JSR 235 and Service Data Objects (SDO)

Stephen Brodsky
sbrodsky@ibm.com
IBM, Silicon Valley Lab

Michael Carey
mcarey@bea.com
BEA Systems, Inc.

TS-3676

Goal of This Talk

What you will learn

Learn how Service Data Objects will simplify and unify your SOA data access architecture and code

Co-Presenters

- Michael Carey–BEA Systems
- Stephen Brodsky–IBM
- Blaise Doughan–Oracle
- Henning Blohm–SAP
- Eric Samson–Xcalia

Agenda

SDO—The Big Picture

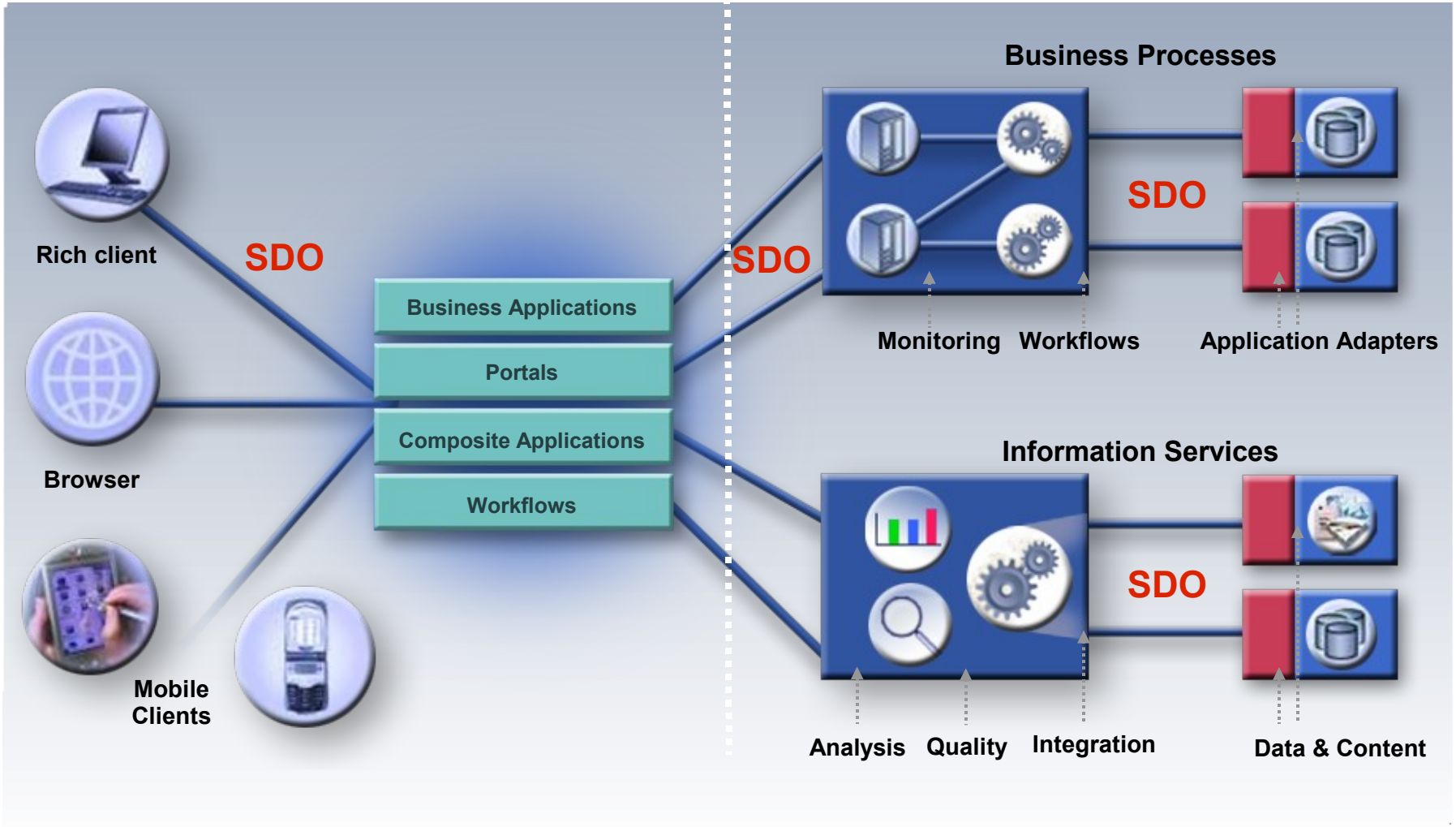
SDO—Key Concepts and Core APIs

SDO in Action

Summary

Information as a Service

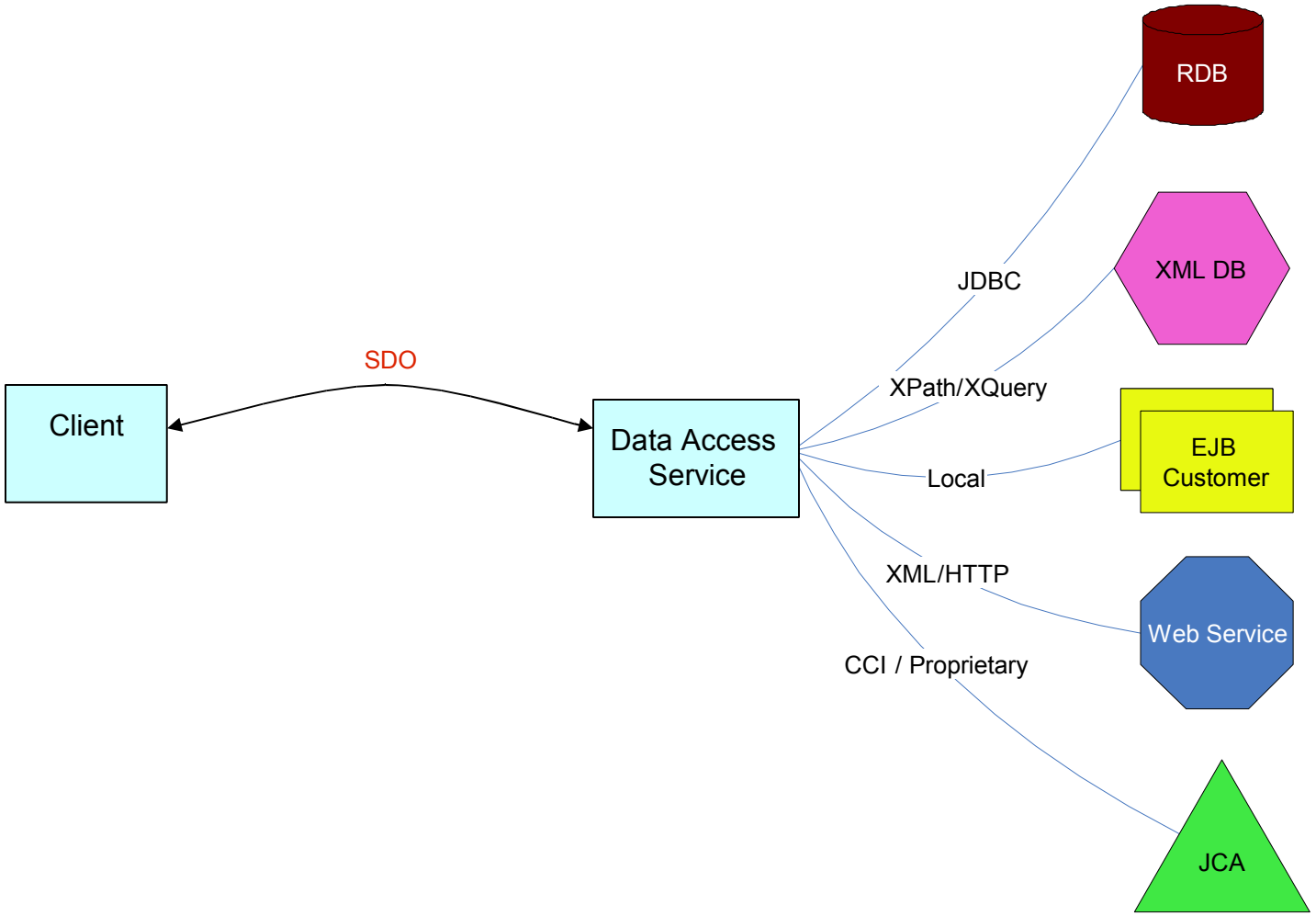
Virtualizing the IT Infrastructure



Programming Data Services—SDO

- Object/Java™ technology API that spans types of data to provide
  - Object(value)-based read-modify-write
  - Complex data structures (not just rows)
  - Disconnected access pattern
  - Optimistic concurrency control model
  - Tools for data binding
- Industry Support
  - Joint specification: BEA, IBM, Oracle, SAP, Sybase and Xcalia
  - Open source implementations (Apache, Eclipse)
  - Products

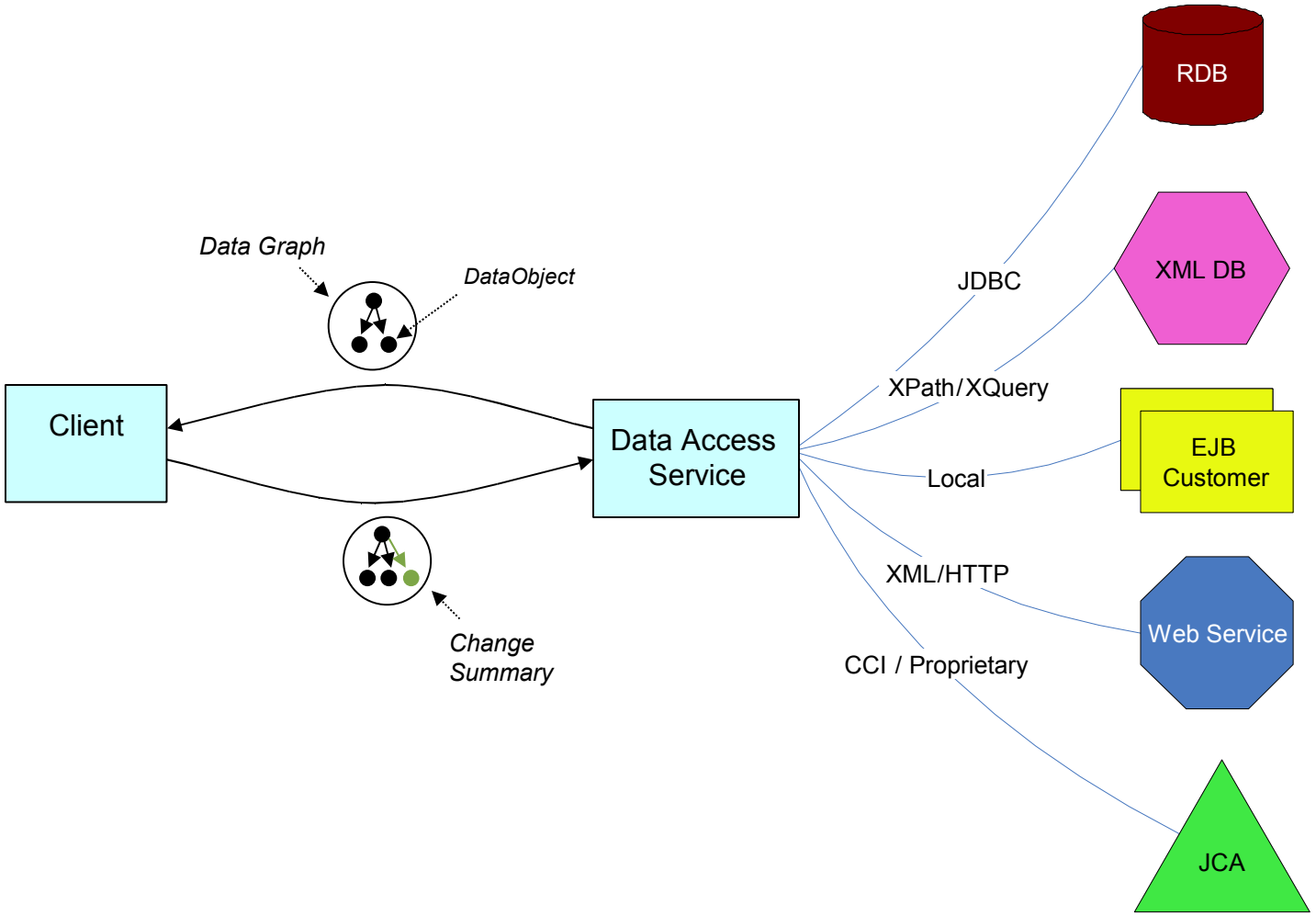
Heterogeneous Data Access



Usage Patterns

- Web apps are semi-connected, optimistic-concurrency based apps
  - Open DB connection
  - Retrieve data from DB
  - Close DB connection
  - Generate HTML;…
  - Receive response (e.g., HTML form post)
  - Open DB connection
  - Perform update to DB
  - Close DB connection
- Standard patterns for optimistic collision detection, pagination, sorting, etc.

SDO Data Transfer Objects (DTOs)



Service Data Objects—Summary of Goals

- Unified and consistent data access to heterogeneous data sources
  - Simplified programming model for the application programmer
  - Enable tools and frameworks to work consistently across heterogeneous data sources
- Robust client programming model for several Java EE platform best practice application patterns
  - Disconnected client programming model
  - Custom data access layers based on common design patterns
- First class support for XML Schema, XML InfoSet, and XML data sources
  - XML/Java technology bindings
  - Java API for XML-based RPC (JAX-RPC) objects

Agenda

SDO—The Big Picture

SDO—Key Concepts and Core APIs

SDO in Action

Summary

SDO Key Features

- Generated Data API
- Dynamic Data API
- Rich Data Objects
- XML and XML Schema integration
- XPath Navigation through graphs of data
- Change Summary
- Metadata
- Validation and Constraints
- Relationship integrity

SDO Components

- Generated data API: POJO beans
- Dynamic data API: DataObject
- Change summary API: ChangeSummary
- Introspection API: Type and Property
- XML-based serialization for transferring data sets on the wire
  - Can conform to pre-defined XML Schema
  - Can generate XML Schema

DataObject

- Composed of properties
- Single and many-valued properties
- Properties accessed and modified by name, offset, Property, XPath
- Can contain other DataObjects as properties
- Reverse link to containing DataGraph

Generated Data API Example

```
public interface Person {
  String getName();
  void setName(String name);
  int getPostalCode();
  void setPostalCode(int code);
}
```

```
Person p = (Person) dataFactory.create(Person);
p.setName("John");
p.setPostalCode(94133);
System.out.println(p.getName());
```

Dynamic Data API Example

```
<complexType name="Person">
  <attribute name="name" type="string"/>
  <attribute name="postalCode" type="int"/>
</complexType>
```

```
DataObject o = dataFactory.create(tns, "Person");
o.set("name", "John");
o.set("postalCode", 94133);
System.out.println(o.get("name"));
```

DataObject

- get(Property)
- set(Property)

- Properties by String, int, Property, XPath
  - get("address")
  - get(1)
  - get(address)
  - get("address/zip")

- isSet(Property)
- unset(Property)

- create(Property)
- delete()

DataObject—Typed Accessors

- getXXX(property). XXX is
  - primitives: int, float, boolean, byte[],…
  - String
  - BigDecimal, BigInteger
  - Date
  - List for multi-valued properties
  - Converts between primitives and Objects
  - Converts between data types
    - getInt("width") of 5.123 returns 5

Update SDOs

```
DataObject customer1 =
   customers.getDataObject("customer[1]");
customer1.setString("firstName", "Kevin");
```

- ChangeSummary updated to mark object changed with old value of "Adam"

```
<customers xmlns="http://customers.com">
   <customer SN="1" firstName="Kevin" />
   <customer SN="2" firstName="Baker" />
</customers>
```

Example 1—Accessing DataObjects

```
// Get an employee using an SDO xpath expression
// starting from the company
DataObject employee =
company.getDataObject("departments[number=123]/employees[SN=0002]");

// Or, an SDO xpath expression can find the employee
// based on positions in lists:
DataObject employee =
company.getDataObject("departments.0/employees.1");

// Or, use the API to go step by step to find the employee
// Get the list of departments starting from the company
List departments = company.getList("departments");
// Get the department at index 0 on the list
DataObject department = (DataObject) departments.get(0);
// Get the list of employees for the department
List employees = department.getList("employees");
// Get the employee at index 1 on the list
DataObject employeeFromList = (DataObject) employees.get(1);
```

Example 2—Updating DataObjects

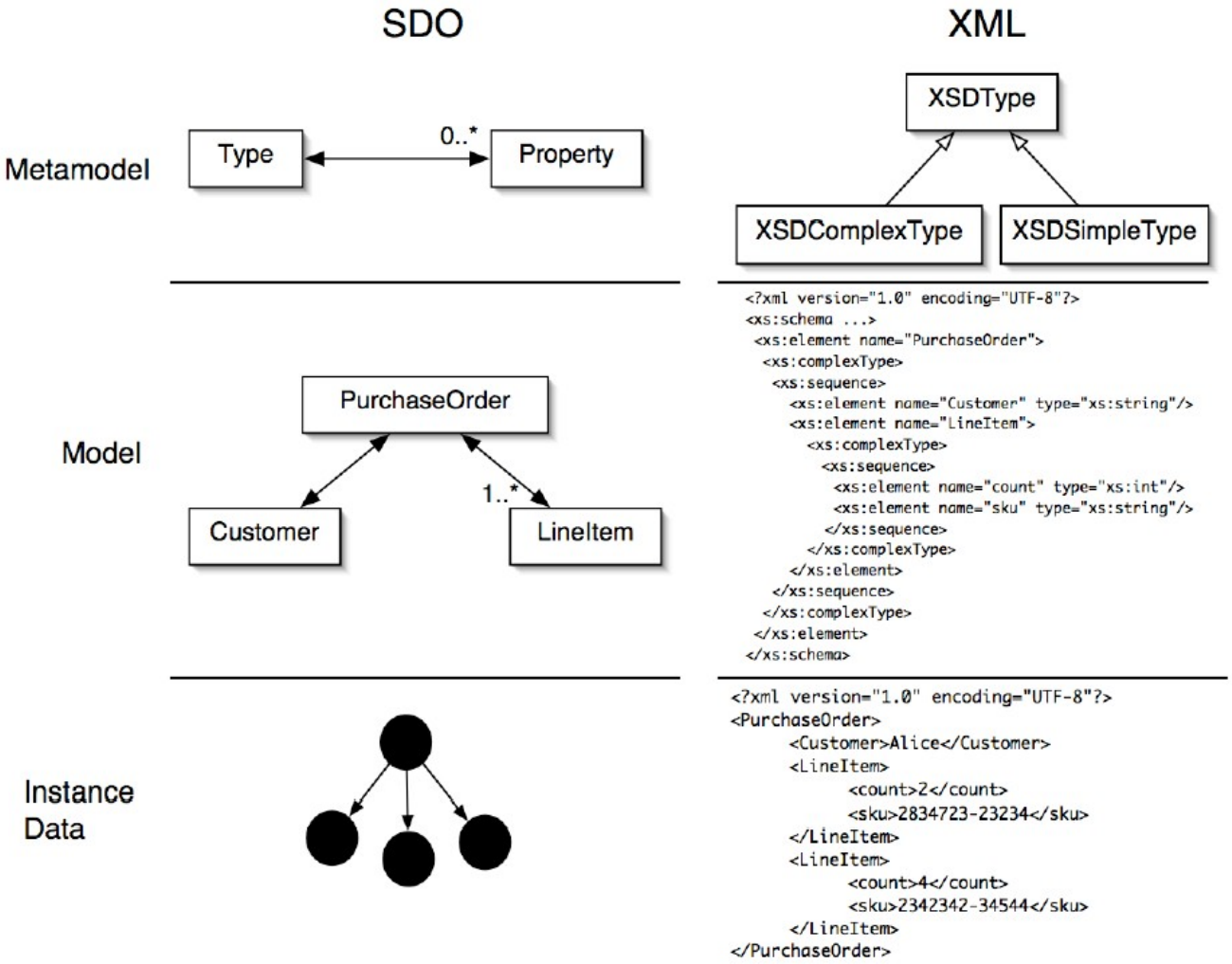
```
// Set the company name
company.setString("name", "ACME");


// create a new employee
DataObject newEmployee = department.createDataObject("employees");
newEmployee.set("name", "Al Smith");
newEmployee.set("SN", "0004");
newEmployee.setBoolean("manager", true);

// Set employeeOfTheMonth to be the new employee
company.set("employeeOfTheMonth", newEmployee);
```

```
<company name="ACME" employeeOfTheMonth="0004">
  <departments name="Advanced Technologies" location="NY" number="123">
    <employees name="John Jones" SN="0001"/>
    <employees name="Jane Doe" SN="0003"/>
    <employees name="Al Smith" SN="0004" manager="true"/>
  </departments>
</company>
```

Instance, Model, and Metamodel



SDO Meta—Model

- SDO provides a simple, universal meta-model
  - Used across JavaBeans™ architecture, XML, or any data source
  - Useful for tools and IDEs (Model in MVC)
- Meta-data Classes
  - “Type”
    - Has name, URI, instance class, and properties
  - “Property”
    - Has name, type, default value, numeric index within Type

Example—SDO Metadata

```
DataObject obj = …;
Type type = obj.getType();
Collection c = type.getProperties();
Iterator i = c.iterator();
while (i.hasNext()) {
   Property prop = (Property) i.next();
   System.out.println(prop.getName());
}
```

XML/SDO Mapping

SDO: 2 objects
DOM: 14 objects

XML

PurchaseOrderType

SDO

```
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>PA</state>
    <zip>90952</zip>
  </shipTo>
  ...
</purchaseOrder>
```

Values

Properties

orderDate=1999-10-20

shipTo

country= US
name= Alice Smith
street= 123 Maple Street
city= Mill Valley
state= PA
zip= 90952

USAddress Type

XML/XSD Integration

- Direct correspondence between XML and DataObjects
- XMLHelper
  - Load and save DataObjects to XML streams
- XSD mapping to and from SDO
- XSDHelper
  - Get XML specific information–isElement, isMixed, local name, appinfo
  - Define Types and Properties from XSDs
    - Annotations or XSLT for mapping control
  - Generate XSDs from Types and Properties

XSD Mapping <–> SDO

XML Schema Concept	SDO Concept	Java Technology Concept
Schema	URI for Types	Package
Simple Type	Type, dataType=true	int, String, BigDecimal, etc.
Complex Type	Type, dataType=false	Interface
Attribute	Property	getX( ), setX( )
Element	Property	getX( ), setX( )

Agenda

SDO—The Big Picture

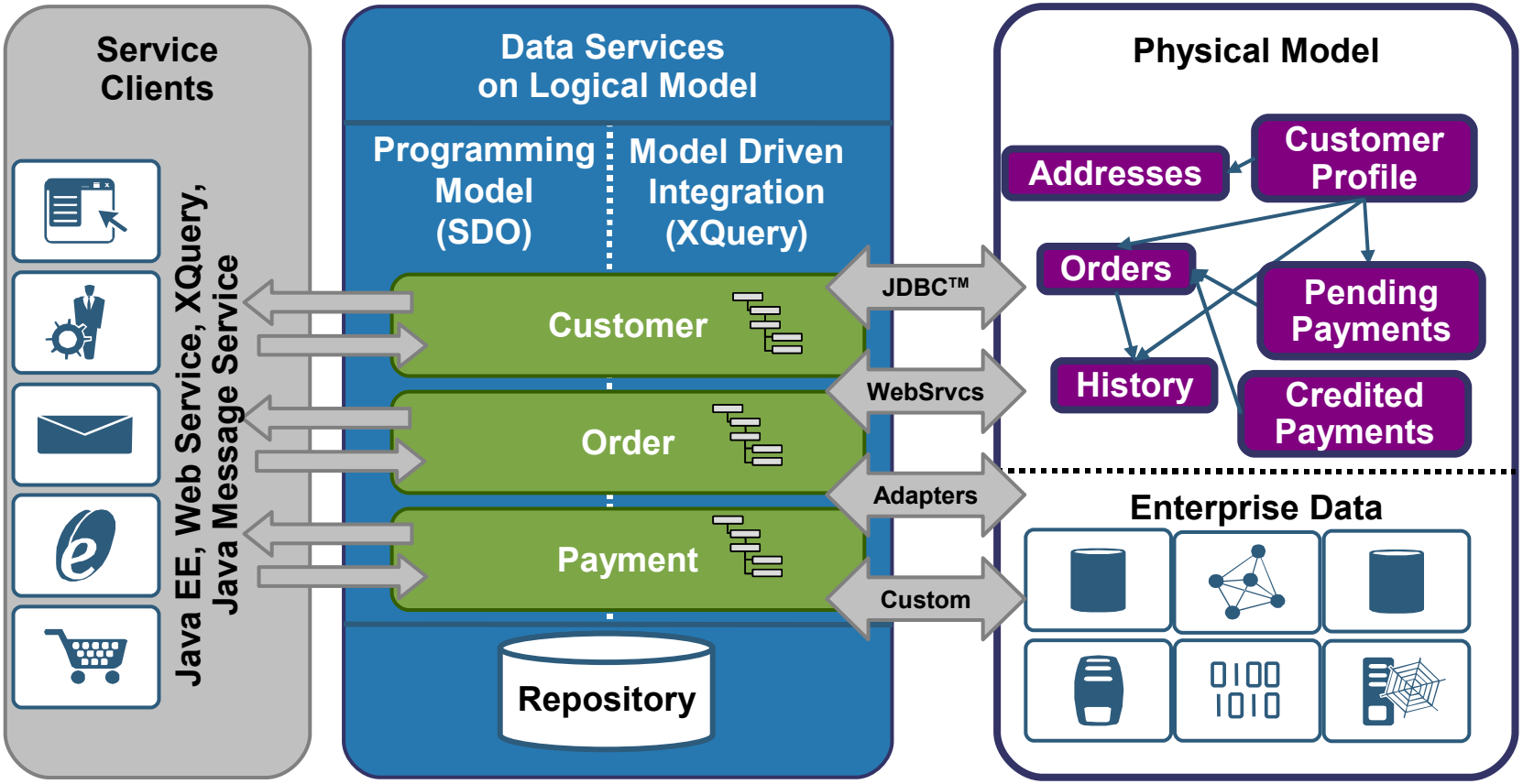
SDO—Key Concepts and Core APIs

SDO in Action

Summary

SDO in Action—BEA

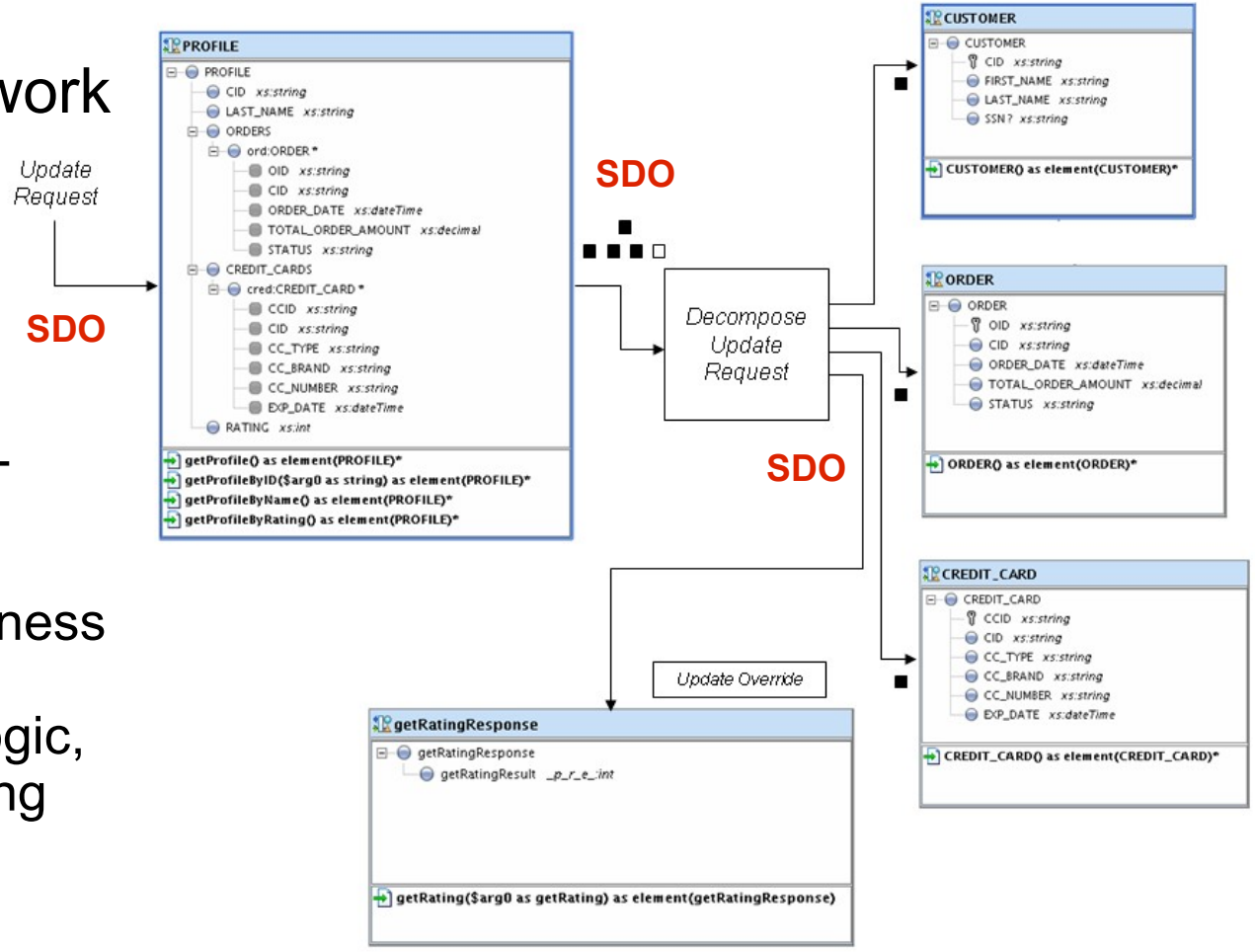
Data Services in AquaLogic DSP



- Logical models capture data access and integration complexity once
- Same data model, programming model, and API for all enterprise data

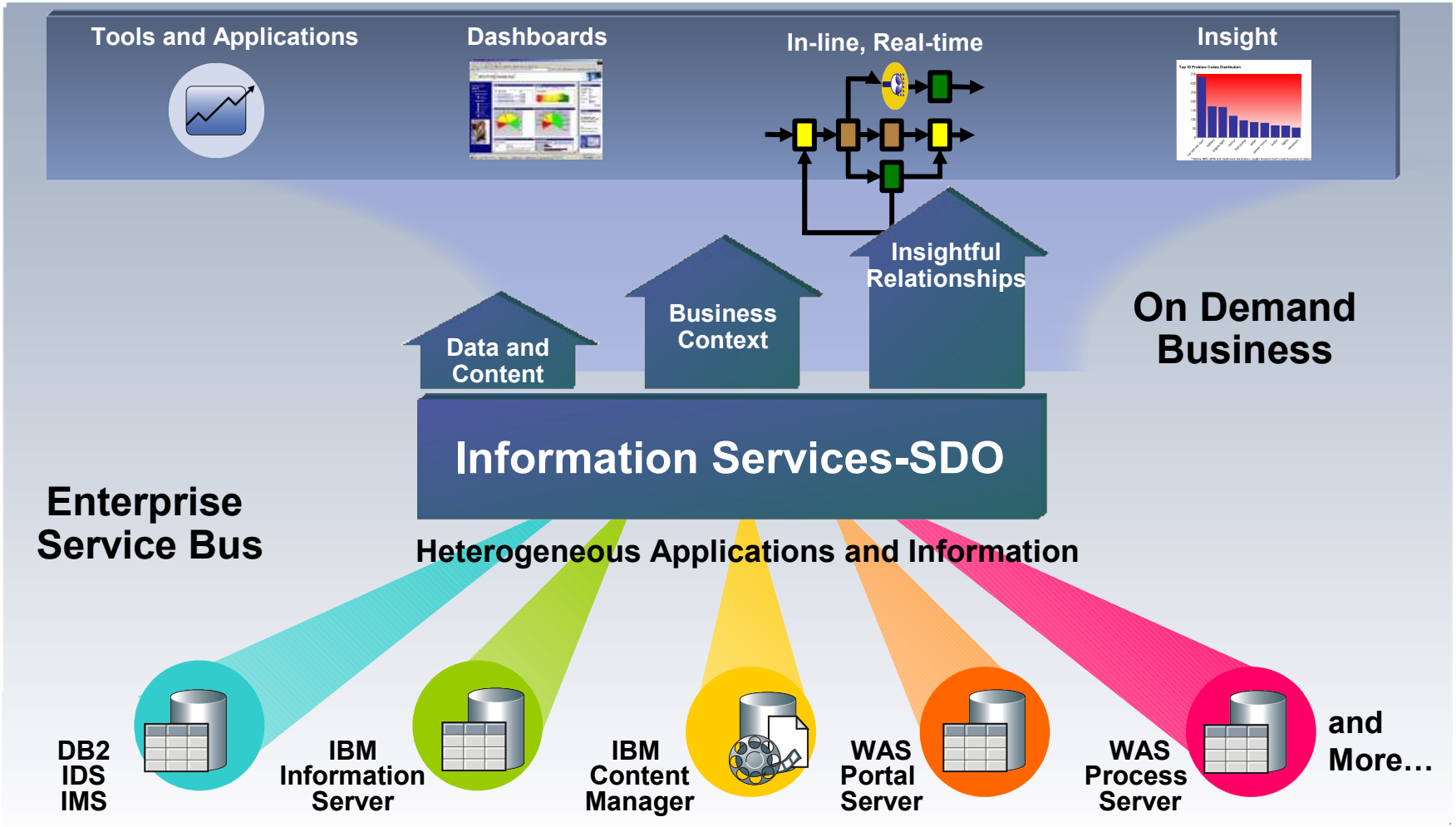
AquaLogic DSP Update Automation

- Update Framework
  - XA and non-XA sources
  - Automated change decomposition
  - Automatic SQL generation for RDBMS
  - Hooks for business validations, replacement logic, or compensating transactions
  - ADO.NET interoperability



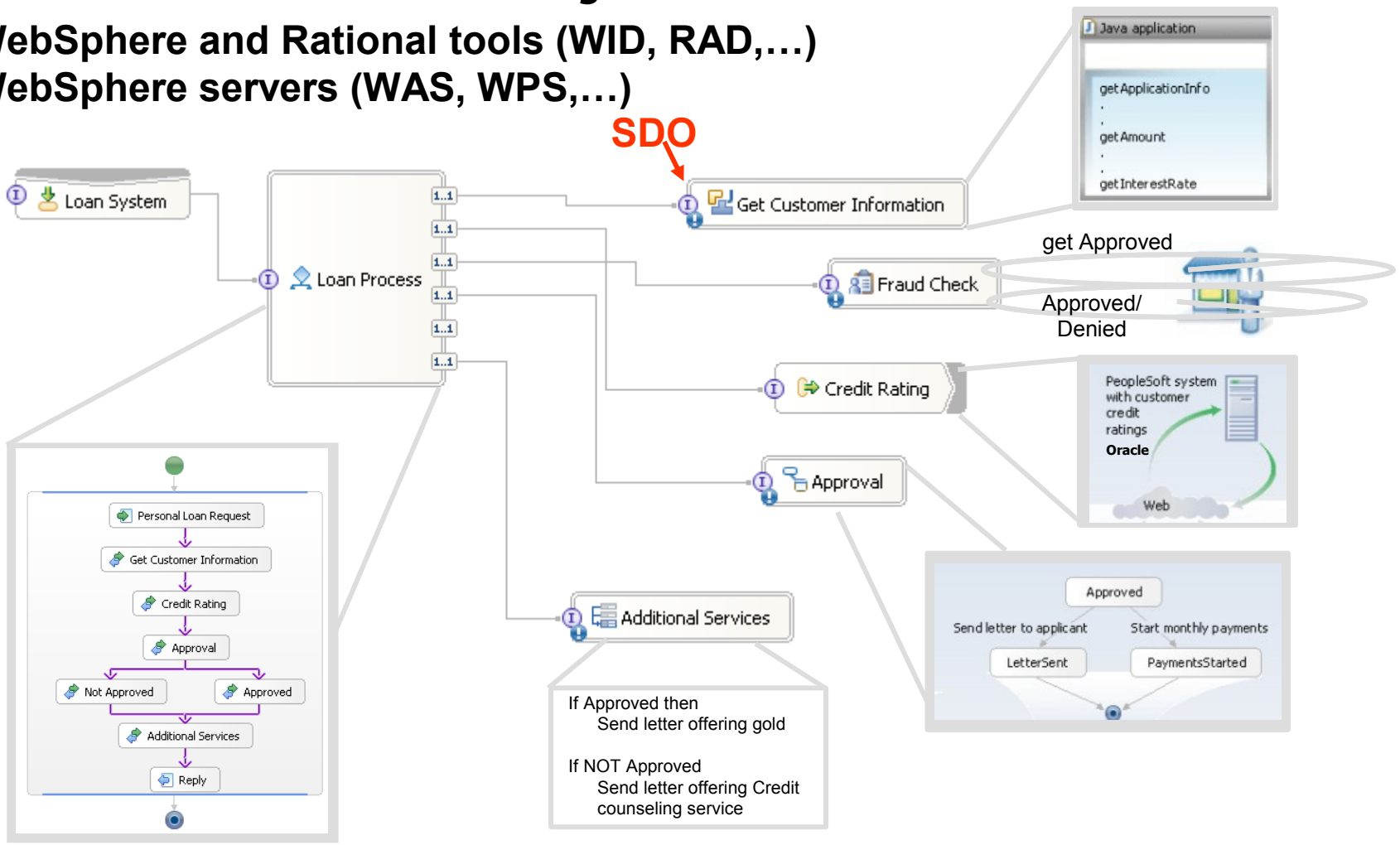
SDO in Action—IBM

Information as a Service—IBM SOA Foundation



IBM Business Objects Are SDOs

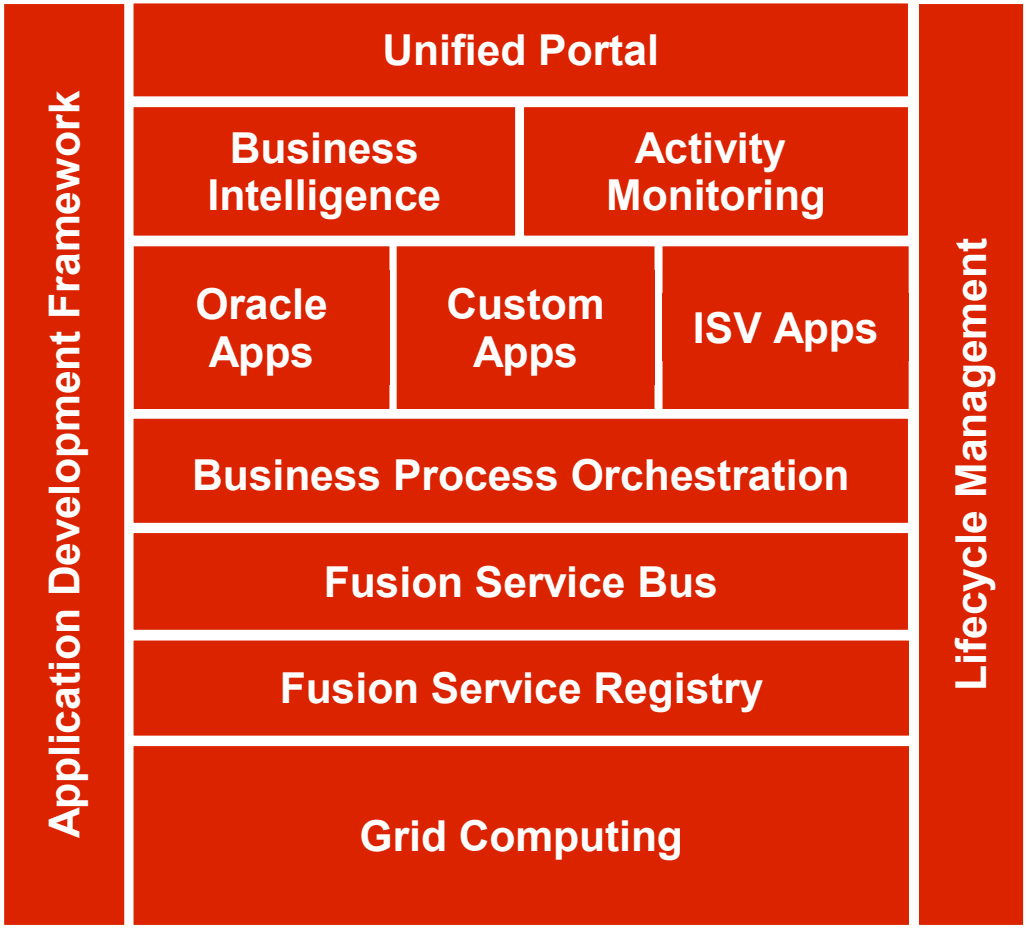
WebSphere and Rational tools (WID, RAD,…)
WebSphere servers (WAS, WPS,…)



SDO in Action—Oracle

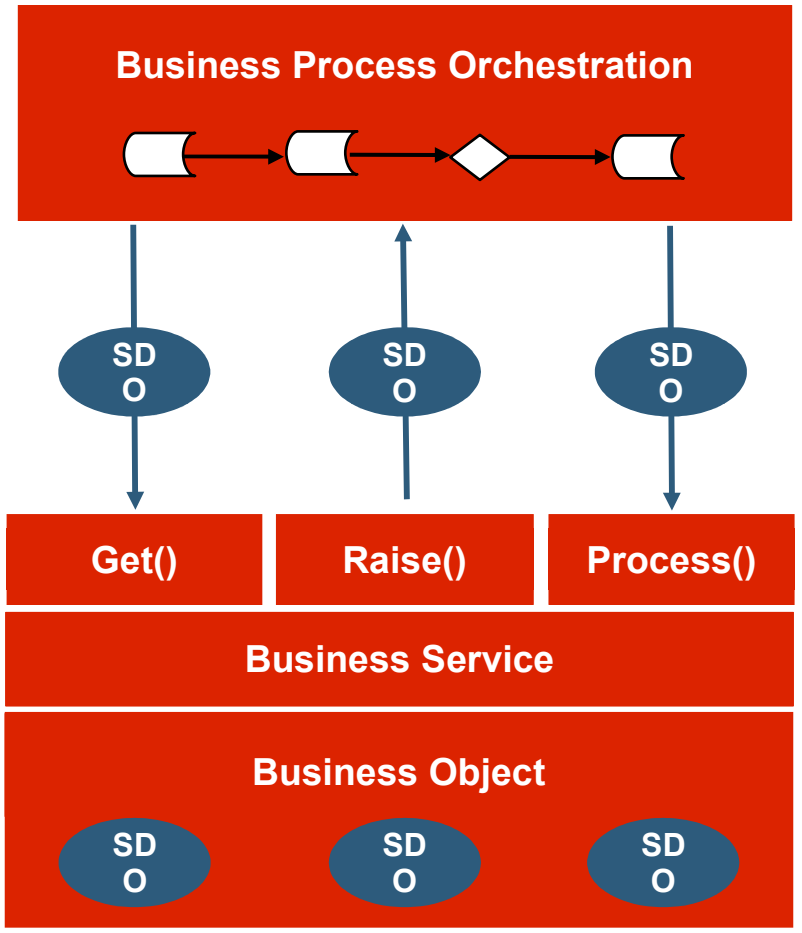
Oracle Fusion Architecture

- Model Driven
- Service Oriented
- Grid architecture
- Information Centric
- Standards based
- Hot-pluggable



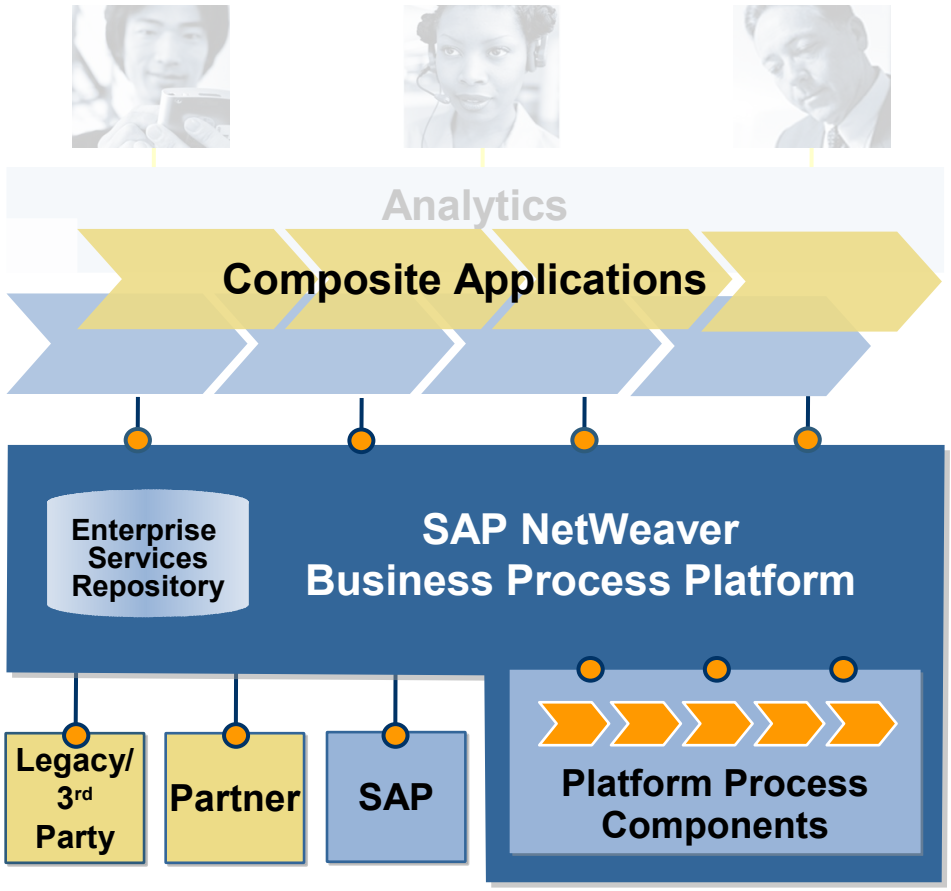
SDO in Fusion Applications

- Oracle Fusion Application Services support the SDO Standard
- Service Data Objects facilitate
  - Separate Data Object implementation from service
  - Easy vertical extensions
  - Standard means for integration (a2a, .net,…)
- Oracle Business Objects are SDOs



SDO in Action—SAP

SAP's Business Process Platform



PEOPLE PRODUCTIVITY

EMBEDDED ANALYTICS

APPLICATION COMPOSITION

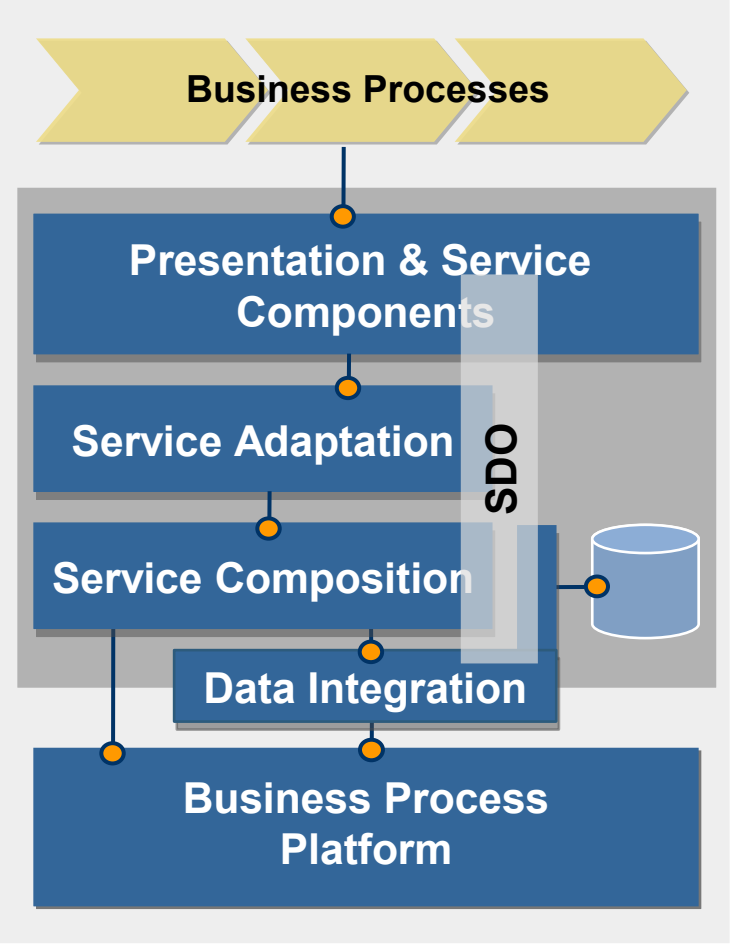
SERVICE ENABLEMENT

BUSINESS PROCESS PLATFORM

- Business level models
- Enterprise services/ Business objects
- Ready-to-run business processes

Service Data Objects in Netweaver

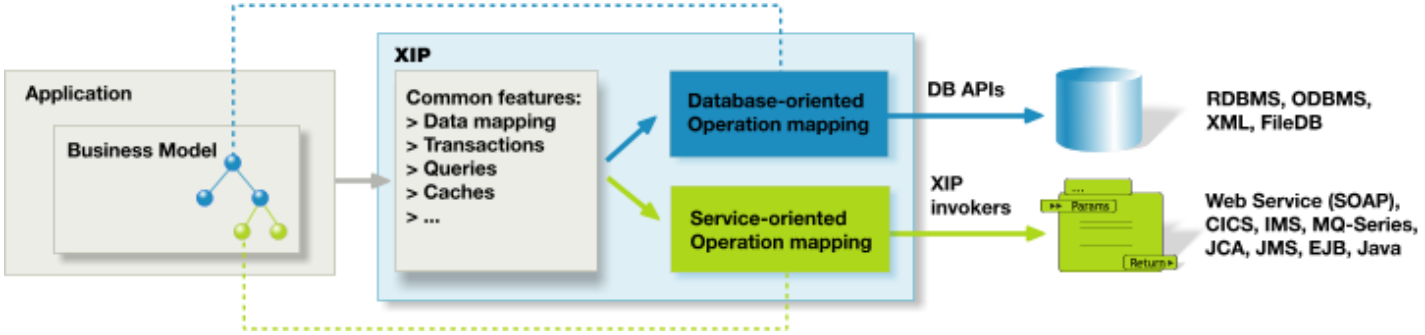
- SAP NetWeaver enables loosely coupled and distributed business processes
- Efficient and useful provision of distributed data is crucial!
- Service Data Objects provide the language bindings for
  - Data representation,
  - Meta-data access, and
  - State transfer
- Of business data in composite applications
- SAP considers SDO a key technology in the next major SAP NetWeaver release



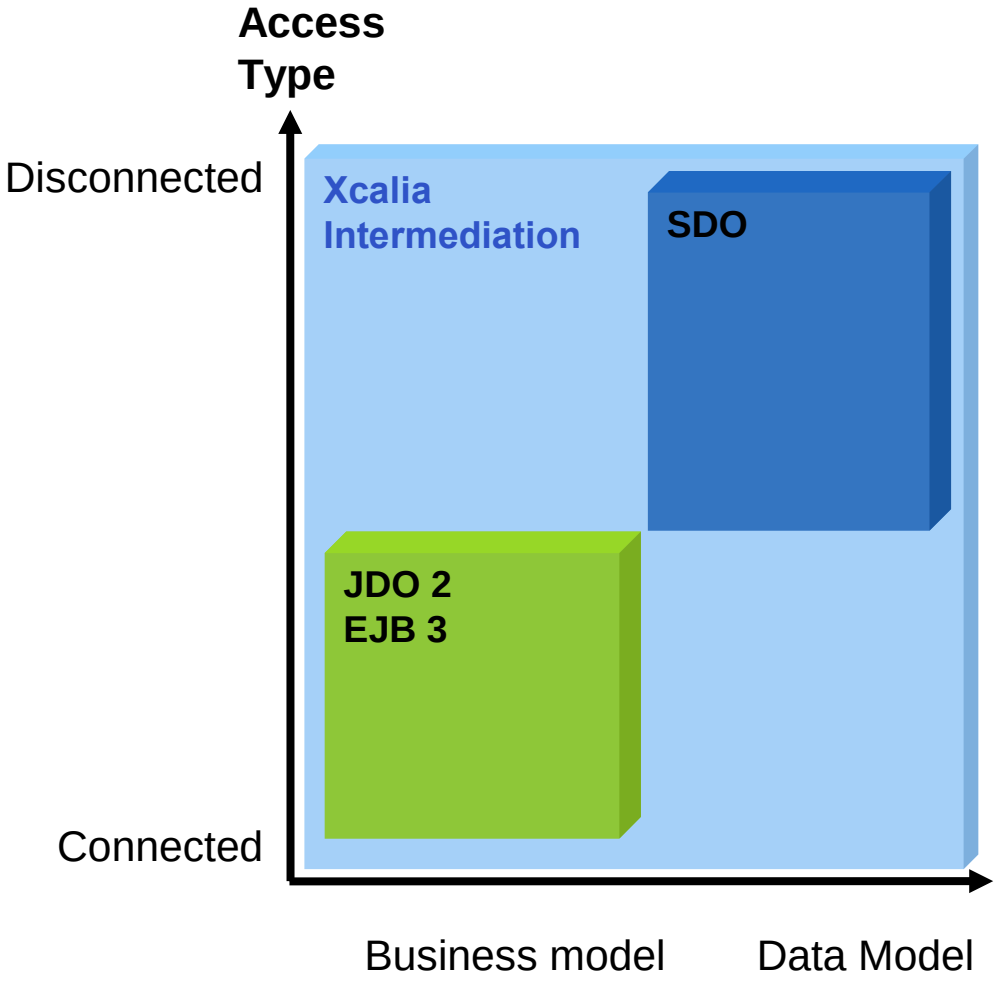
SDO in Action—Xcalia

Service Data Objects and Xcalia

- Xcalia is a core contributor and has implemented SDO since 2005
- We have several SDO deployments in production to:
  - Deliver data to rich clients, manifest an extended DTO pattern
  - Manage dynamic, reflexive data models
- The Xcalia Intermediation Platform provides access to both data and service resources
  - We offer a complete solution that implements Java Data Objects (JDO), EJBTM 3.0 architecture's JPA and SDO
  - Our SDO implementation is complemented by our mature data access service



SDO and Xcalia



- Xcalia's SDO implementation deals with dynamic mapping of data when no business logic is required (web services, simple web applications, reporting, etc.)
- SDO compliments and does not replace POJO oriented-persistence (EJB 3, JDO 2 specs)!

SDO in Action—Open Source

Open Source SDO—Apache Tuscany

- Tuscany Project (http://incubator.apache.org/tuscany)
- Provides runtime capabilities for applications built using a Service Oriented Architecture (SOA)
- Implementations of two specifications:
  1. Service Component Architecture (SCA)
  2. Service Data Objects (SDO)
- plus
  + Data Access Service (DAS)
- In total SCA Runtime for Java technology
  1. SDO 2.01 Runtime for Java technology
  2. Data Access Service for Java technology
  3. SCA runtime for C++
  4. SDO 2.01 Runtime for C++
  5. Tuscany currently includes 5 subprojects

Apache Tuscany Status

- Initiated January 2006 and currently in incubation
- SDO for Java technology (sub)project goals
  1. 100% implementation of 2.01 Specification by end of 2006
     - Currently approximately 60-70% complete
  2. Identify issues with SDO spec and provide feedback for future versions (e.g., 2.1 and 3.0)
  3. Provide value add features possibly including
     - Static Java technology code generation
     - Dynamic java class (bytecode) generation (using ASM)
     - Import SDO metadata from (annotated) Java language interfaces
     - High performance generated loaders/serializers
     - Lots of other possibilities
- Looking for volunteers to contribute!
  - Go to http://incubator.apache.org/tuscany/ for more info on how to get involved

Agenda

SDO—The Big Picture

SDO—Key Concepts and Core APIs

SDO in Action

Summary

Summary

- SDO Programming Model
  - One model for data across the enterprise
    - XML, Relational, Object
    - Generated and Dynamic
  - SOA patterns (disconnected clients, data services)
  - Efficient change communication across services
- SDO Components
  - Generated data API: POJO beans
  - Dynamic data API: DataObject
  - Change summary API: ChangeSummary
  - Introspection API: Type and Property
  - XML serialization on the wire

SDO Is Useful Everywhere

- Service Oriented Architecture (SOA)
  - SDOs are the input and output of services
- Web Services
  - SDOs represent the XML on the wire
- XML
  - When XML-enabling an application
  - When accessing XML files/documents/resources/messages
- EJB
  - SDOs are Data Transfer Objects (DTO), value objects
  - Java EE Design Pattern
- Enterprise Service Bus (ESB)
  - SDOs are the input and output of services
- Model Driven Architecture (MDA)
  - SDO model (Type and Property) defined by Unified Modeling Language (UML) Classes and Components
  - SDO applications follow UML Sequence, Flow, State, and Collaboration
- Data access
  - Access relational, XML, EJB, JDO, Hibernate data sources
  - SDOs are the DTOs
- Messaging
  - SDOs represent the messages
- Connectors/Adapters (EIS, CICS)
  - SDOs represent the data records
- BPEL-J
  - SDOs are the Java business objects
- ADO.NET
  - DataSet is a subset of SDO Data Graphs
- Cross-language programming model
  - Complete applications may span tiers, languages
- Java technology
  - SDOs are smart POJOs with POJO interfaces

For More Information

- SDO Specification
  http://dev2dev.bea.com/pub/a/2005/11/sdo.html
- BEA AquaLogic Data Services Platform
  http://www.bea.com/dataservices/
- IBM
  http://ibm.com/SOA
- Oracle
  http://www.oracle.com
- SAP
  http://www.sap.com
- Xcalia
  http://www.xcalia.com

Q&A

Stephen Brodsky	IBM
Michael Carey	BEA Systems
Blaise Doughan	Oracle
Henning Blohm	SAP
Eric Samson	Xcalia Add title here

JSR 235 and Service Data Objects (SDO)

Stephen Brodsky
sbrodsky@ibm.com
IBM, Silicon Valley Lab

Michael Carey
mcarey@bea.com
BEA Systems, Inc.

TS-3676

Comparison of Data APIs

	Model	API	Data Source	MetaData API	Query Language
SDO	Disconnected	Both	Any	SDO Metadata API, Java Introspection	Any
JDBC Rowset	Connected	Dynamic	Relational	Relational	SQL
JDBC Cached Rowset	Disconnected	Dynamic	Relational	Relational	SQL
Entity EJB	Connected	Static	Relational	Java Introspection	EJBQL
JDO	Connected	Static	Relational, Object	Java Introspection	JDOQL
JCA	Disconnected	Dynamic	Record-based	Undefined	Undefined
DOM and SAX	Disconnected	Dynamic	XML	XML InfoSet	XPath, XQuery
JAXB	Disconnected	Static	XML	Java Introspection	N/A
JAX-RPC	Disconnected	Static	XML	Java Introspection	N/A