**Definición / Definiciones**

1. Flujo de pantallas con las que interactúa el usuario para completar una tarea
2. Flujo de lógica de negocio dentro de una aplicación o servicio
3. Ineracción de varias personas para completar una tarea
4. Coordinación de múltiples mensajes entre sistemas para procesar una transacción de negocio
5. Coordinación de pasos para manejar datos de un repositorio

**Agrupación de WF**

* De interacción humana
* De aplicación
* Integración de la Empresa
* De gestión de datos

**Escenarios en los que es útil**

**Como “Controller” del UI**

* **Fuerza la navegación de pantalla según reglas**
  + “wizards”, MVC

**Cuando existe lógica de negocio de larga duración**

* **El usuario tiene que parar o esperar en medio del proceso**
  + Sin la habilidad de “hibernar” (persistir) el estado, el programador tiene que diseñar de arriba abajo todo el proceso de salvar estados y revivirlos.

**Flujos de proceso dinámicamente modificables**

* **Cambios en la lógica sobre la marcha**
  + “Esta vez que firme fulano…”

**Abstracción de las Reglas de la Lógica de Negocio.**

* **Conseguir separar reglas y lógica de negocio debido a que se incorporan escenarios.**
  + Un mismo formulario (de petición hipoteca) puede ser cumplimentado en una u otra fase del proceso, y aun siguiendo la misma lógica, se aplicarán diferentes reglas.

**Agregador de Servicios Web**

* **En procesos en que se recopila información de distintos servicios y, en función de los datos, ajustar la lógica a seguir.**

**Cuando existen Procesos de Negocio de larga duración**

* **Son aquellos aplicativos de servidores que interactúan entre sí, implementado multiproceso, asíncronos, persistencia, correlación de mensajes, escalabilidad, fiabilidad, transacciones…**

**B2B**

* **Cuando tienen que interactuar distintas organizaciones, necesitando compartir no aplicativos sino datos.**

**Abstracción de Reglas de los Procesos de Negocio.**

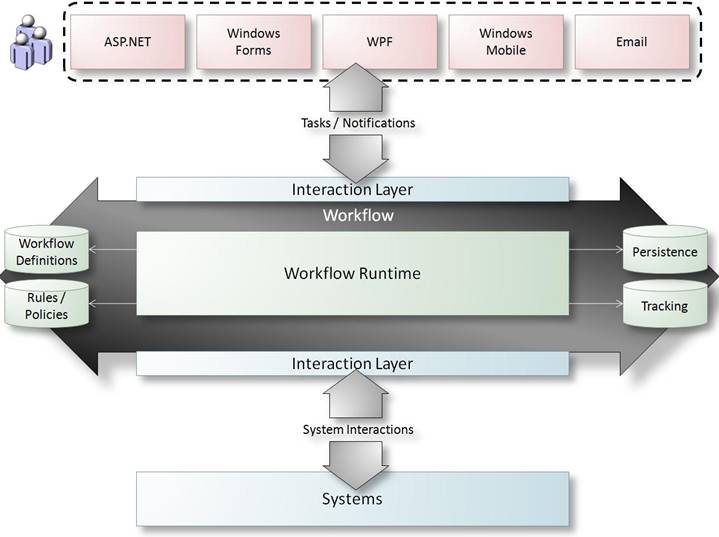
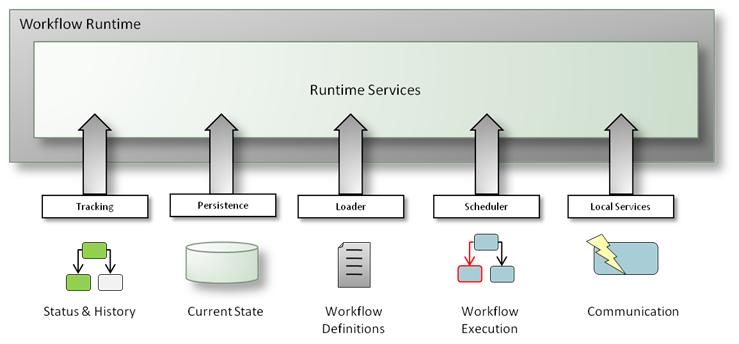
* **Repositorio Corporativo de Reglas**
  + El objetivo es conseguir un repositorio central de reglas compartidas que pueden ser invocadas por todas las aplicaciones de la Empresa.
  + Consistencia
  + Interacción de usuarios no programadores

**ESB**

* **Infraestructura estandarizada de comunicación entre todos los servicios.**
  + Entreprise Service Bus
  + Message Broker
  + Publish Subscribe

**Servicios centrales (core services) que ofrece WF**

* Scheduling
* Runtime Engine
* Transacción de operaciones
* Persistencia
* Tracking



**Host Services**

* Scale-out configuration
* Load balancing
* Fail-over
* Archiving and purging
* Identity and impersonation
* Multi-environment deployment model
* Throttling
* Thread management
* Memory management
* Service isolation
* Exception configuration
* Failed-message management
* Message tracking
* Health monitoring
* Utilization/Performance tracking
* Composite-process state management
* Scripting
* Disaster recovery
* Regulatory compliance
* Configuration management

**Escogiendo el modelo de WF**

**Actividades sueltas (leaf activities)**

* **Conocen cómo ejecutar código arbitrario dentro de su ámbito**

**Actividades compuestas (Composite)**

* **Más que ejecutar código, orquestan sus actividades hijas de forma determinada**

**Modelo Secuencial**

* **Enfoca en alcanzar un objetivo, organizando las rutas para conseguirlo**

**State Machine**

* **Enfoca en los eventos o estados del proceso, tanto si se llega o no a una culminación.**

**Escenarios**

**Validación de documentos**

A primera vista, un secuencial parece lógico (crear, revisar, validar, entregar…) pasa de un actor a otro

Pero si un actor devuelve varias veces al predecesor el documento para efectuar cambios…

Ahora parece más lógico un State Machine

**Human Wf**

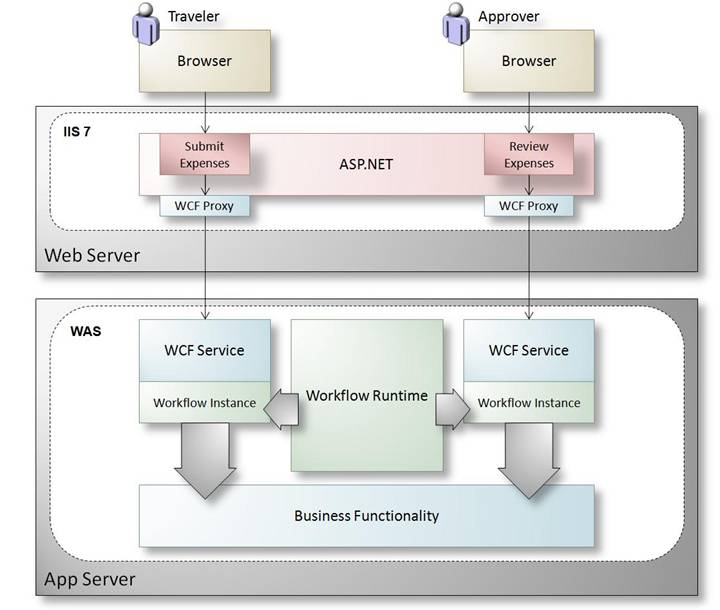
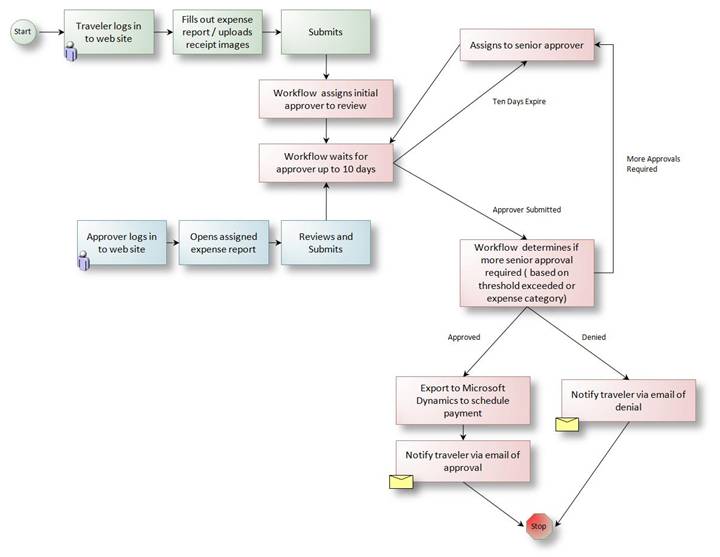
Herramientas para orquestar la interacción hombre-sistema de acuerdo a reglas de negocio

Ayuda a contestar:

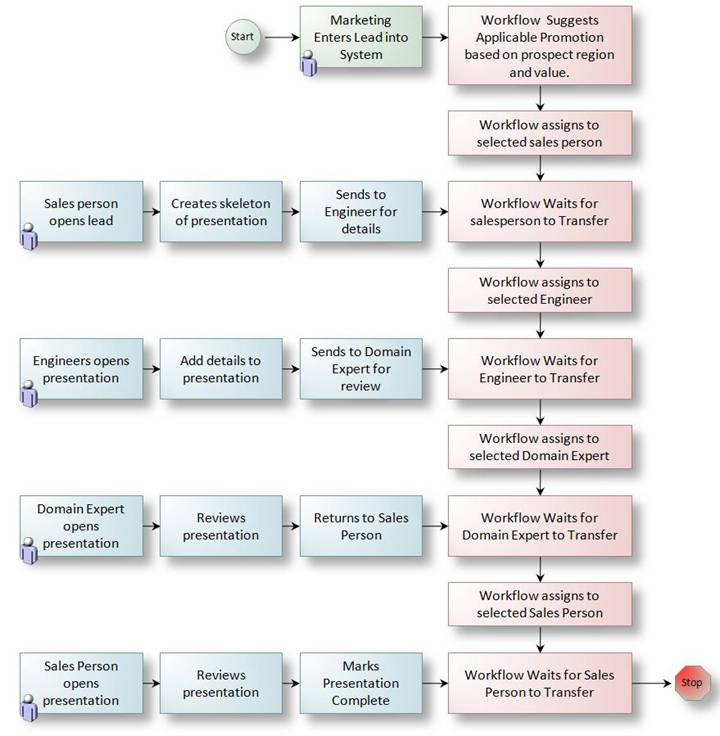
* ¿Dónde está ahora mi documento?
* ¿Quién lo retiene?
* Cuánto le queda al proceso
* Qué pasos están pendientes

Ejemplos:

* Revisión y Aprobación de documentos
* Colaboración

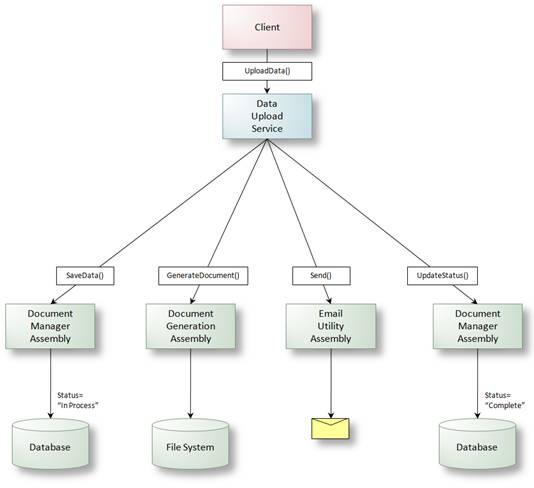
**Aprobación de gastos de viaje**

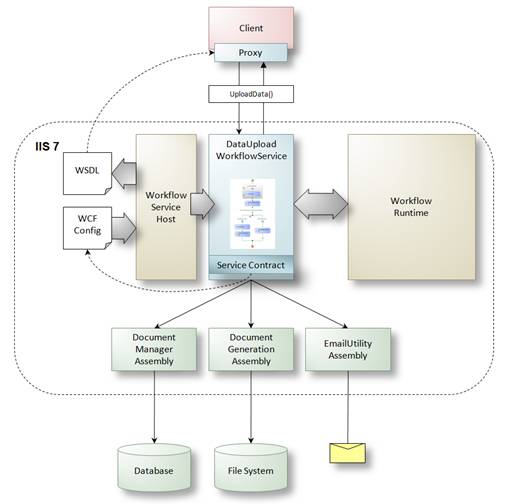
**Colaboración**



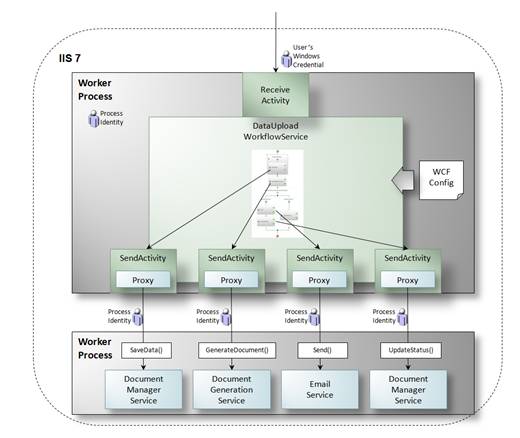
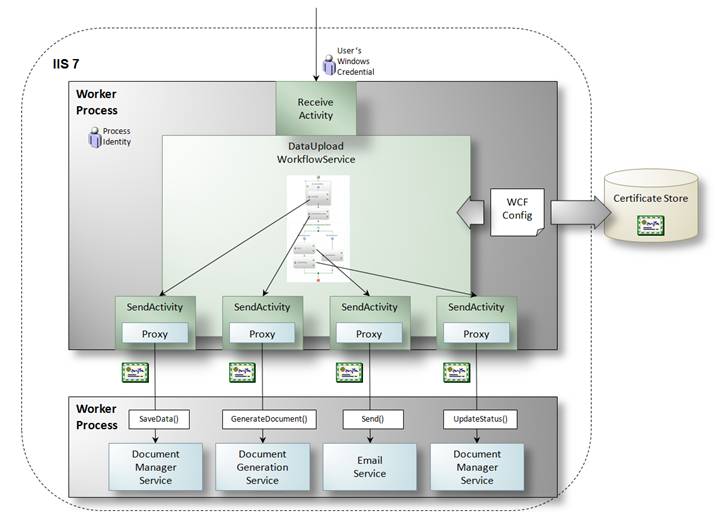
**Wf service de generación de documentos**

Descripción de la funcionalidad

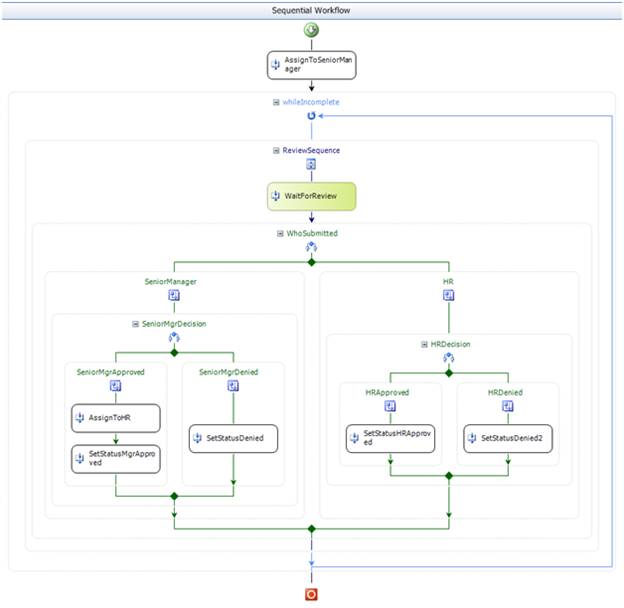




**La seguridad**

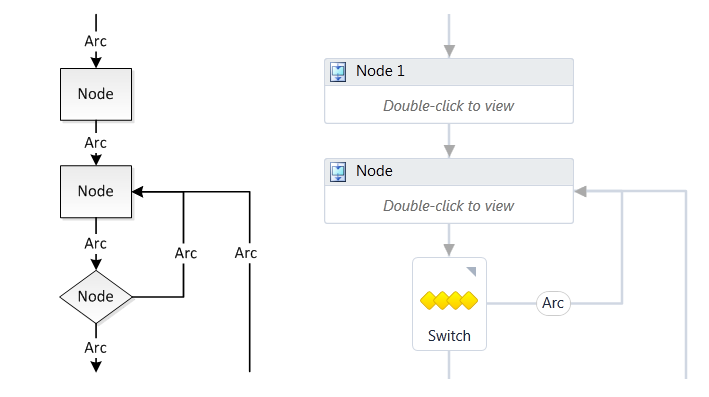


**Modelado de WF**

El modelo secuencial (Sequential)

**Flowchart**

Cuando se tiene un proceso que a menudo es secuencial por naturaleza, pero con “loop backs” en algunas circunstancias



<http://www.theworkflowelement.com/workflow-patterns/>

**Qué modelo de WF elegir**

El wf incluye muchos procesos en determinado orden, con poca interacción humana, pocas recursiones

* Secuencial

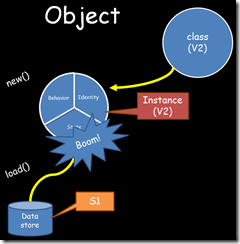
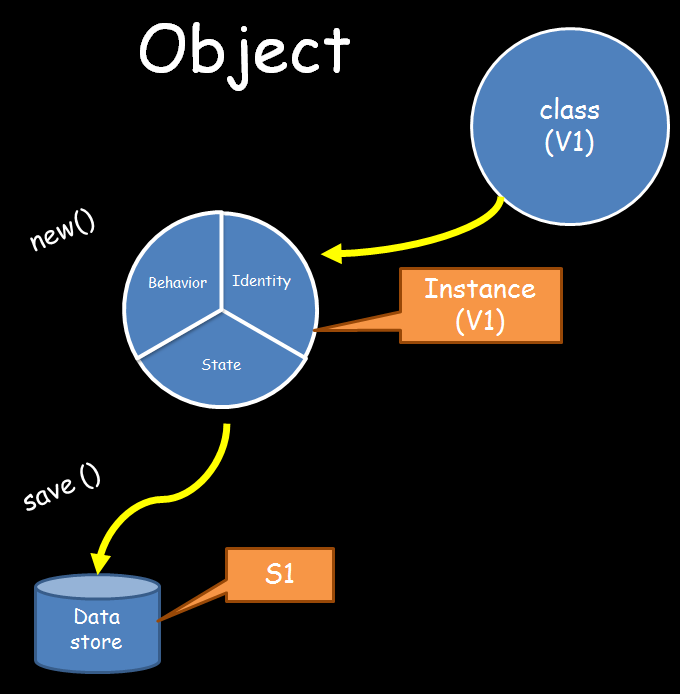
Cuando contiene recursiones

* Flowchart

Cuando es totalmente dependiente de la interacción humana

* State Machine

**El problema del versionado**



**Abordar el problema del versionado**

* Always use the Name and Namespace parameters on all contracts.
* Always use the Order parameter on data members.
* Implement the IExtensibleDataObject on all data contracts.
* Use namespaces for contract versioning.
* Always remember that new versions require new endpoints.
* When using strict schema validation, do not change contracts. Create new versions.
* When removing an operation from a service contract, create a new version.
* When changing the signature of an operation, create a new version.

**Los data objects:**

* Nunca jamás usar entidades del dominio
* Usar siempre DTO’s para argumentos o variables en los WF de larga duración
* Asignar IsRequiered = false en los DataMember de las propiedades Nullable
* Asignar el Name en los DataMember
* Asignar el Orden en los DataMember
* Asignar el nombre en los [DataContract](http://msdn.microsoft.com/en-us/library/system.runtime.serialization.datacontractattribute.aspx)

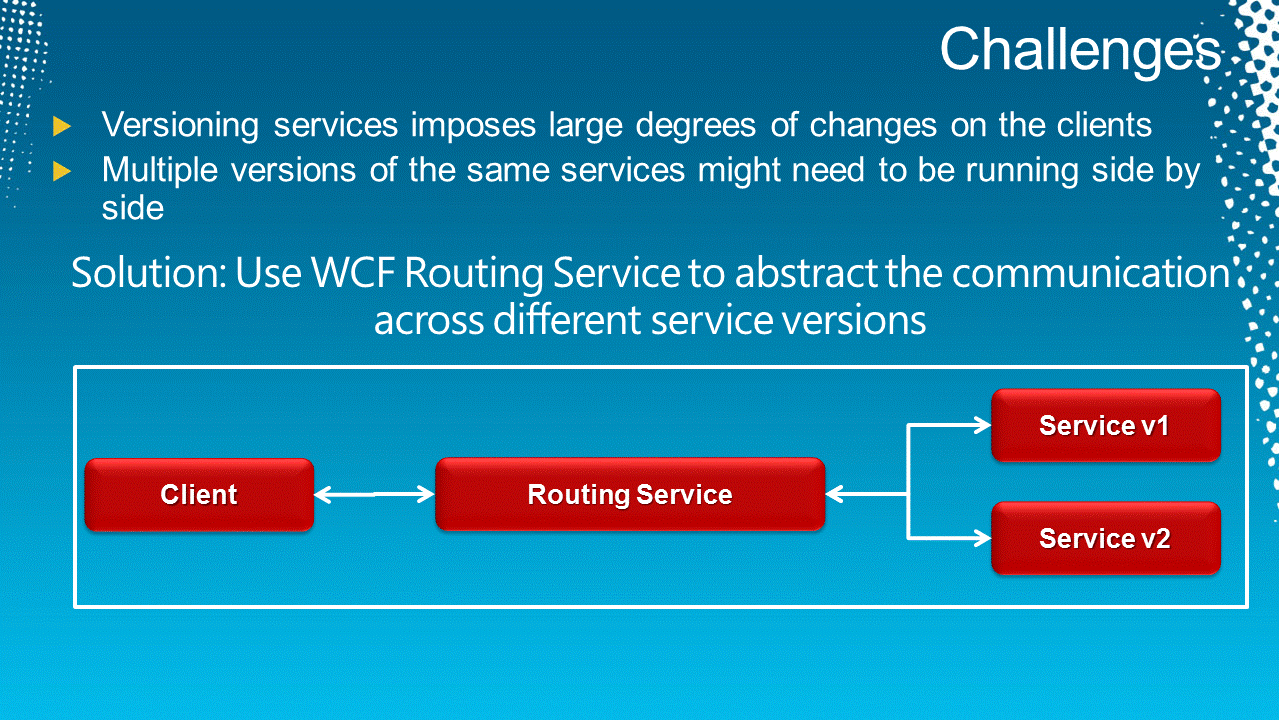
|  |
| --- |
| [DataContract(Name="Category")]  public class Category  {  [DataMember(IsRequired = false, Name = "Id", Order=0)]  public int Id { get; set; }  [DataMember(IsRequired = false, Name = "Products")]   * public List<Product> Products { get; set; } } |

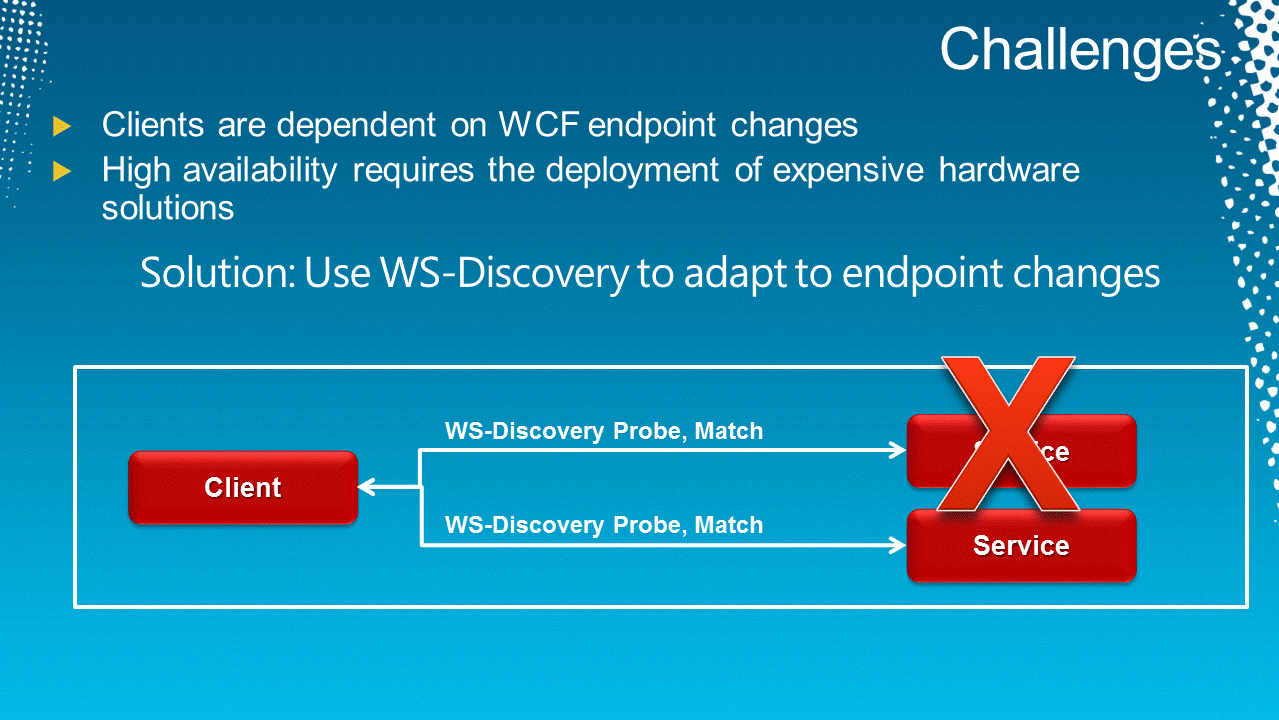
**Non-Breaking Changes:**

* Changes to the activities code
* Changes which don’t affect the number or types of arguments/variables
* Addition or removal of member fields or properties

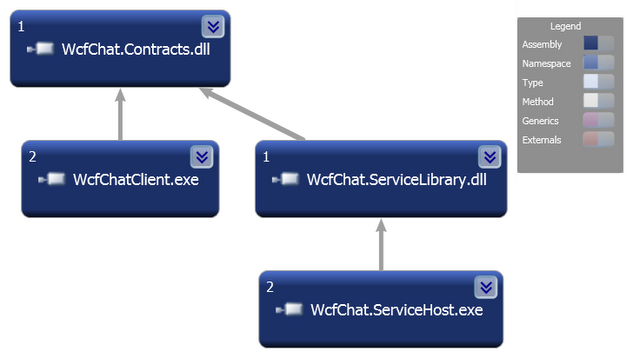
**Breaking Changes:**

* Renaming or removing methods that had been used as bookmark, fault or completion callbacks
* Adding new arguments
* Adding new variables
* Adding new children

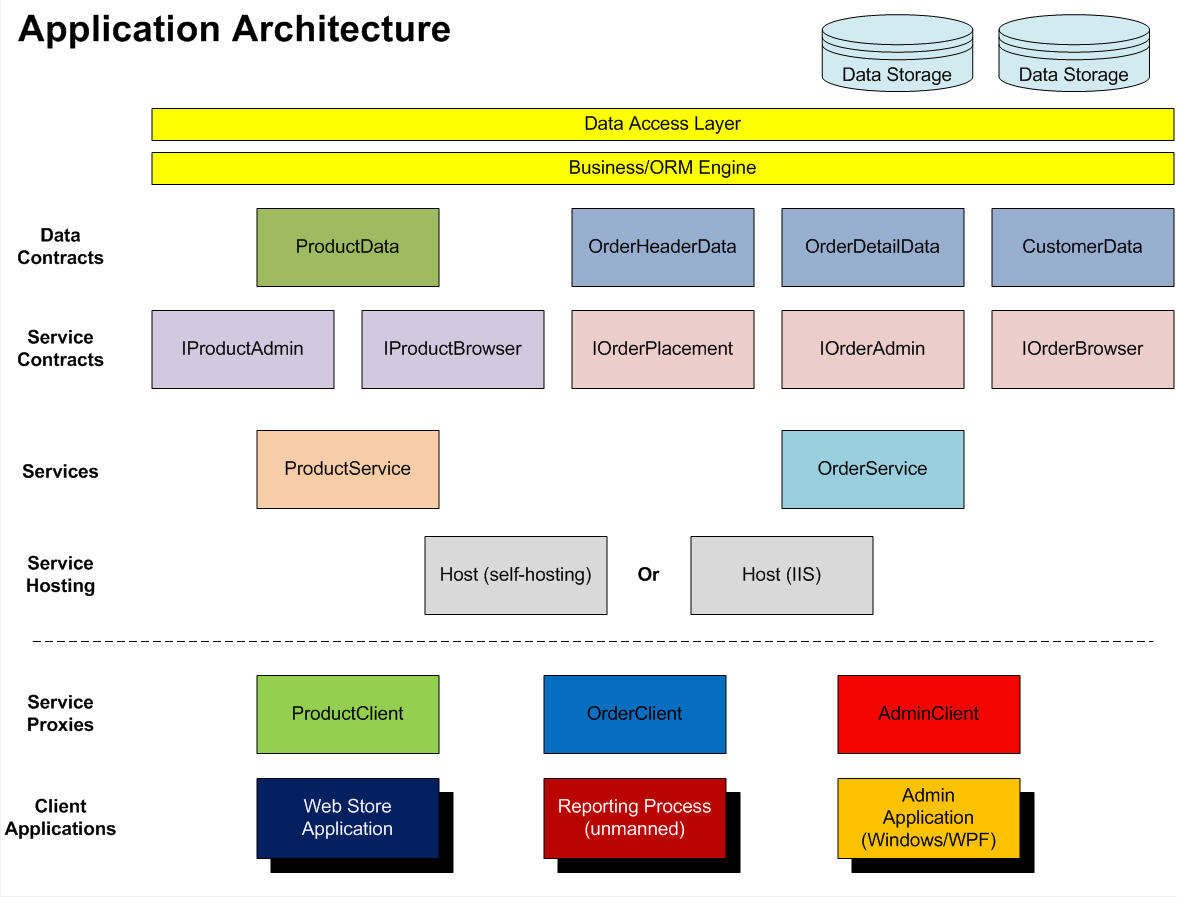


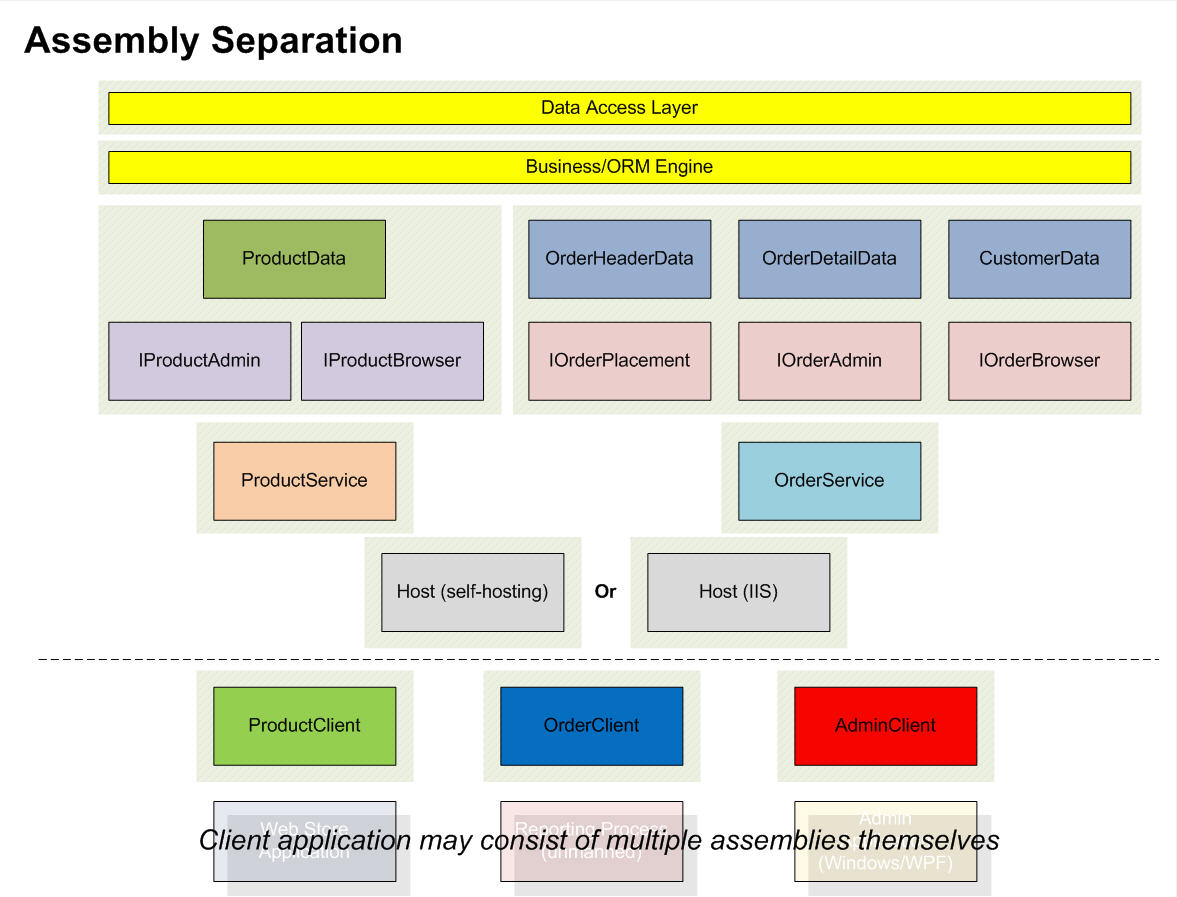


**Capas de una arquitectura de WF**

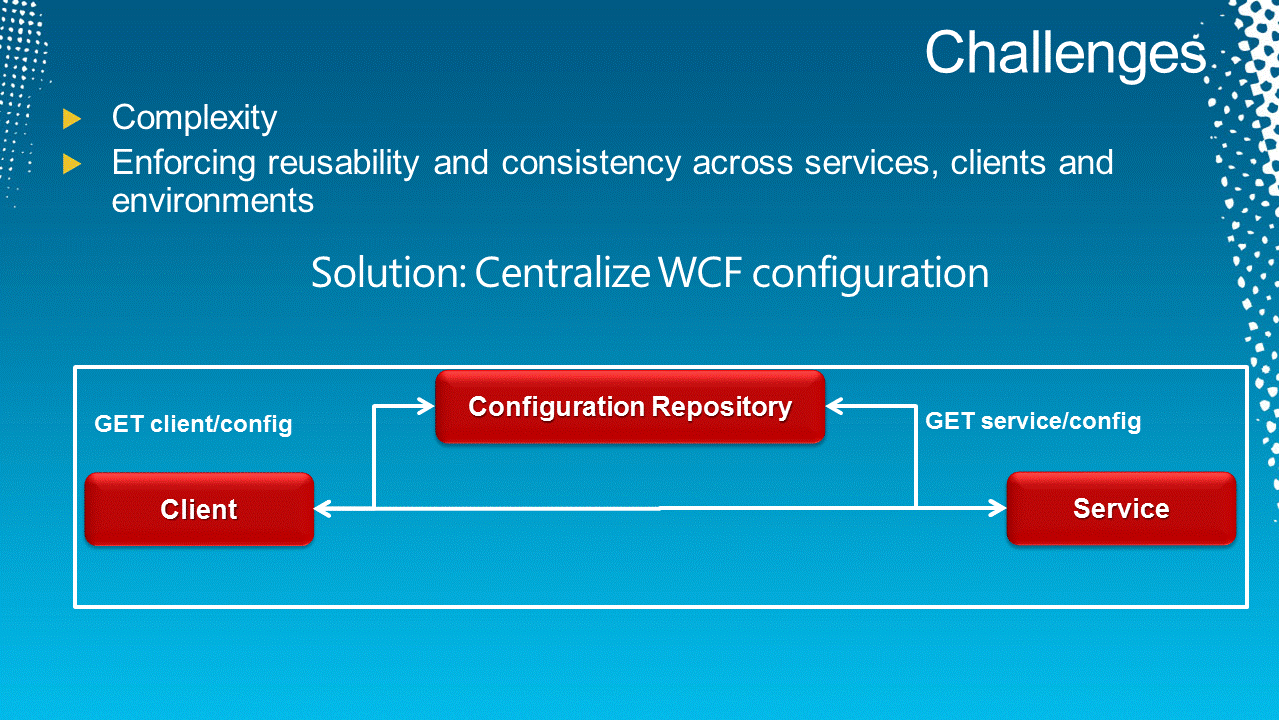


Todos los contratos en una dl

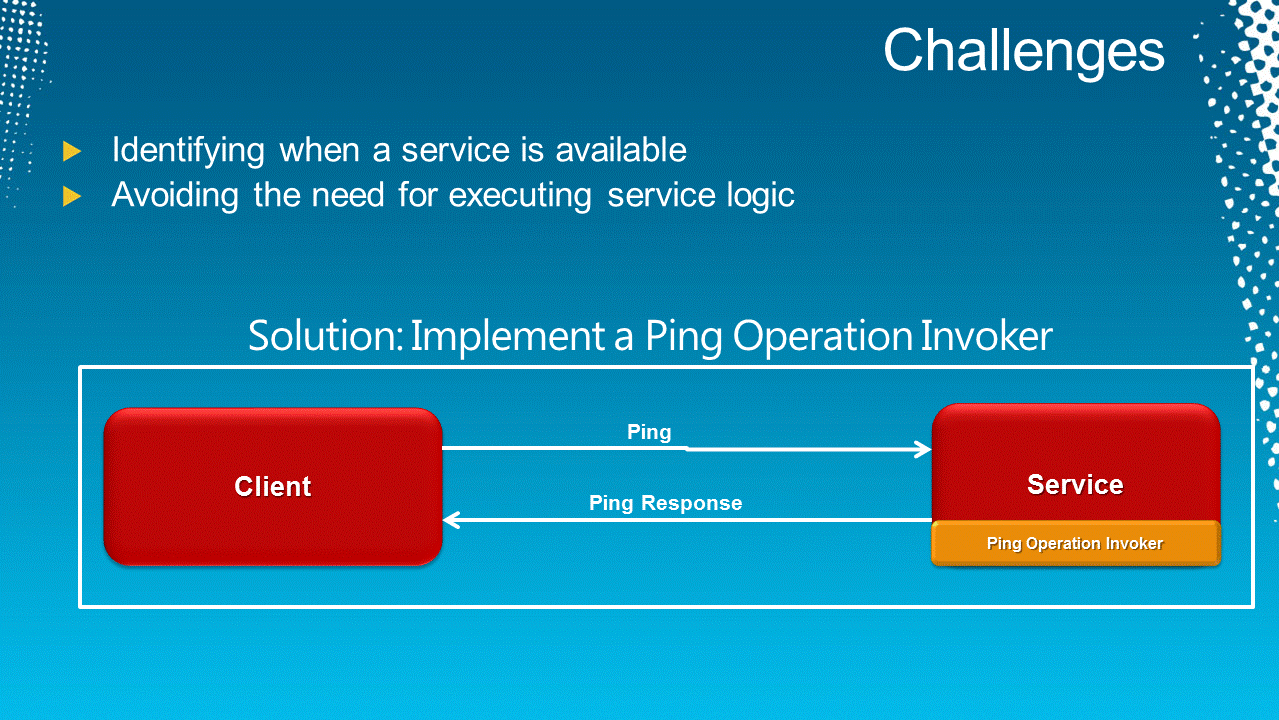




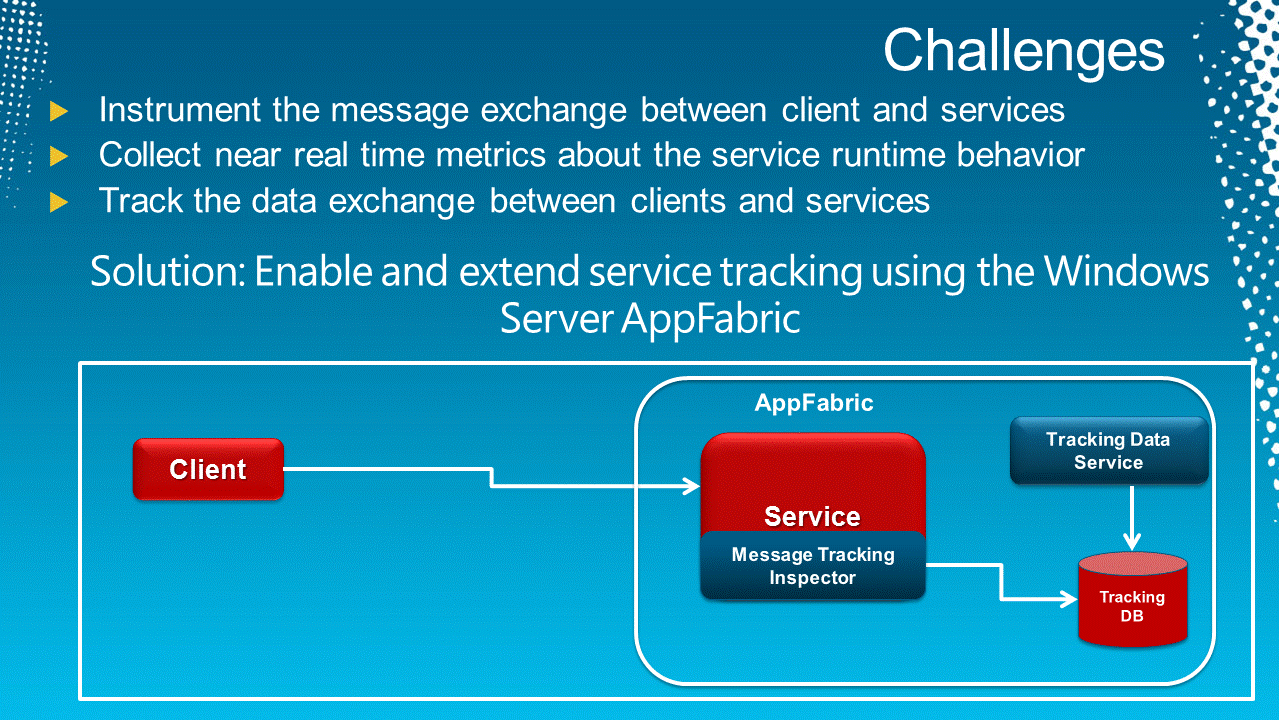
**Configuración centralizada**



**Determinar la disponibilidad de servicio**



**Métrica y seguimiento:**



**Más problemas (prestaciones)**

