Service Data Objects in java projects: practical experience

Agenda

- Introduction in SDO
- SDO scenarios and use cases
- Concrete SOA project using SDO: practical experience and lessons learned

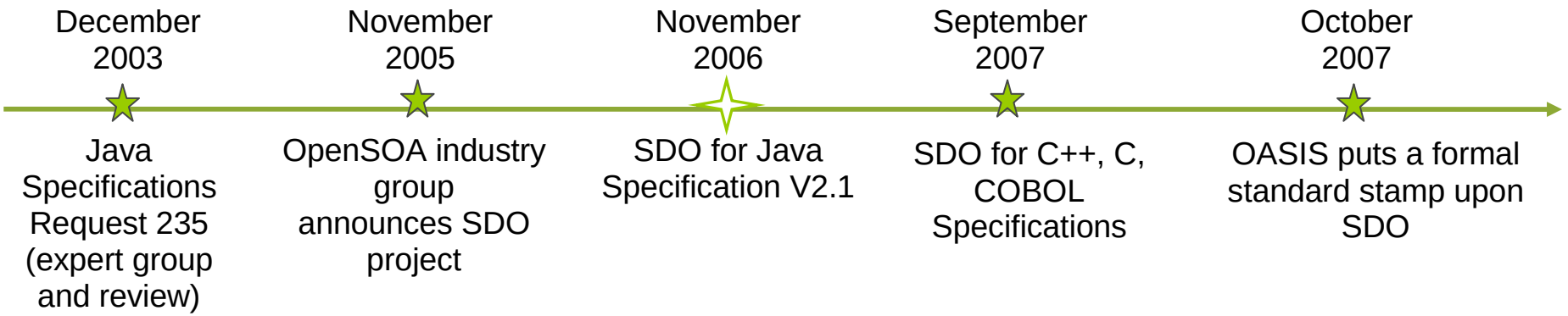
My Background

Andrei Shakirin

- Software architect in SOPERA Team
- 7 years of developing open source SOA Platform
- Long term experience in design and implementation SOA projects using different technologies
- Active commiter of Eclipse SOA Tool Platform project
- Contributor of Eclipse Swordfish Project

Introduction in SDO

SDO History



SDO Industry Group Members: BEA, IBM, Oracle, SAP, Xcalia

Introduction in SDO

Definition

SDO is a framework for Data Application Developing including Architecture and API.

Hmm ... One more data framework? Why??? What is the added value of SDO?

- **Unifying** data programming across different data formats and data source types (RDBs, EJBs, Web Services, Messaging Providers, JCA, POJOs)

- **Incorporating** a number of application patterns and best practices (change history, disconnected programming model, compositor)

- **Enabling** application, tools and frameworks to more easily query, view, update, bind, transfer data and keep it consistent

Introduction in SDO

SDO and other Data Application Frameworks

SDO and JAXB	Similar in sense of model Java-XML binding, but one XML is not only data source for SDO. SDO provides uniform access for various types of data. There is a bridge between SDO and JAXB models.
SDO and JPA	Similar in sense of simplifying java data programming and abstract from specific technology. Both frameworks provide a context and manage attached data objects. JPA is concentrated on data persistency in RDBs, SDO is more general and represents data that can flow between any application tier.
SDO and EMF	Very similar in purpose, SDO can be interpreted as a thin layer (facade) over EMF. IBM reference SDO implementation is EMF based.
SDO and SCA	Complementary specs. SCA defines assembly of service components into business solution. SDO concentrates on the data model representation. Could be used together and separated.

Introduction in SDO

Key Concept

- **DataObject**
- **DataGraph**
- **Data Access Service**

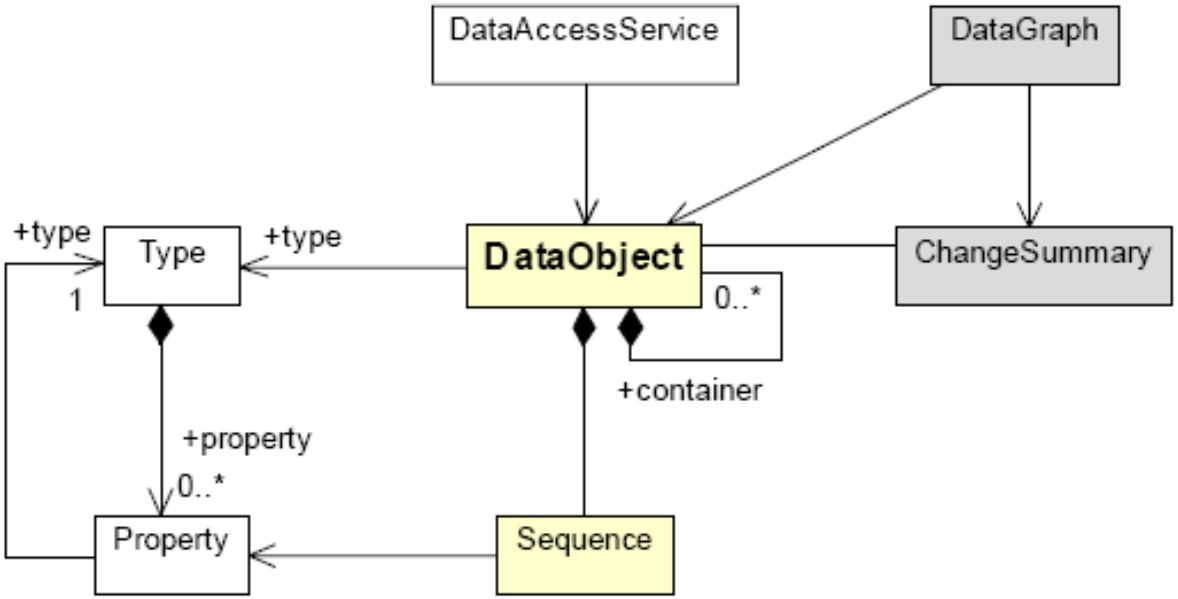
Introduction in SDO

DataObject

Represents a business data.

Holds a set of named properties, each of which contains either a simple data-type value or a reference to another Data Object (simple, containment, read-only, multi-valued, open).

The Data Object API provides a dynamic and static data API for manipulating these properties.

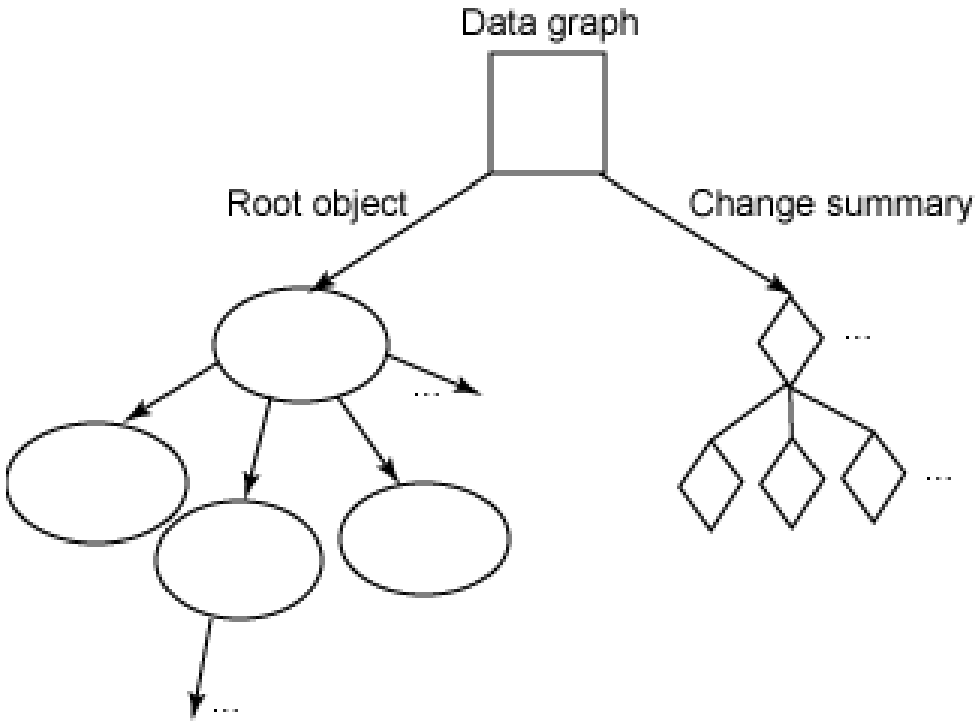


Introduction in SDO

DataGraph

Envelope for Data Objects, and it is the normal unit of transport between components. DataGraph is a closed tree (cannot have a circles, exactly one parent)

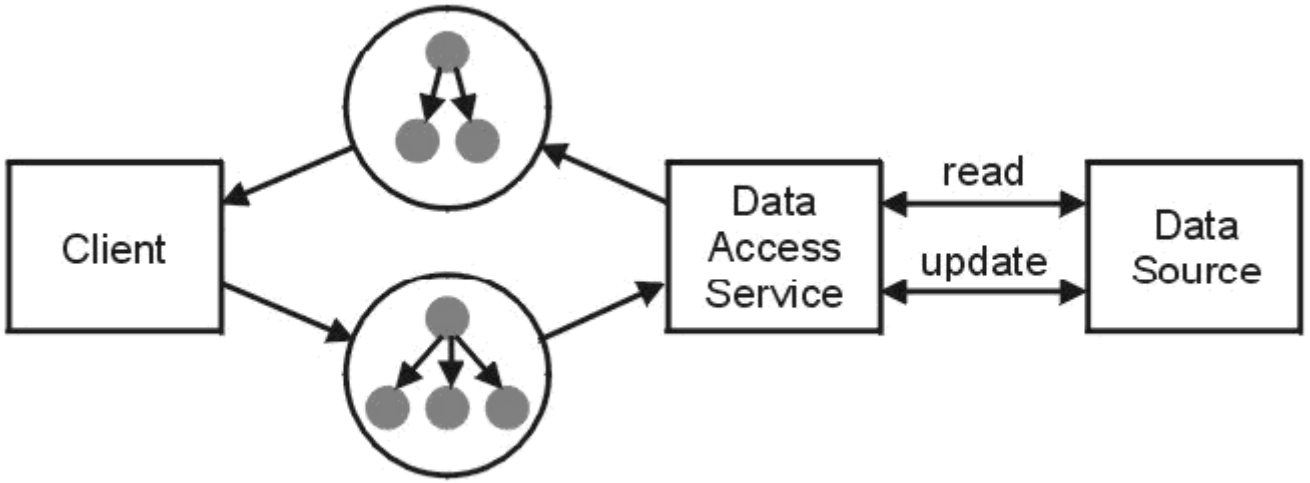
Data graphs can track changes made to the graph of Data Objects.



Introduction in SDO

Data Access Service

Populates data graphs from data sources and commits changes to data graphs back. Uses disconnected data architecture



Introduction in SDO

Additional Aspects of Specification

- Dynamic and Static APIs
- Change Summary (including XML representation)
- Validation, Contrains (via metadata and extensions) and Relationship Integrity: cardinality, closure, ownership semantic and inverses
- Helpers API
- Sequences, open properties, containment and non-containment references

Out of scope:

- DAS specification

Introduction in SDO

Implementations

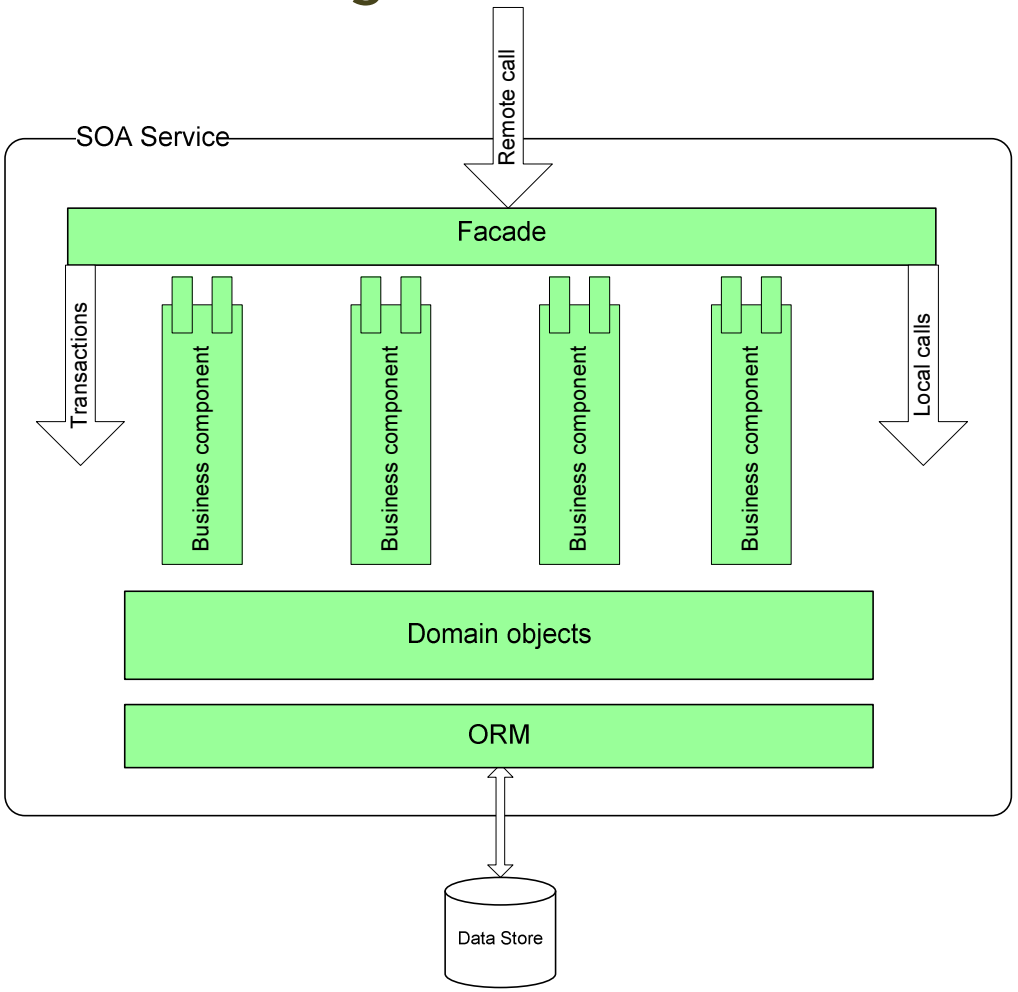
- EclipseLink
- Apache Tuscany (integrated with Apache CXF)
- SCA and SDO for PHP (PECL extension)
- Rogue Wave Software (Hydro SDO)
- Xcalia
- IBM WebSphere (Ecore based)

Introduction in SDO

Live Demo: Hello World SDO Project

SDO scenarios

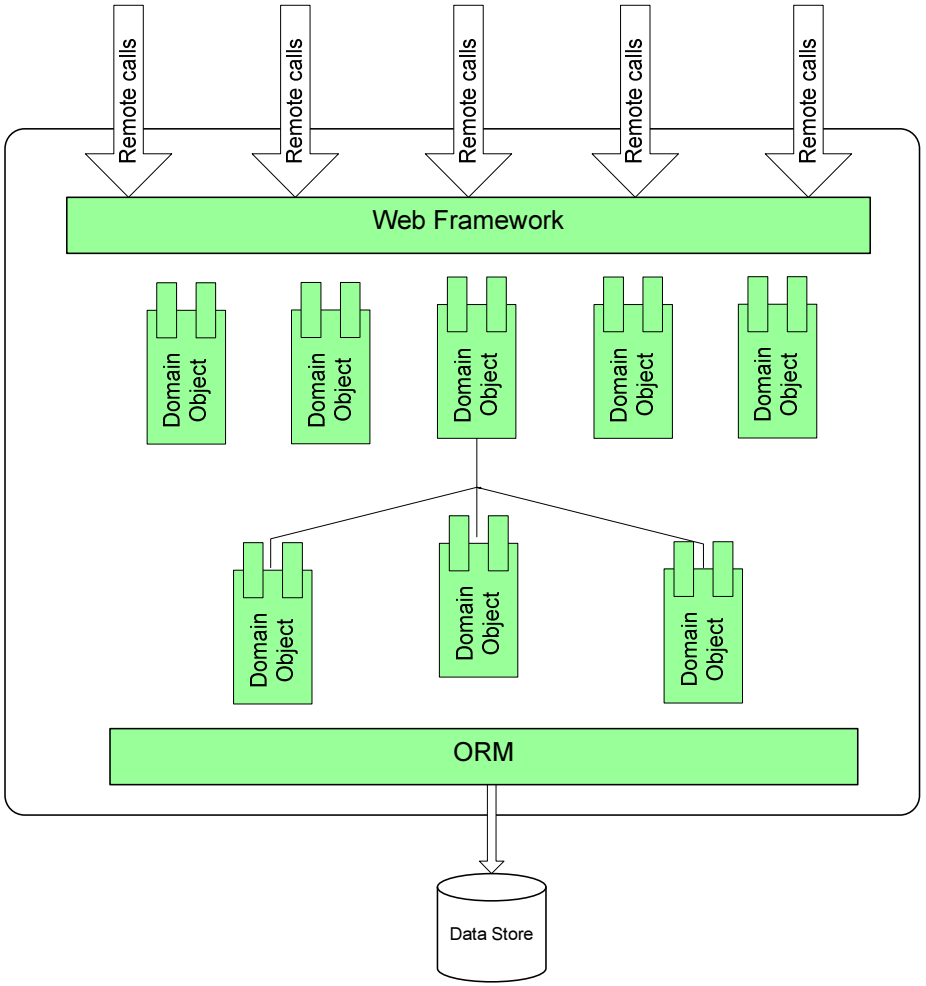
SOA design



- Operation centric
- Decoupling of client from business components
- Anemic domain objects
- Normally stateless
- Good match to SOAP stack

SDO scenarios

Resource-oriented design



- Resource centric
- Domain objects are exposed to the client via URLs
- Stateless
- Domain objects have multiple representations
- Domain objects containing business logic
- More compatible with REST stack

SDO scenarios

Stateless Design challenges

- Heavily interconnected objects become hard to merge on the server side
- It is not always possible to merge graph automatically and even consistently
- Because the transport of whole graph is too expensive or not desired, service facade is extended with additional methods to manage each particular part of graph. The server likes to know in advance which subgraph should be proceed (causes „God facades“: **`getCustomerWithAddress() / setCustomerWithAddress()` which are redundant and very difficult to maintain**).

SDO approach is a possible solution for these challenges!

SOA project using SDO

Greenpeace UK Project

Requirements: Greenpeace Web Portal should provide following functionality:

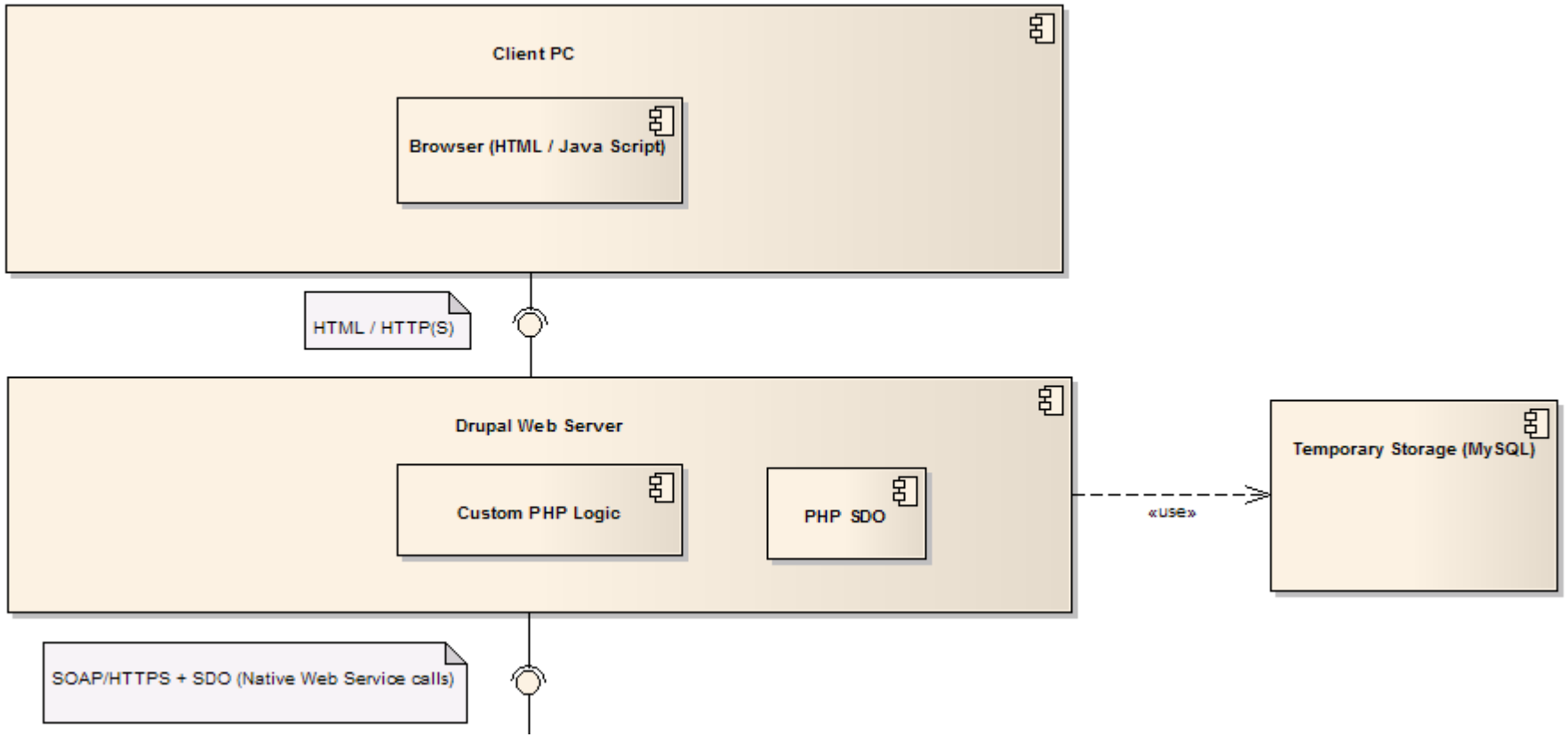
- Self Registration (CRUD for visitors and supporters including email confirmation, control of web accounts, managing and validation of personal data)
- Payment (single donations and recurring payments)
- E-mail news (notifications about actual Greenpeace events and actions)
- Campaigning service

Project info:

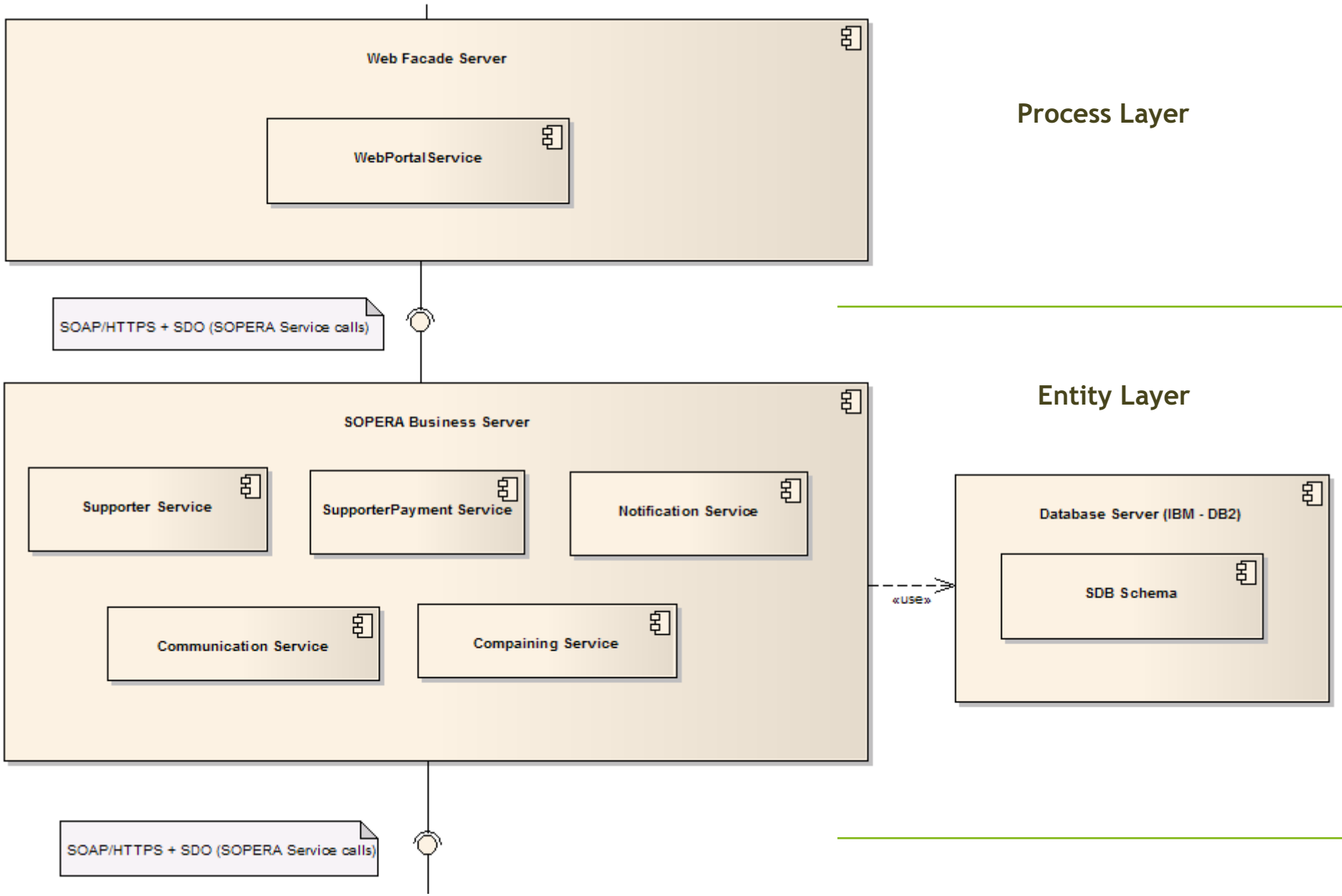
- Terms: 02.09 – 08.09
- Team: 3 persons
- Platform, technologies: SOPERA ESB; EclipseLink SDO/JPA; DB2
- Methodology: SCRUM

SOA project using SDO

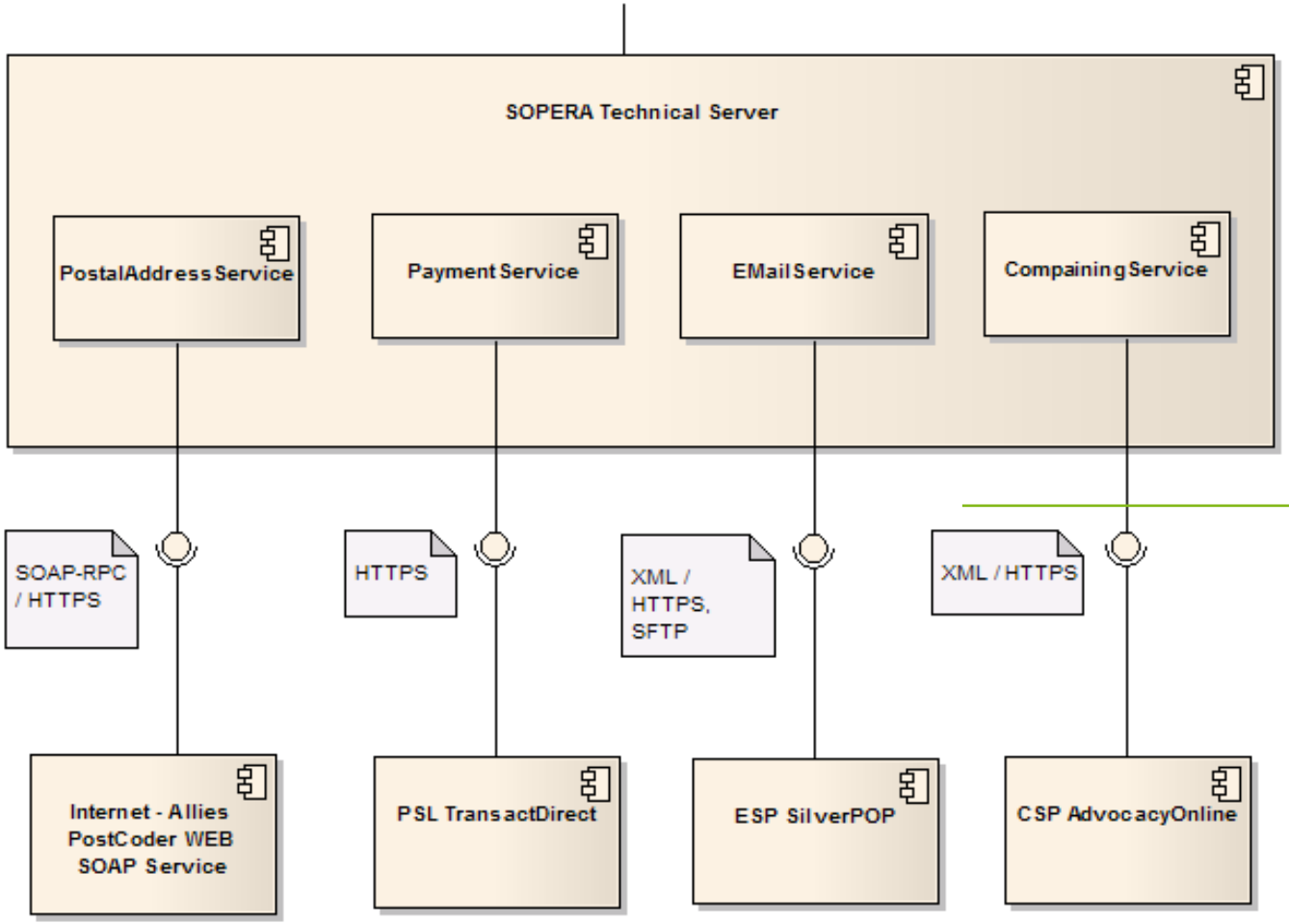
Architecture of Greenpeace UK SOA project



SOA project using SDO



SOA project using SDO

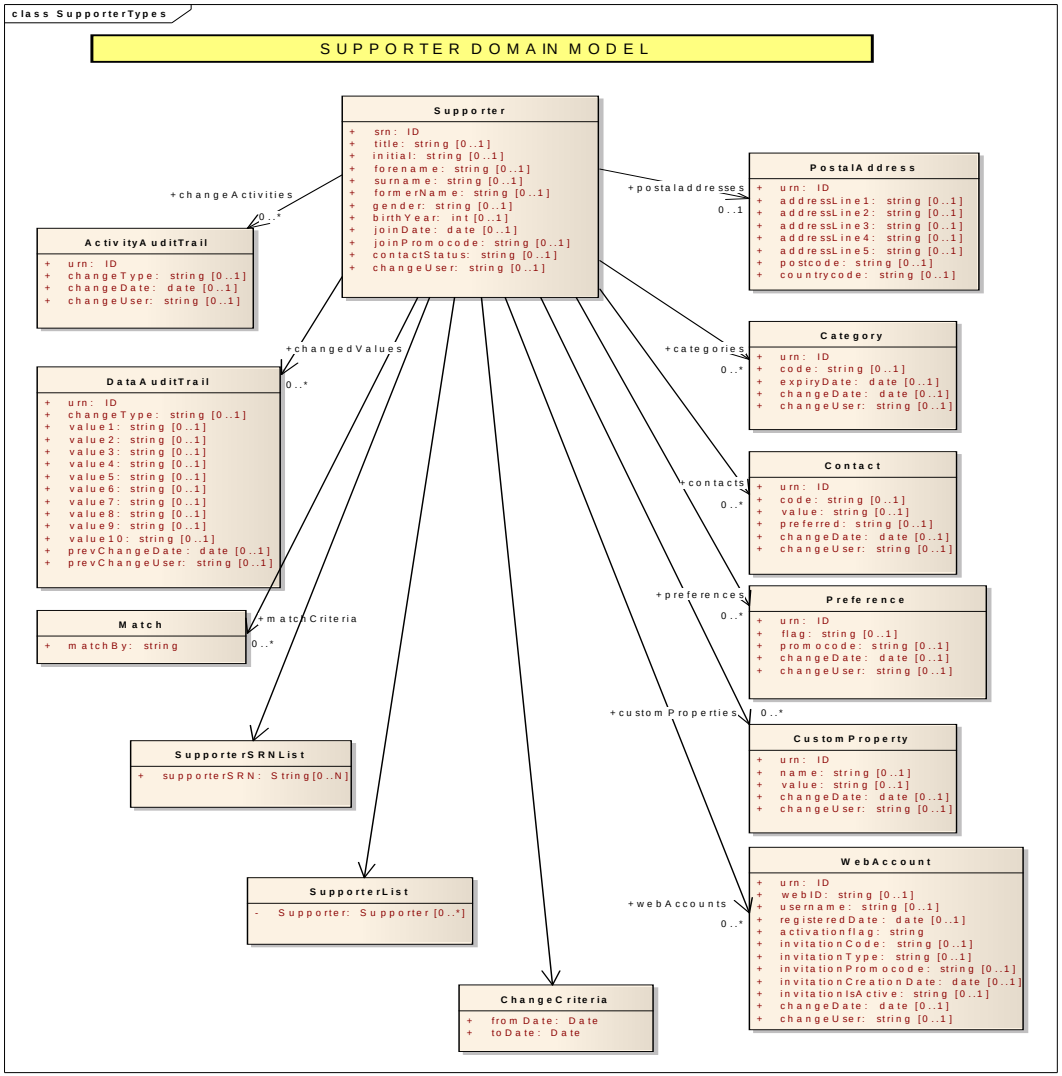


Utility Layer

SOA project using SDO

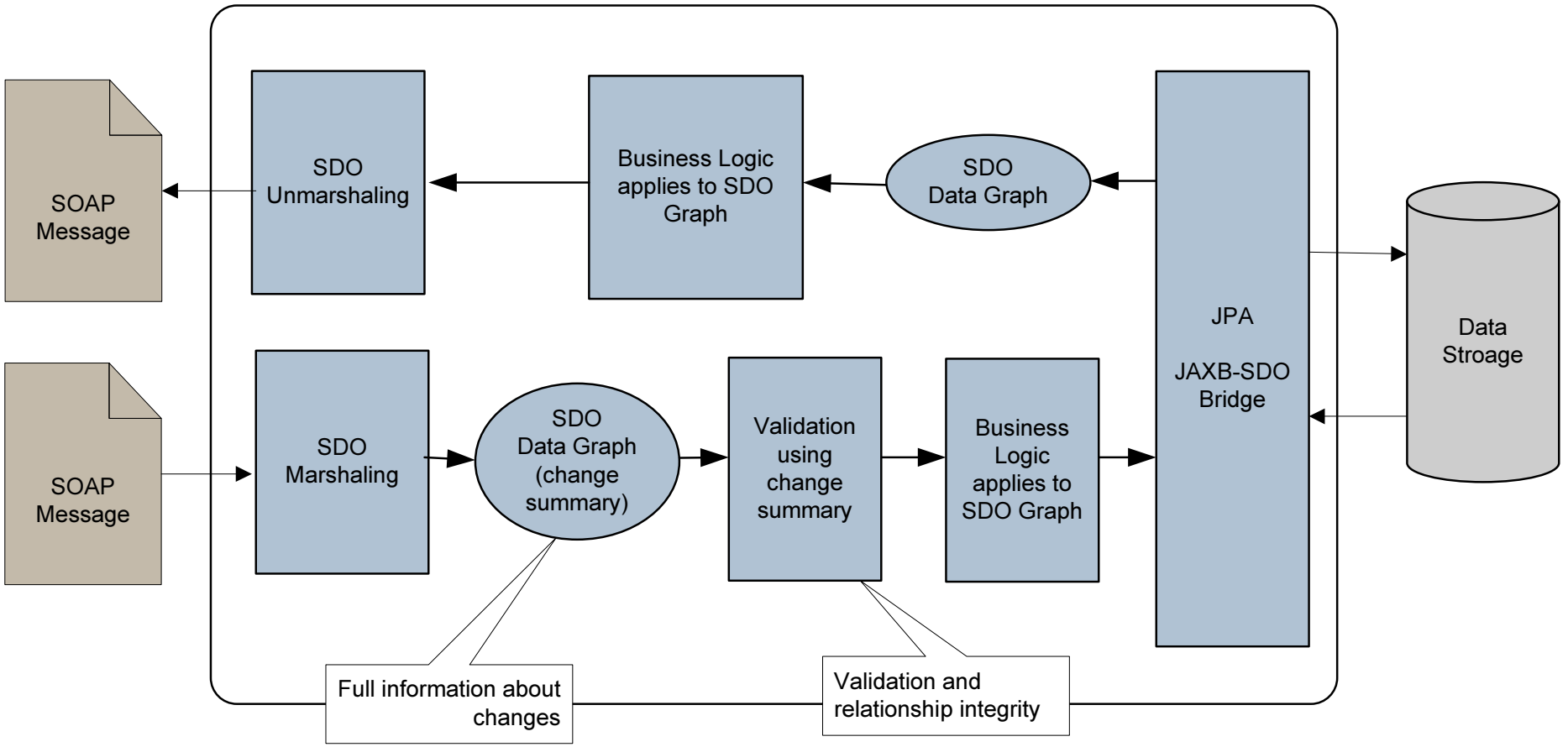
Supporter Domain Object:

complex enough?



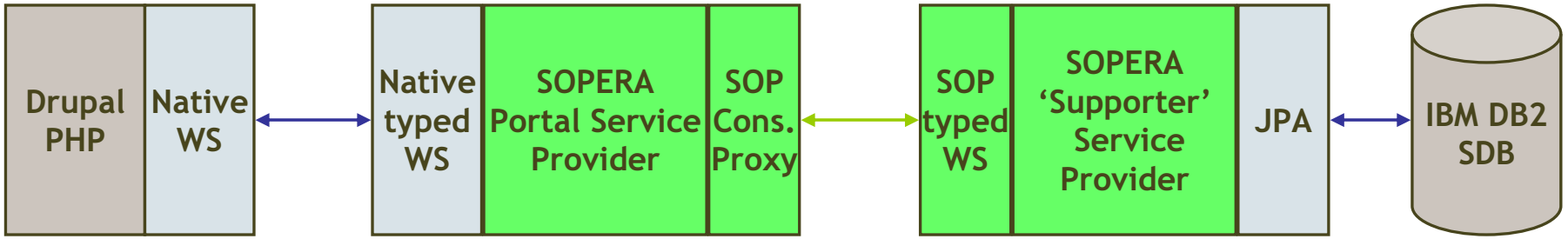
SOA project using SDO

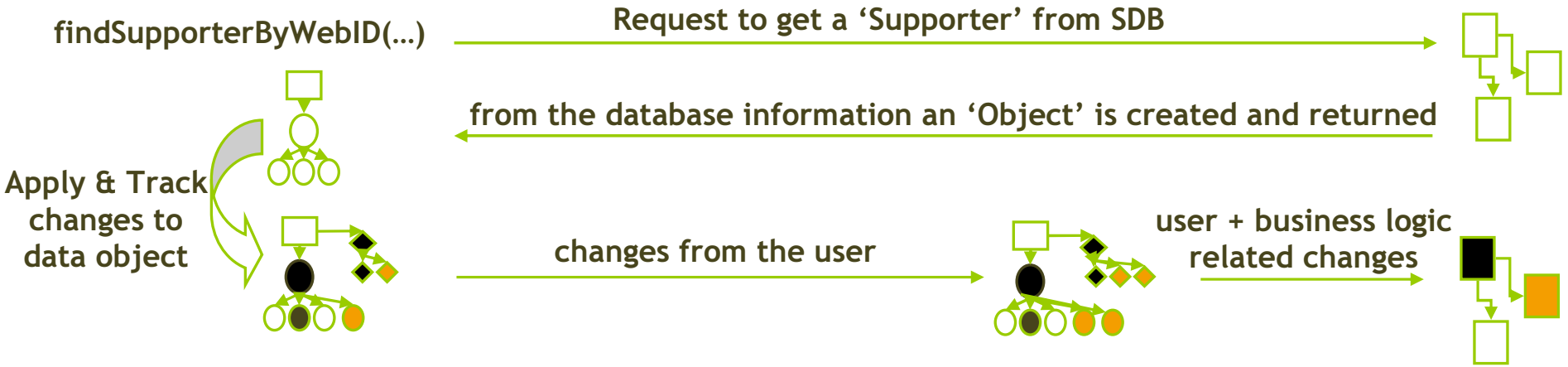
Inside the Service



SOA project using SDO

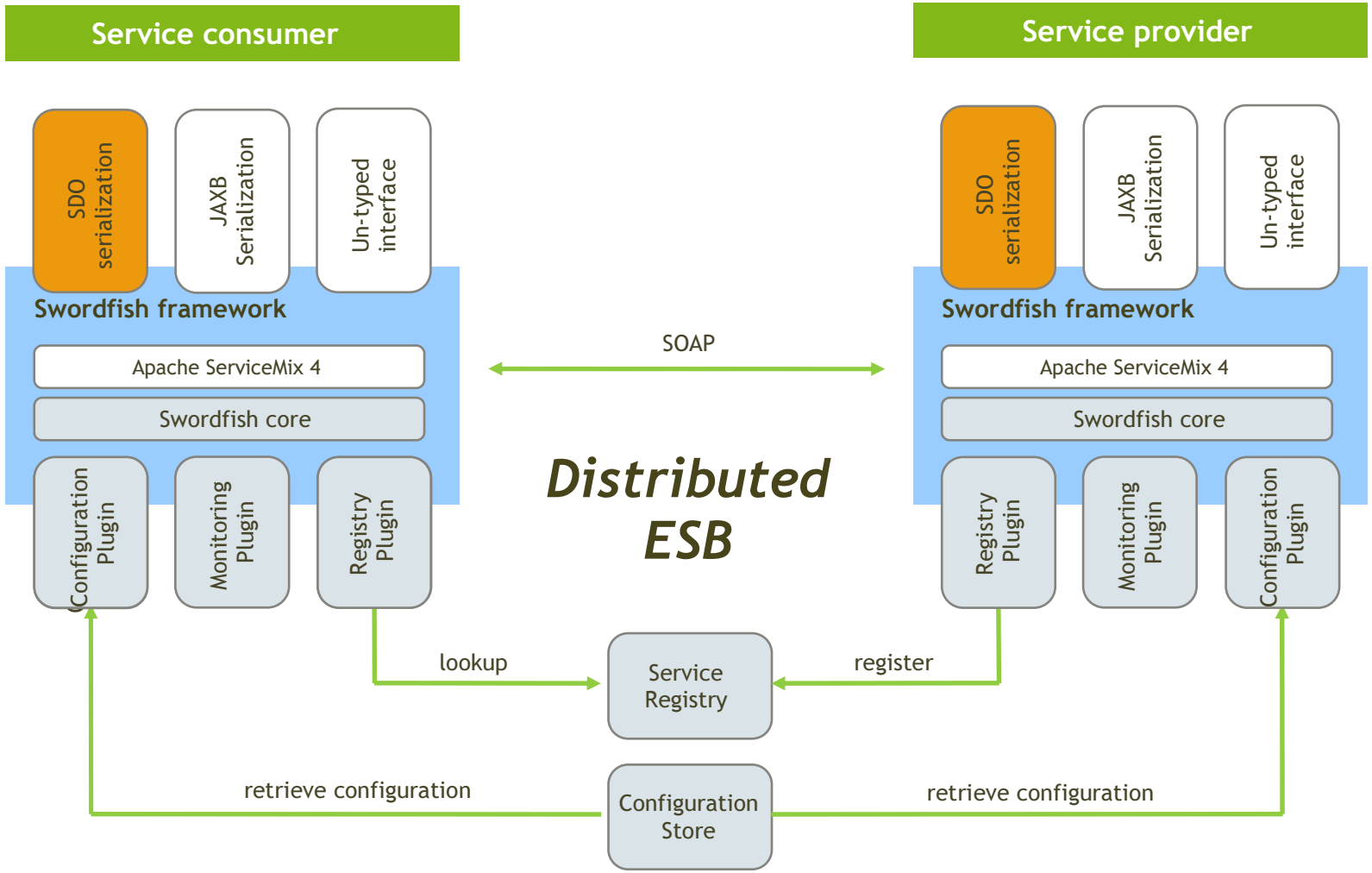
Inside the Service





SOA project using SDO

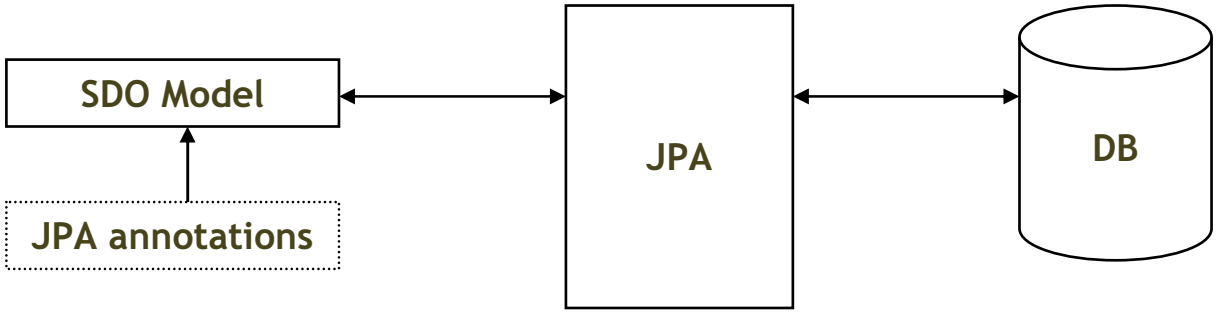
Integration SDO and SOA ESB



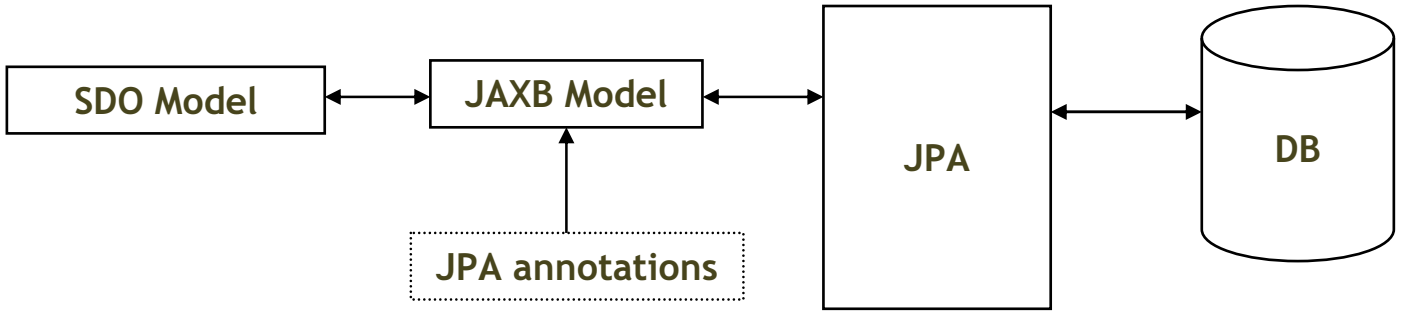
SOA project using SDO

SDO and JPA: two approaches

- **Persist SDO model directly**

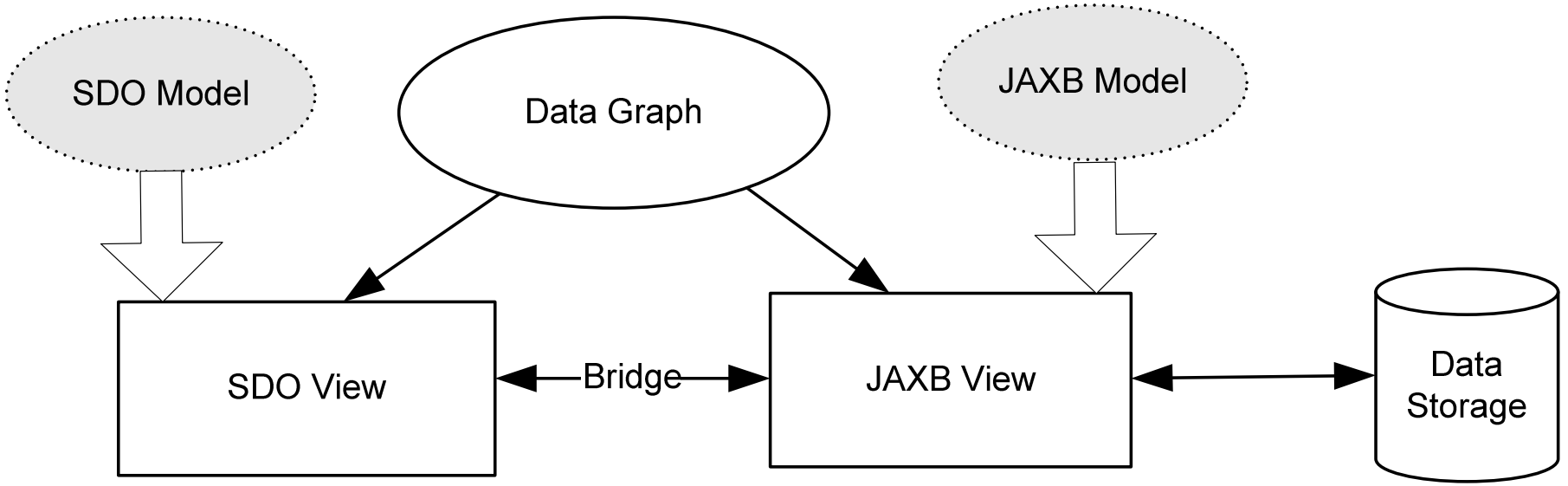


- **Using JAXB bridge for persistency**



SOA project using SDO

SDO and JPA: JAXB Bridge



SOA project using SDO

SDO and JPA: JAXB bridge

```
// 1. = Define an JAXB context for the POJOs

JAXBContext jaxbContext =
     JAXBContext.newInstance("com.example.employee");


// 2. = Instantiate the JAXBHelperContext based on JAXB context

JAXBHelperContext jaxbHelperContext = new JAXBHelperContext(jaxbContext);


// 3. = Inform SDO context about XSD metadata

jaxbHelperContext.getXSDHelper().define(this.getResourceAsStream(“/employee.xsd“));
```

SOA project using SDO

SDO and JPA: SDO->JAXB

```
// 1. Create SDO object
DataObject employeeSDO =
     jaxbHelperContext.getDataFactory.create(”http://www.sopera.com”,
     ”Employee”);


// 2. Process SDO Object (JAXB object is still attached to EM)
employeeSDO.set(“id“, “65374“);
employeeSDO.set(“first-name“, “Jane“);
employeeSDO.set(“second-name“, “Doe“);


// 3. Convert SDO to JAXB
Employee employee = jaxbHelperContext.unwrap(employeeSDO);


// 4. Persist JAXB
em.persist(employee);
```

SOA project using SDO

JAXB Bridge: JAXB->SDO

```
// 1. Get JAXB employee object from DB using entity manager
Employee employee = em.find(Employee.class, id);

// 2. Wrap it to SDO object
DataObject employeeSDO = jaxbHelperContext.wrap(employee);

// 3. Process SDO Object (JAXB object is still attached to EM)
String firstName = employeeSDO.get(“first-name“);
DataObject address = employeeSDO.get(“address“);
employeeSDO.set(“address/street“, “New Street“)

// 4. Update JAXB object in EM
em.update(employee);
```

Conclusion and lessons learned

- SDO is a possible solution for stateless SOA Design challenges, especially in case of processing complex Data Graph
- Change Summary is very comfortable mechanism to validate, process and propagate changes between service consumer and provider
- It is quite easy to support SDO in ESBs with pluggable marshalling layer (supported by SOPERA, Apache CXF out of the box)
- Persistency for SDO Graph can be proceeded using JAXB bridge. It allows to use all popular JPA implementations
- As open source implementations I would recommend to evaluate EclipseLink and Apache Tuscany

Useful links:

- **SDO specification:**

http://www.osoa.org/display/Main/Service+Data+Objects+Specifications

- **EclipseLink SDO project:**

http://www.eclipse.org/eclipselink/sdo.php

- **EclipseLink samples:**

http://wiki.eclipse.org/EclipseLink/Examples/SDO

http://wiki.eclipse.org/EclipseLink/Examples/SDO/JAXB