



Pattern potenziali di integrazione del mainframe della banca

Massimiliano Bigatti

SUN Certified Enterprise Architect for Java Platform 2 Enterprise Edition Technology



Il case study

- **Integrazione host**
- **Middleware basato su SOAP**
 - XML dentro la risposta
- **Applicazioni di sportello**
 - Erogazione del credito al consumo
 - Erogazione mutui

Pattern discovery

- **Pattern: soluzioni comprovate a problemi ricorrenti**
- **Espressivi: vocabolario comune**
- **Riferimenti**
 - Gof (Design Pattern)
 - Fowler (Patterns of Enterprise Pattern Architecture)
 - SUN (Core J2EE Patterns Catalog)
- **Pattern discovery**
 - Osservazione dei problemi e soluzioni
 - Utilizzo del buon senso

Architettura SOA

- **Evoluzione dei Web Services**
- **Orientata al Business**
- **Tre idee forti:**
 - **Accoppiamento lasco**
 - **Componenti a grana grossa**
 - **Asincronicità**
- **Lo stato dei prodotti**
- **Lo stato del mercato**

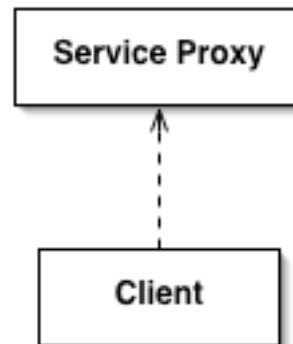
Il catalogo

- **Service Proxy**
- **Proxy with Channel**
- **Service Coordinator**
- **Service Simulator**
- **Configuration Driven Service**

- **Data Logger**
- **Flow Logger**
- **State Logger**
- **Input/Output Validator**

Service Proxy

- Each service is represented by a Service Proxy



Service Proxy - il problema

```
public float getRate( String currency1, String currency2 ) {
    MessageFactory mf = MessageFactory.newInstance();
    SOAPMessage msg = mf.createMessage();
    SOAPPart sp = msg.getSOAPPart();
    SOAPEnvelope env = sp.getEnvelope();
    SOAPHeader hdr = env.getHeader();
    SOAPBody bdy = env.getBody();

    String xsi = "http://www.w3.org/2001/XMLSchema-instance";
    env.addNamespaceDeclaration("xsi", xsi);
    env.addNamespaceDeclaration("xsd", "http://www.w3.org/2001/XMLSchema");
    env.setEncodingStyle("http://schemas.xmlsoap.org/soap/encoding/");

    javax.xml.soap.Name xsiTypeString = env.createName("type", "xsi", xsi);

    SOAPBodyElement gltp = bdy.addBodyElement(
        env.createName("getRate", "ns1", "urn:xmethods-CurrencyExchange")
    );

    ...more JAXM code
    return rate;
}
```

Service Proxy - la soluzione

- Passaggio parametri
- Esecuzione chiamata
- Ottenimento dei risultati

Service Proxy
setParam1() setParam2() call() getResult1() getResult2()

Service Proxy - esempio

- **Esempio:**

ExchangeService
setCountry1() setCountry2() call() getRate()

```
Service service = new ExchangeService();  
service.setCountry1( currency1 );  
service.setCountry2( currency2 );  
service.call();  
System.out.println( service.getRate() );
```

Service Proxy - stile alternativo

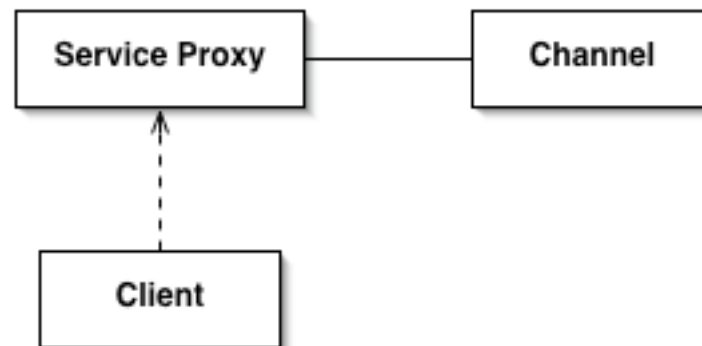
- Metodo funzionale con firma specifica

```
Service service = new ExchangeService();  
System.out.println(  
    service.getRate( currency1, currency2 )  
);
```

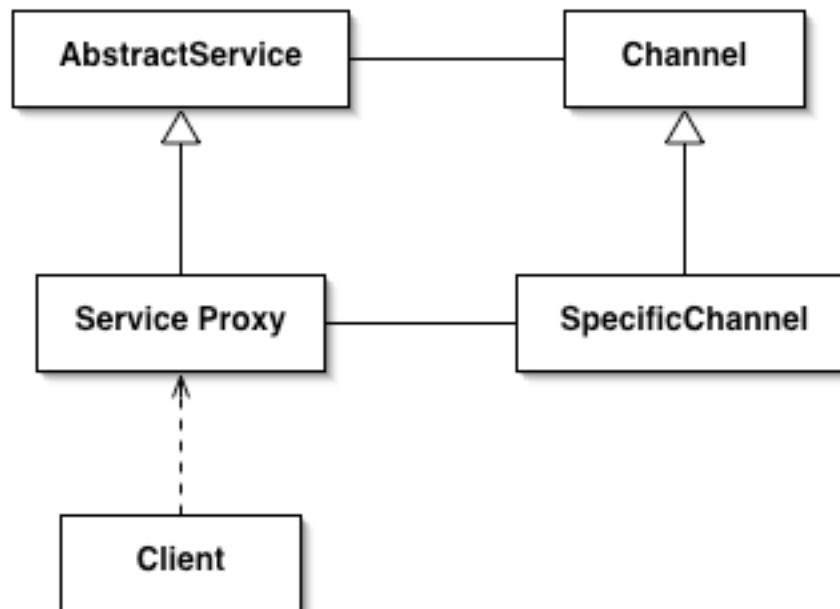
- (p.e. JAX-RPC)

Proxy with Channel

- Each service decouples communication using a channel object



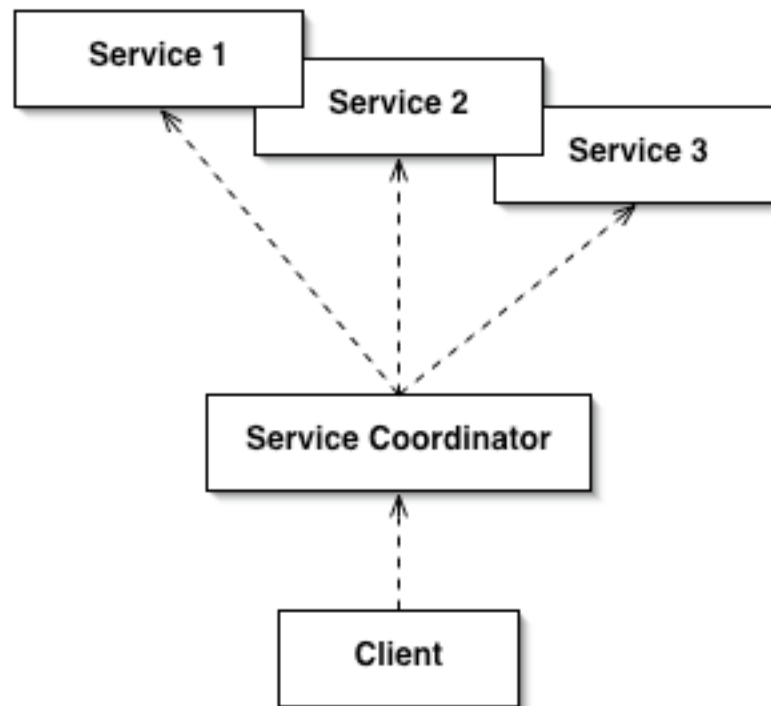
Proxy with Channel - dettaglio e vantaggi



- Supporto ad altri protocolli (SOAP 1.2, XML-RPC, REST)
- Semplificazione del proxy
- Separazione netta tra il livello funzionale e quello tecnico

Service Coordinator

- Interaction between the program and other services is mediated by a coordinator



Service Coordinator - esempio

```
public float exchange( String currencyFrom, String currencyTo,
                      float value ) throws ServiceException {

    lookup.setCurrency( currencyFrom );
    lookup.call();
    String country1 = lookup.getCountry();

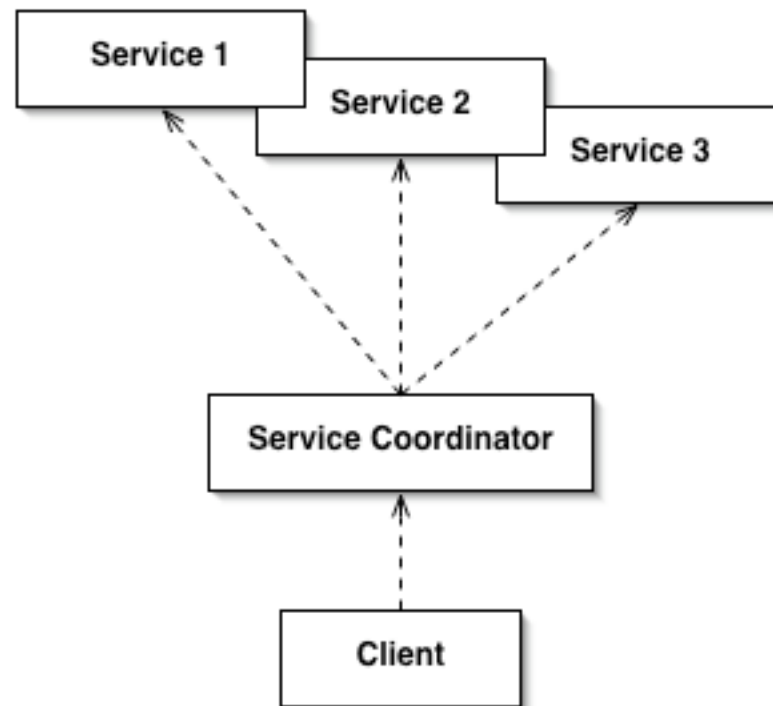
    lookup.setCurrency( currencyTo );
    lookup.call();
    String country2 = lookup.getCountry();

    exchange.setCountry1( country1 );
    exchange.setCountry2( country2 );
    exchange.call();

    return exchange.getRate() * value;
}
```

Service Simulator

- Interaction between the program and other services is mediated by a coordinator



Service Simulator - esempio

```
public class SimulationChannel implements Channel {
    String xml;

    public SimulationChannel( String testFilename )
        throws FileNotFoundException, IOException {

        //load the xml test file in memory
        xml = readFile( testFilename );
    }

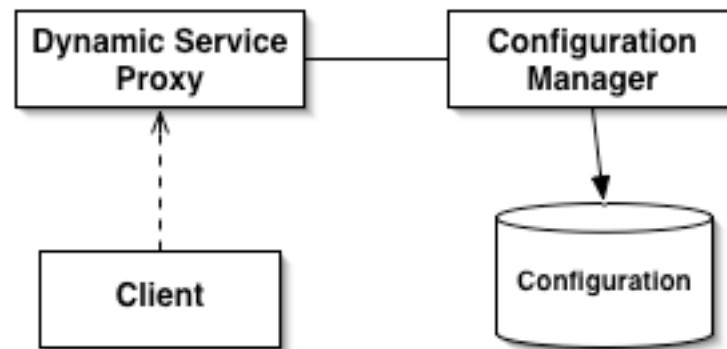
    public void setParameter( String name, String value ) {
    }

    public void call() {
    }

    public String getValue( String expression ) {
        return XPathEvaluator.evaluate( xml, expression );
    }
}
```

Configuration Driven Service

- Input to the service isn't hard coded, but driven by external configuration data



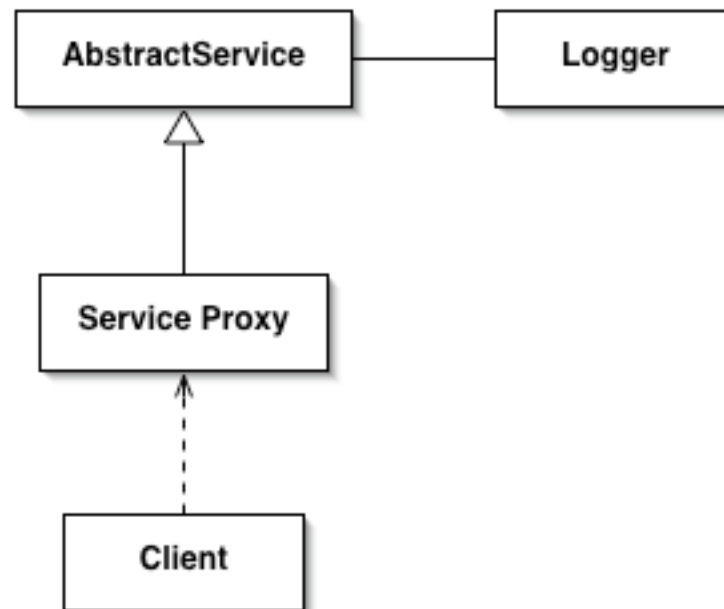
Configuration Driven Service - configurazione

- Una possibile configurazione: per ciascun parametro del messaggio, viene indicata una espressione EL da valutare sul modello applicativo

SOAP parameter	EL like expression
Name	<code>session.currentOrder.requestingCustomer.name</code>
Surname	<code>session.currentOrder.requestingCustomer.surname</code>
Street	<code>session.currentOrder.requestingCustomer.address.street</code>
City	<code>session.currentOrder.requestingCustomer.address.city</code>
Country	<code>session.currentOrder.requestingCustomer.address.country</code>
ssn	<code>session.currentOrder.requestingCustomer.ssn</code>

Data Logger

- Input and output from the service is logged in a database table

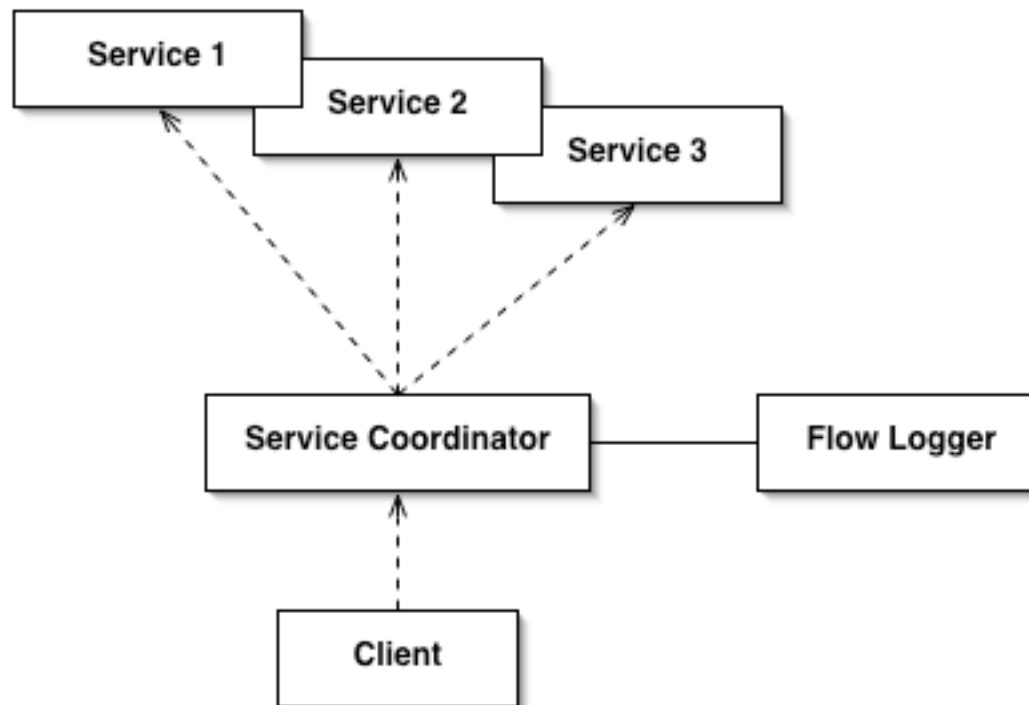


Data Logger - modalità di log

- **File di testo**
 - `System.out.println()`
 - Java 1.4 logging
 - Apache Commons Logging / Jog4j
- **Database (Clob)**
 - Mantiene contigue le informazioni
 - Rende più semplice la formattazione tramite strumenti di debugging
 - Maggiormente scalabile e gestibile

Flow Logger

- Input and output from the services involved in the process is logged in a database table



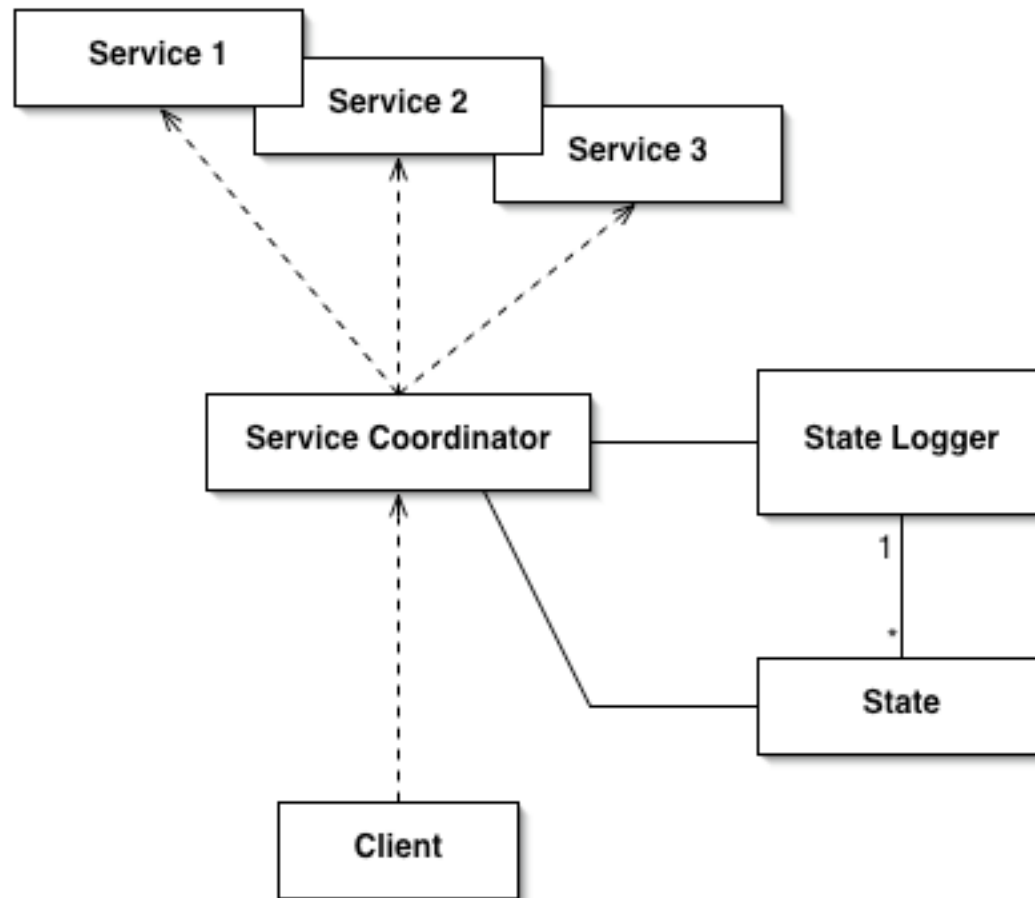
Flow Logger - un flusso di esempio

- Traccia lo stato di esecuzione del processo
- Eventuale rollback
- Mancano standard (ma stanno emergendo: p.e. WS-Transaction)

State	Operation/Service
1	CREATE
2	INIT
3	ADD A
4	ADD B
5	ADD C
6	MODIFY B
7	MODIFY C
8	CLOSE

State Logger

- Stores the state of a transaction to restart in a sequent call from that point

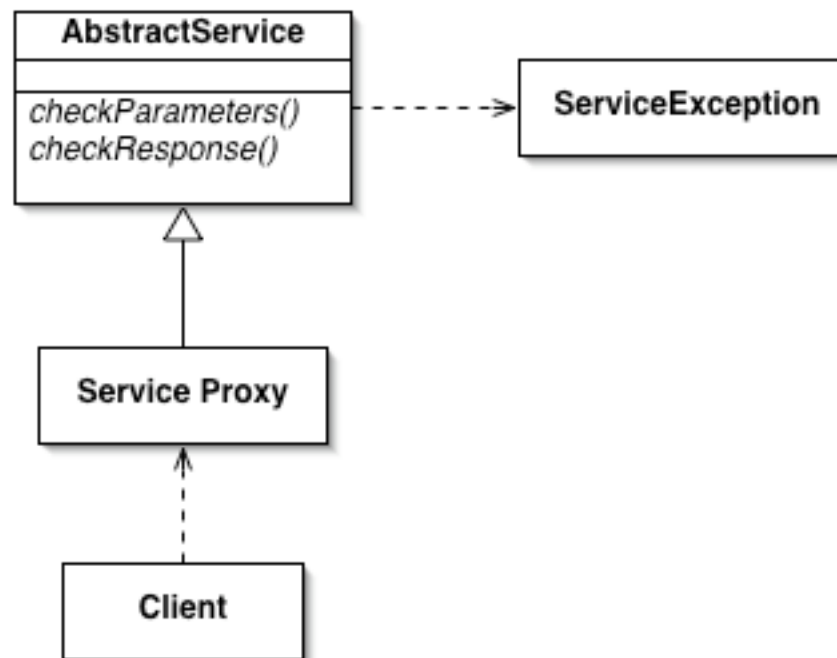


State Logger - esempio

```
if( !flowLogger.wasDone( data.key, CREATE ) ) {  
    CreateService createService = new CreateService();  
  
    createService.setKey( data.key );  
    createService.setType( data.type );  
    createService.setName( data.name );  
    createService.setSurname( data.surname );  
  
    create.call();  
  
    accountId = createService.getAccountId();  
    flowLogger.track( data.key, CREATE );  
}
```

Input/Output Validator

- Validates input and output from the service to reduce garbage data



Input/Output Validator - esempio

```
public void checkParameters() throws ServiceException {
    if( country1 == null || countryList.get( country1 ) == null ) {
        throw new ServiceException(
            "Country1 " + country 1 + " not valid");
    }
    if( country2 == null || countryList.get( country2 ) == null ) {
        throw new ServiceException(
            "Country2 " + country 2 + " not valid");
    }
}

public void checkResponse() throws ServiceException {
    if( rate == null || rate == Float.NaN ||
        rate == Float.NEGATIVE_INFINITY ||
        rate == Float.POSITIVE_INFINITY ||
        rate < 0 ) {
        throw new ServiceException(
            "The service returned an invalid value");
    }
}
```

Conclusioni

- **Framework**
- **In evoluzione**
- **Vocabolario comune**
- **Maggiore flessibilità**
- **Servizi di base condivisi**
- **Esplicitazione delle componenti funzionali**
- **Esplicitazione dei processi di coordinamento**

Domande e risposte

