



SOA != OO

Andrea Saltarello

Software Architect @ Managed Designs S.r.l.
andrea.saltarello@manageddesigns.it
<http://blogs.ugidotnet.org/pape>



GUISA
Gruppo Utenti Italiani
Solution Architect



<http://creativecommons.org/licenses/by-nc-nd/2.5/>

Chi sono

- Solution Architect @ **Managed Designs S.r.l.**
(<http://www.manageddesigns.it>)
- Presidente dello **User Group Italiano .NET**
(**UGIdotNET** – <http://www.ugidotnet.org>)
- Fondatore del **Gruppo Italiano Utenti Solution Architect** (**GUISA** – <http://www.guisa.org>)



demo

OO vs. SOA: da ASMX a WCF



GUISA
Gruppo Utenti Italiani
Solution Architect

OO vs. SOA

SOA \neq OO: SOA è "message oriented"

I servizi:

- veicolano messaggi composti da *dati* privi di *comportamento*
- non distribuiscono oggetti



SOAP!=Simple Object Access Protocol

La versione 1.1 di **SOAP** permetteva di “provare” ad implementare una comunicazione “RPC over HTTP”, includendo la tipologia di *encoding* **RPC** che preservava l'*identità* degli oggetti

Le specifiche **WS-I Basic Profile** rendono deprecato l'encoding RPC

A partire dalla v1.2, **SOAP** non è più un acronimo



demo

SOAP!=Simple Object Access Protocol



GUISA
Gruppo Utenti Italiani
Solution Architect

SOA

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

[OASIS]



GUISA
Gruppo Utenti Italiani
Solution Architect

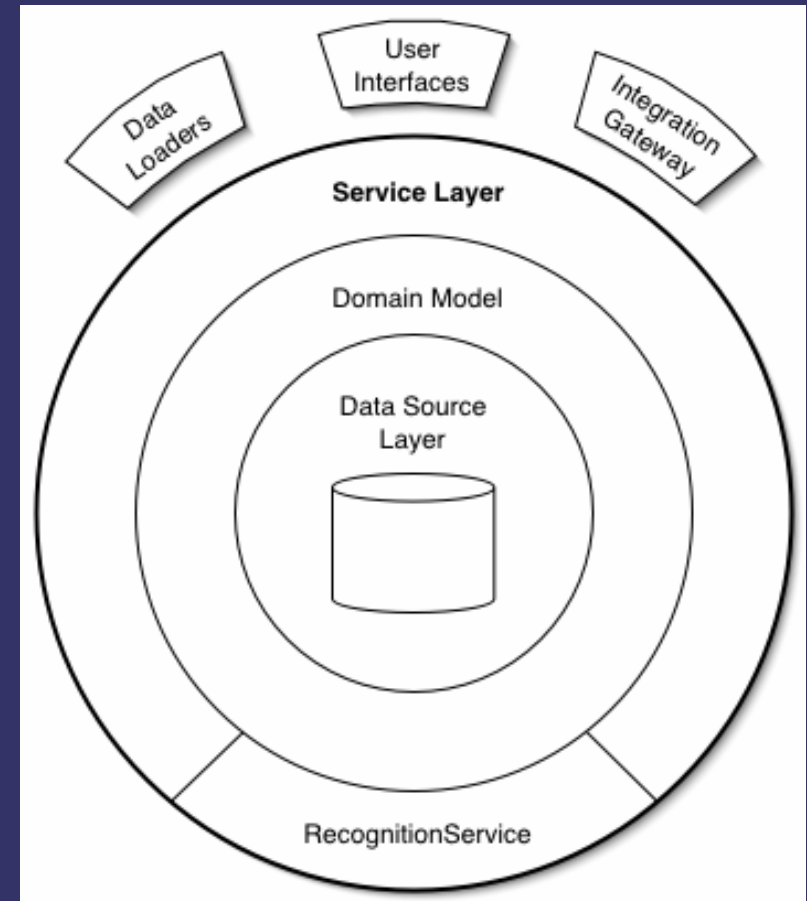
SOA Tenets (Don Box, Pat Helland)



“Service Layer” [P of EAA, 133]

Defines an application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation.

A Service Layer defines an application's boundary [Cockburn PloP] and its set of available operations from the perspective of interfacing client layers. It encapsulates the application's business logic, controlling transactions and coordinating responses in the implementation of its operations



SOA... In pratica

Quando implementiamo una SOA, i servizi fungono da **Remote Facade** [P of EAA, 388] verso la *business logic* da essi incapsulata, quindi:

- interfacce “CRUDy” non sono SOA
- gli oggetti veicolati sono dei semplici **DTO**
- Il servizio non implementa la business logic, bensì la usa (ad esempio, “scriptando” il **Domain Model** [P of EAA, 116])

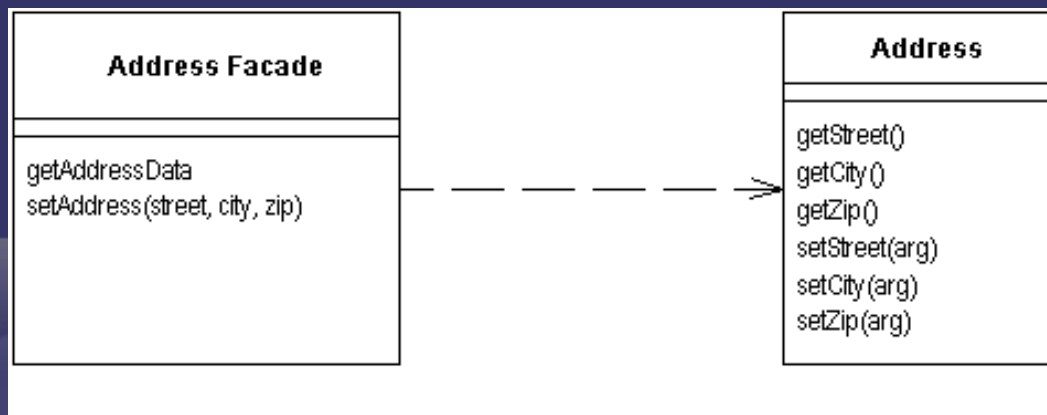


“Remote Facade” [P of EAA, 388]

Provides a coarse-grained facade on fine-grained objects to improve efficiency over a network.

Within a single address space fine-grained interaction works well, but this happy state does not exist when you make calls between processes. Remote calls are much more expensive because there's a lot more to do...

A Remote Facade is a coarse-grained Facade [GoF] over a web of fine-grained objects. None of the fine-grained objects have a remote interface, and the Remote Facade contains no domain logic. All the Remote Facade does is translate coarse-grained methods onto the underlying fine-grained objects.

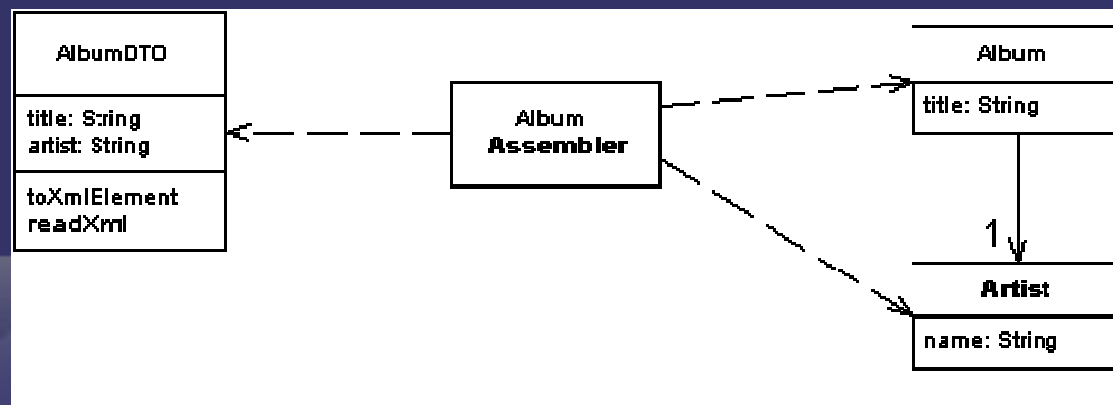


“Data Transfer Object” [P of EAA, 401]

An object that carries data between processes in order to reduce the number of method calls.

When you're working with a remote interface, such as Remote Facade (388), each call to it is expensive. As a result you need to reduce the number of calls, and that means that you need to transfer more data with each call. One way to do this is to use lots of parameters. However, this is often awkward to program - indeed, it's often impossible with languages such as Java that return only a single value.

The solution is to create a Data Transfer Object that can hold all the data for the call. It needs to be serializable to go across the connection



Anti-Pattern: CRUDy Interface

```
[OperationContract]
```

```
public void UpdateCustomer(string Name, string  
Address, string City, string Region, string PostalCode,  
string Phone) [...]
```

- Cosa c'è di "sbagliato" in questa interfaccia?
 - L'update del cliente è una semplice operazione sul database o è un processo di business?



Anti-Pattern: Über Service

```
[ServiceContract]
public class UberService
{
    [OperationContract]
    public XmlDocument Execute(XmlDocument request)
    {...}
}
```

Quale è l'errore in questo servizio?

Viola il tenet #3 – I servizi condividono solo contratti e policy

- Cosa fa questo servizio?
- Quali dati accetta?
- Quali dati restituisce?



Anti-Pattern: Boundary violation

```
[OperationContract]
public void FakeService()
{
    try {
        /* ... */
    }
    catch(SQLException ex) {
        throw new Exception("", ex);
    }
}
```

Quale è l'errore in questo servizio?

Viola il tenet #3 – I servizi condividono solo contratti e policy

– Cosa fa questo servizio?

Ricapitolando

- In ottica SOA, i servizi:
 - non distribuiscono *oggetti*, bensì *messaggi*: il servizio ed il consumer concordano sul contratto da essi definito
 - sono autonomi, e fungono da *remote facade* verso *business logic* ad essi locale
- Se dobbiamo distribuire oggetti, .NET offre una **vera** tecnologia ORB: si chiama **Remoting** e può essere utilizzato anche con *http*



Link

<http://www.guisa.org>

<http://www.guisa.org/forums>



GUISA
Gruppo Utenti Italiani
Solution Architect

Dubbi/Domande/Curiosità? 😊



GUISA
Gruppo Utenti Italiani
Solution Architect