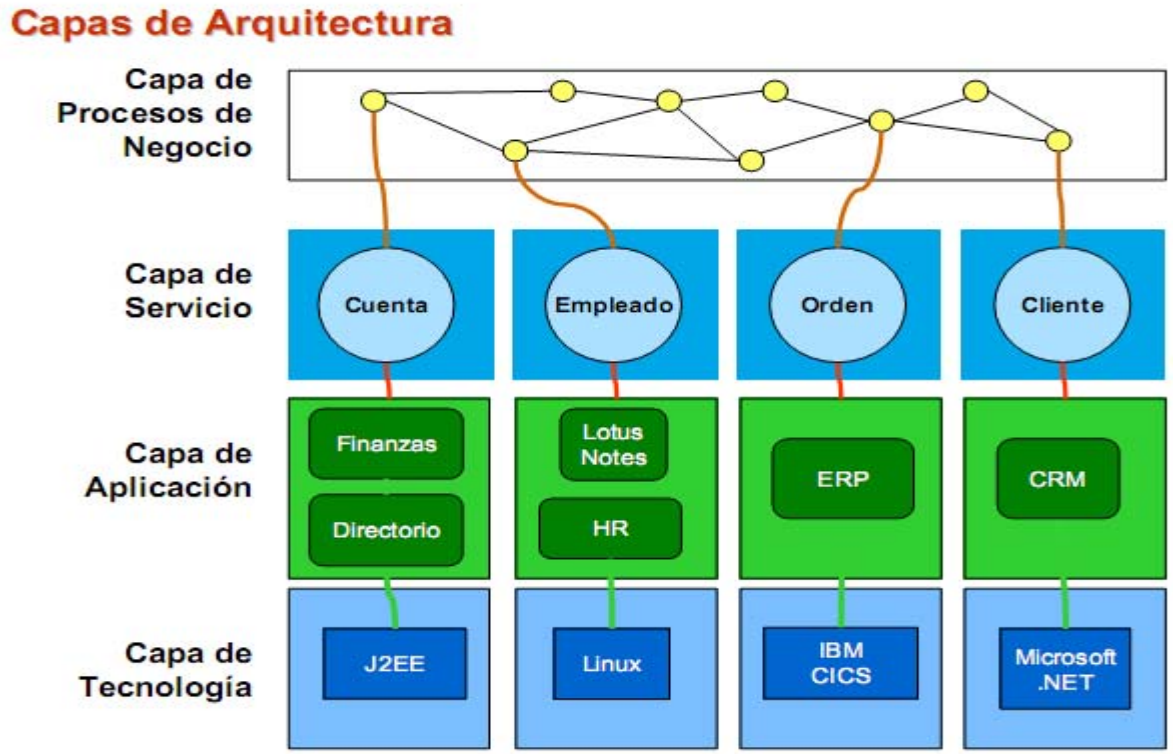
Tipos de servicios y niveles

- Servicios de negocio
- Servicios de integración

Ilustración: Jerarquía en la que los servicios de negocio en una capa pueden estar compuestos de servicios colaboradores o orquestados en un nivel inferior.

- El concepto de CAPAS es fundamental en la IDENTIFICACIÓN DE SERVICIOS.

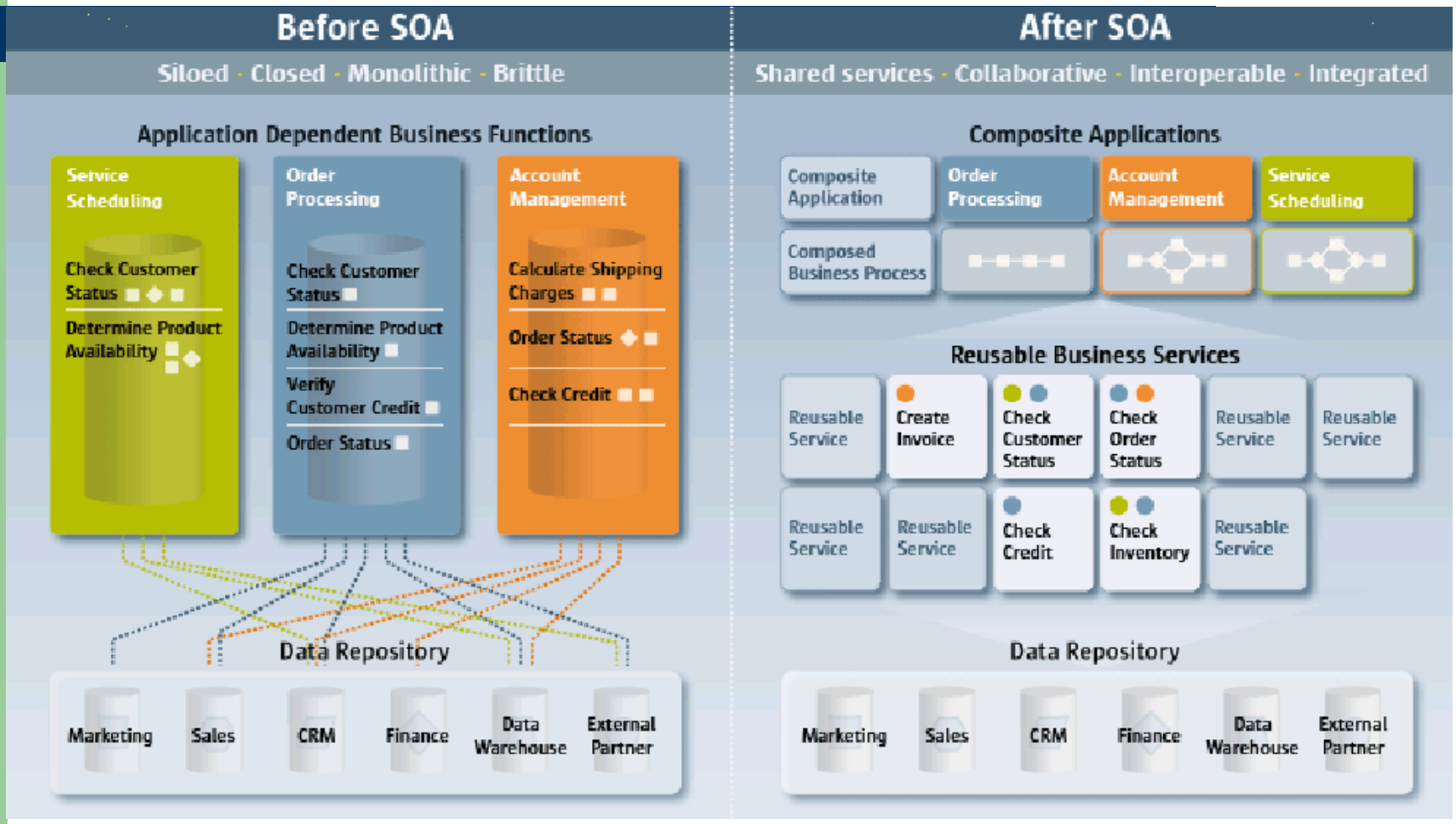
Capas de SOA


Capas de Arquitectura
Capa de Procesos de Negocio
Capa de Servicio
Cuenta
Empleado
Orden
Cliente
Capa de Aplicación
Finanzas
Directorio
Lotus Notes
HR
ERP
CRM
Capa de Tecnología
J2EE
Linux
IBM CICS
Microsoft .NET

Beneficios de SOA frente a una arquitectura tradicional

- **Reduce la complejidad**
- **Reutiliza los servicios**
- **Integra aplicaciones** → menor coste de mantenimiento e integración
- Orientada a **procesos** y enfocada al **cambio**
- **Independencia** entre las aplicaciones, la infraestructura y plataforma tecnológica.
- Posibilidad de **reconfigurar sus recursos de TI** sin necesidad de realizar una integración profunda.

Antes y después de SOA



SOA y Web Services

- Web Services : caso particular de servicio: **componente software identificado por un URL** cuyas interfaces públicas están definidas y descritas usando XML.

- Se puede implementar cualquier tecnología basada en servicios.

- **Servicios Web para SOA**:

  XML: *representación de datos.*
  HTTP: *Protocolo estándar Web.*
  SOAP: *intercambio de datos.*
  WSDL *(Lenguaje de Descripción de Servicios Web)*
  UDDI *(Descripción, descubrimiento e Integración Universal)*

Puntos de función IFPUG

Asume un modelo de software que consta de dos partes:

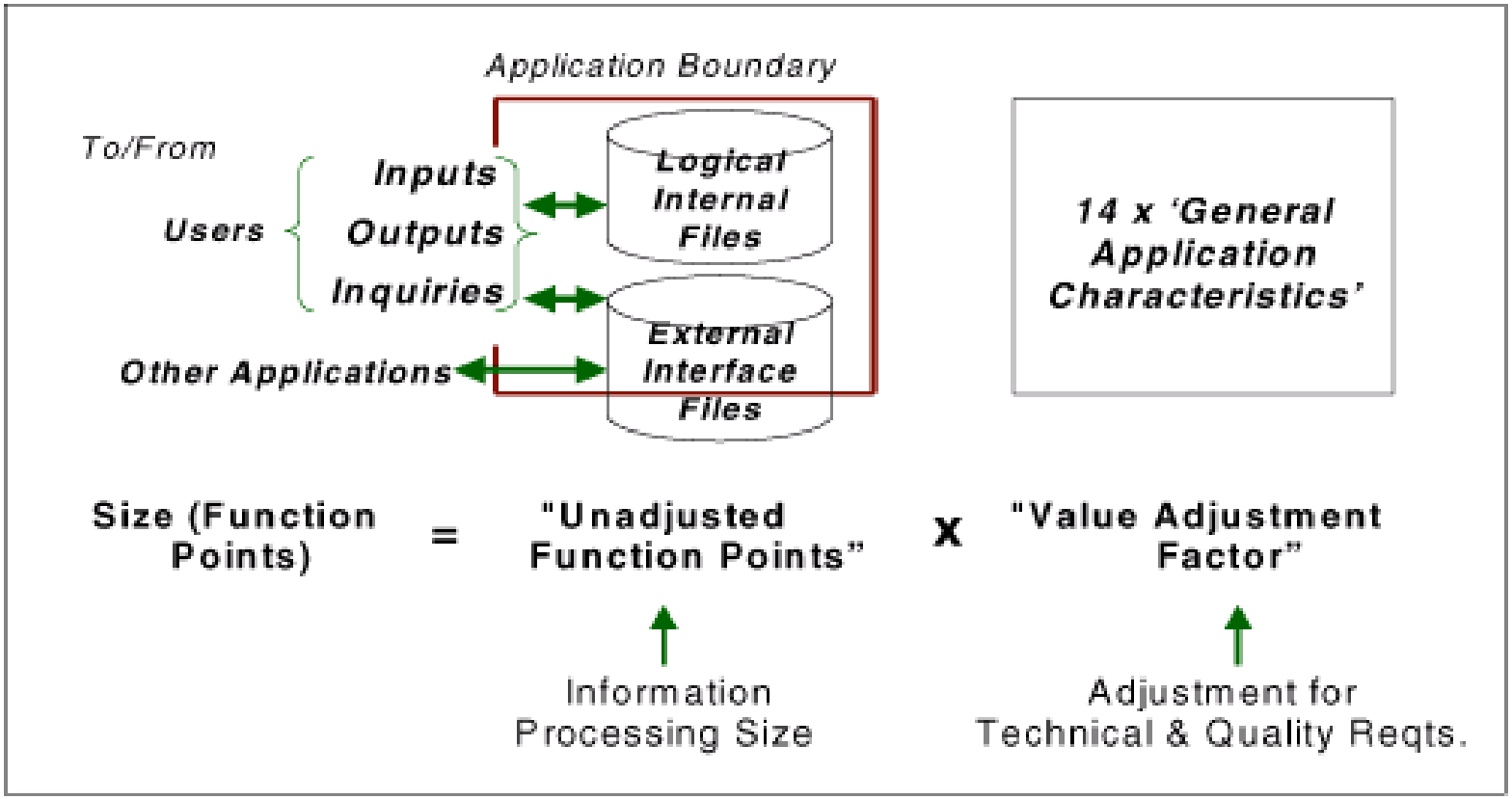
- ***Modelo de Datos:*** **archivos lógicos internos y externos.**
- ***Modelo de Transacciones:*** operaciones que el usuario desea hacer con los datos: **entrada, salida y consulta**.

Función de medición

Variables de entrada: los tipos de función y la complejidad de las mismas.

Frontera de la aplicación (application boundary) separa el sistema a ser medido del dominio del usuario.

Puntos de función IFPUG



IFPUG Y SOA

- **Limitado superiormente**:

  Sin importar el tamaño o complejidad, un proceso elemental de entrada no puede obtener más de 6 FP y un fichero no puede obtener mas de 15 FP.

  Diferentes interpretaciones con respecto a la agregación es contrario a la autonomía de procesos elementales: fácilmente conducen a diferentes asignaciones numéricas para el mismo conjunto de requisitos.

  IFPUG no es “asociativo” en un sentido matemático, o **“el total no es igual a la suma de las partes”.**

IFPUG Y SOA

- Se emplea un **factor de ajuste** para requisitos no funcionales:

  Tal factor no toma en cuenta la **reusabilidad del software**: *una función (servicio u operación de servicio) provista varias veces a diferentes sistemas se cuenta esas tantas veces, independientemente de que sea e implementada una o varias veces.*

- **Caja negra:** *ningún proceso interno puede ser identificado y medido, a no ser que* ***dispongamos de otro proceso*** *que envie o reciba datos desde o hacia el usuario final fuera del* ***límite del sistema completo****.*

Puntos de función COSMIC

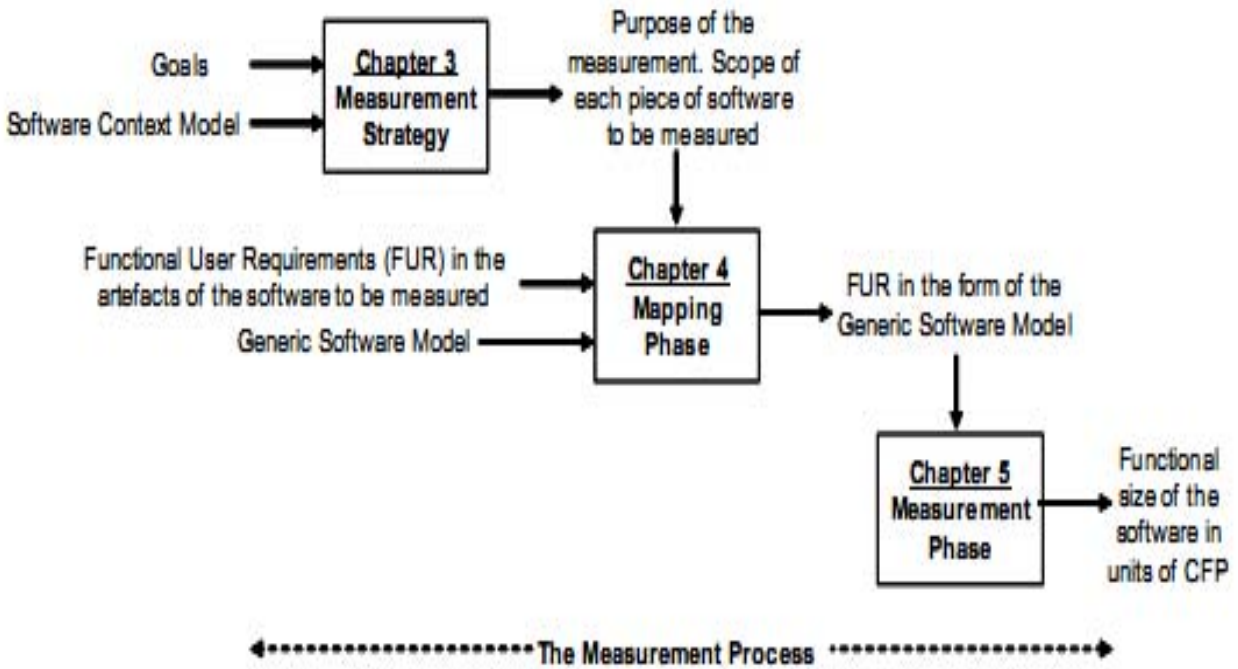
- Aplicable a una ***arquitectura en una capa o en múltiples capas, o a cualquier peer item dentro de una capa (layer).***

- **Requisitos Funcionales del Usuario (FUR):** prácticas y procedimientos de usuario que el software debe llevar a cabo para cumplir las necesidades de los usuarios.

- **Tamaño de FUR:** se obtiene mediante la suma aritmética del tamaño de los movimientos de datos asociados al mismo, según:

$$Size_{Cfsu}(\text{functional process}_i) = \sum \text{size}(\text{entries}_i) + \sum \text{size}(\text{exits}_i) + \sum \text{size}(\text{reads}_i) + \sum \text{size}(\text{writes}_i)$$

Puntos de función COSMIC



COSMIC Y SOA

- Posibilidad de considerar:
  - ***diferentes capas unidas*** *mediante* ***componentes software separados dentro de cada capa y****,*
  - ***diferentes perspectivas de medida*** *(a parte de "usuario final" , basados en distintos propósitos de medida.*
- Las escalas de valor **no están limitadas superiormente.**
- COSMIC es **más asociativo** que IFPUG: **a *procesos muy complejos (servicios u operaciones de servicios) se le asignan números mayores que a procesos.***

COSMIC Y SOA

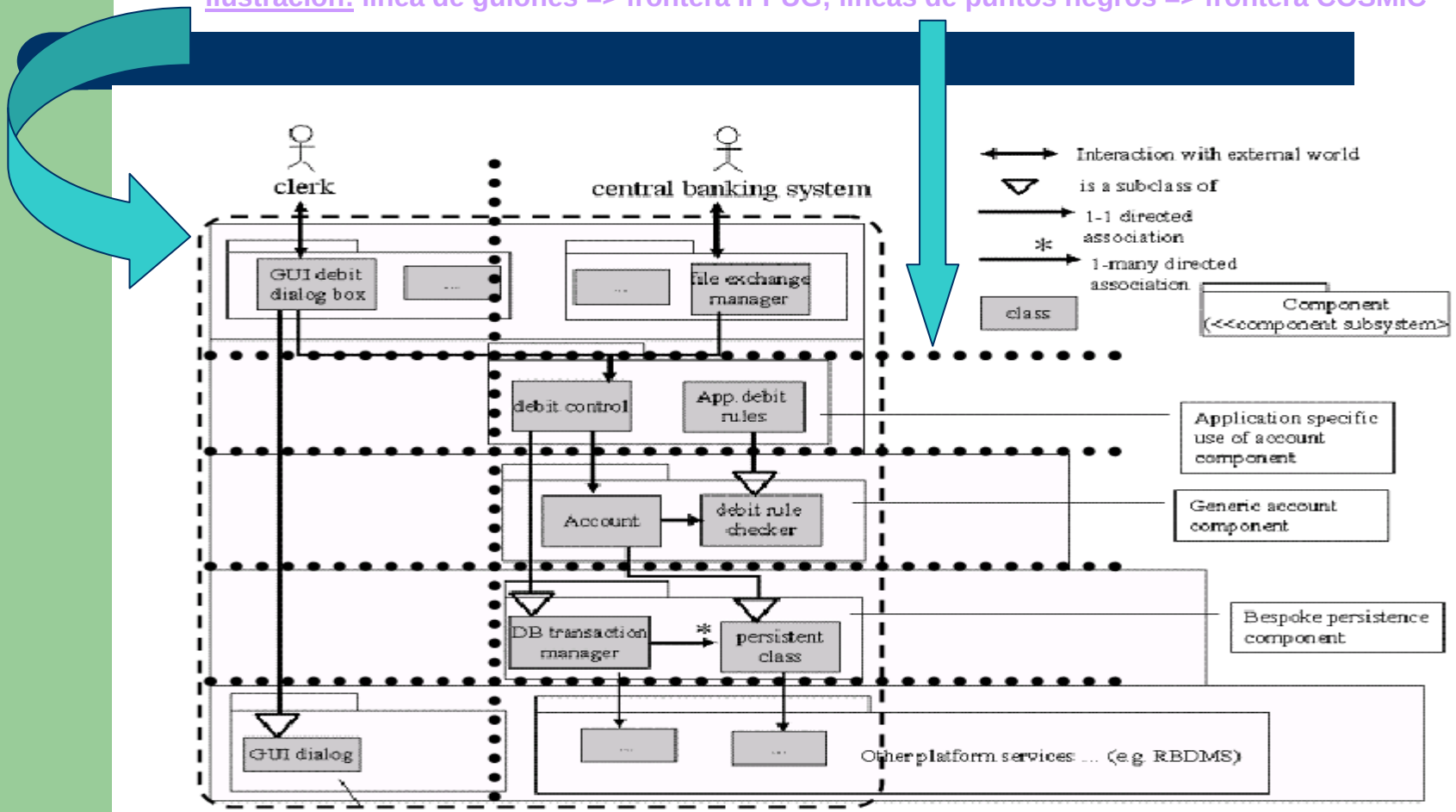
- COSMIC **no emplea ningún factor de ajuste**.

- La capacidad de descomponer el sistema en *capas* pares de elementos software dentro de cada capa permite **considerar algunos procesos sólo una vez en lugar de muchas**. Lo que a su vez permite hacer que los **procesos internos sean visibles** para los fines específicos de medición.

- Enfoque de ***caja blanca: cada capa y elemento, cuando son identificados, tiene su propia frontera***, *revelando a su vez procesos internos a sus usuarios (otras capas o elementos).*

Arquitectura en capas:

Ilustración: línea de guiones => frontera IFPUG; líneas de puntos negros => frontera COSMIC



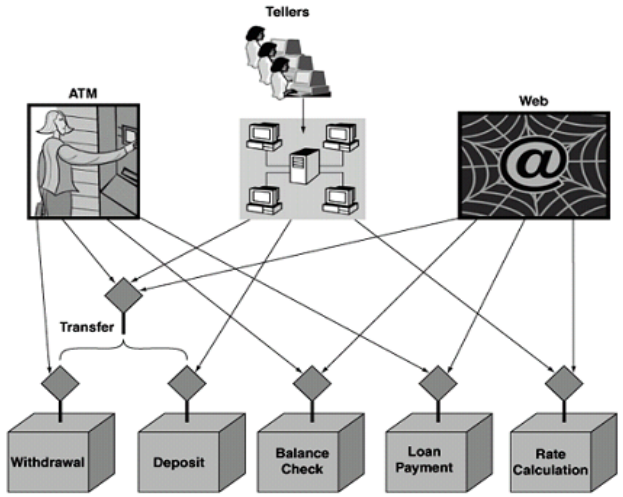
Aplicaciones / límites monolíticos

- En Cosmic:
  - **Particiones** ➔ diferentes posibilidades dependiendo del propósito de la medida y de la arquitectura.
  - Cualquier capa es una colección de unidades (elementos) software ➔ **servicios.**
- ***Cada unidad (servicio) debe tener una interfaz mediante la cual sus operaciones pueden ser iniciadas o accedidas.***

Servicios autónomos y diferentes canales software

- Cada una de las fronteras separadoras en el método COSMIC y su enfoque de caja blanca, permitiría identificar los *servicios internos como requeridos y provistos autónoma e independientemente.*

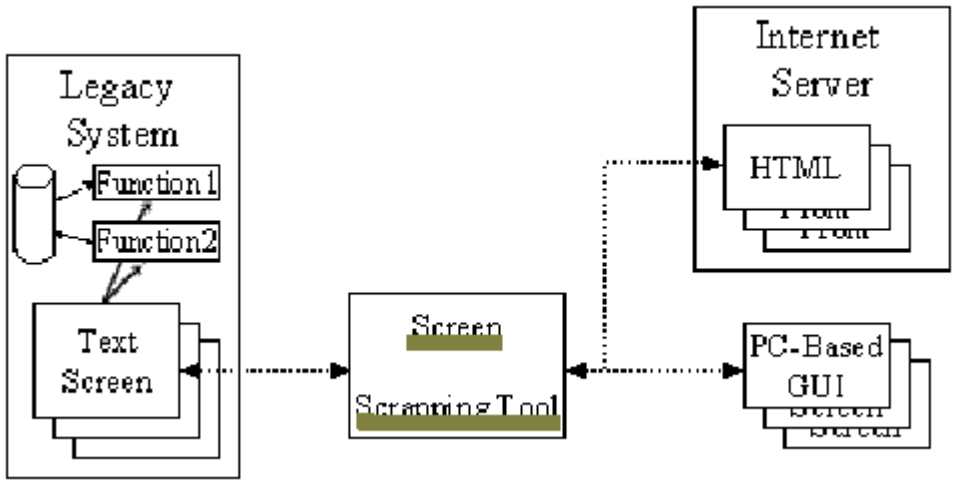
- Diferentes canales proveen diferentes funciones, incluso cuando la situación inicial y los resultados finales puedan ser lo mismo.



accessing and composing services
Several boundaries are possible (not shown).

Compartición de datos

- Cosmic permite *identificar los servicios intermedios como autónoma e independientemente requeridos y provistos* (ej.:Screen Scrapping Tool en compartición de datos entre 2 o más aplicaciones).



Complejidad de los servicios (procesos)

- ***Estimación del esfuerzo:*** La cuestión de tener delimitados y no delimitas escalas de valores para los objetos de medida.
- Ciertos casos en IFPUG emplean una escala de ***tres niveles de complejidad.***
- COSMIC provee escalas de rango abiertas para tomar en cuenta ***funciones de alta complejidad (intercambio de datos o accesos)***
- ***Mayor exactitud*** en la asignación numérica para diferenciar servicios en un entorno SOA y en otros tipos de arquitectura.

Conclusiones

- SOA es una manera de organizar soluciones que promueve **reusabilidad, crecimiento e interoperabilidad**.
- En SOA, el método de medida elegido debería ser capaz de *diferenciar funcionalidad basada en los aspectos y características de la arquitectura*.
- IFPUG no lo permite, mientras que **COSMIC** tiene más posibilidades al respecto.
- Es mejor que las *diferentes porciones de software* (servicios) se estiman *individualmente, por separado.*