

ONE CLIENT THAT RULES THEM ALL

Thomas Liou

Department of the Treasury

10/04/2012

JavaOne 2012

Agenda

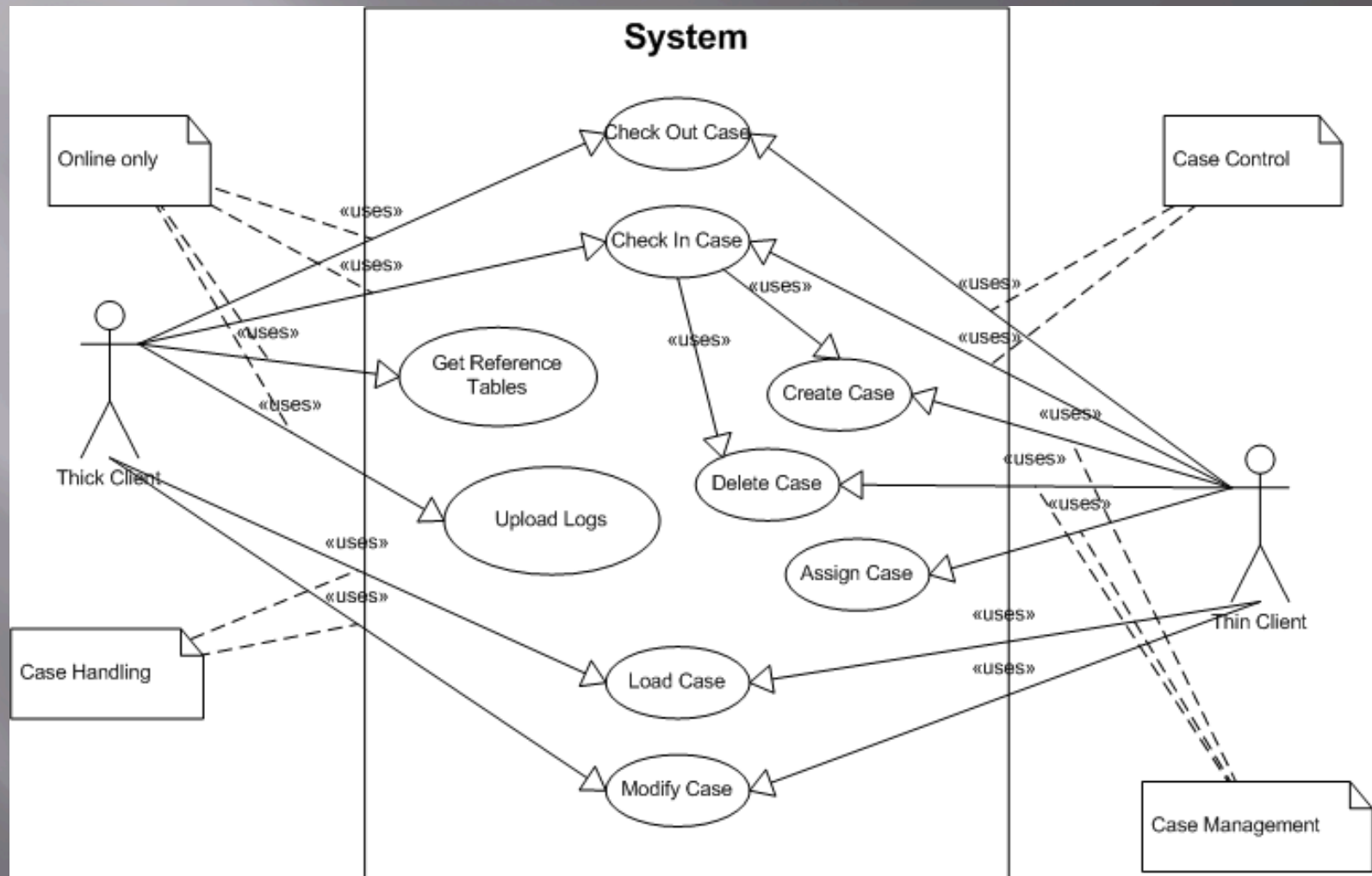
1. *Business Needs*
2. *Architecture*
3. *JavaFX Client*

Demo

References

1. Business Needs

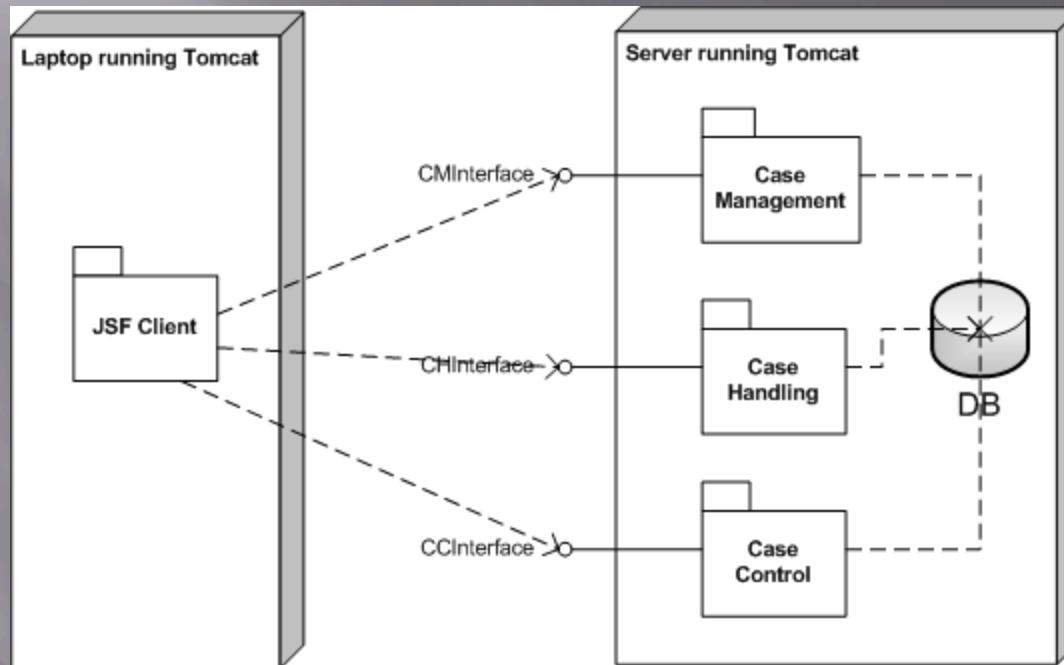
Use Case Context Diagram



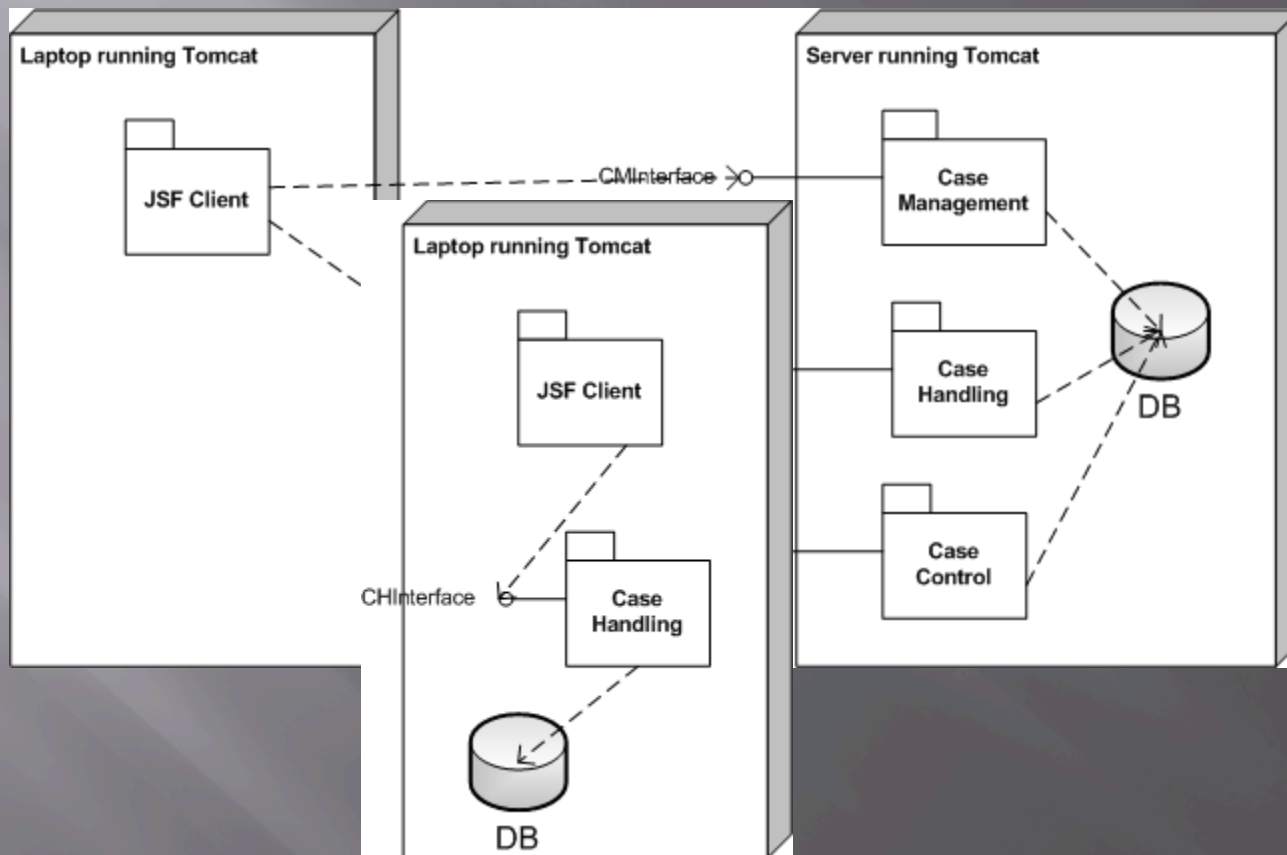
Needs

1. Stand-alone Swing application for occasionally disconnected client
2. Web application for always connected client
3. Database synchronization software to keep database on disconnect client in sync with server database

Client with Network Access



Client without Network Access



Major Benefits

- ▣ No need to create stand-alone Swing client
- ▣ Seamless switch when network is available
- ▣ Centralized control of the application

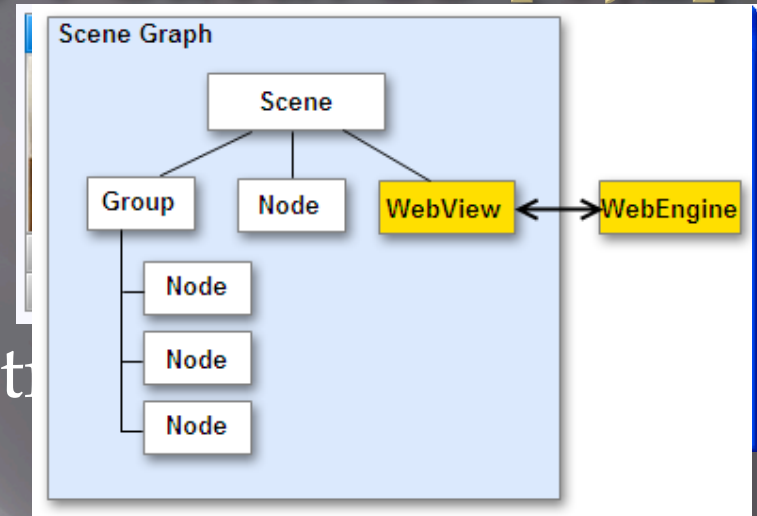
JavaFX Client

Write GUI once, deploy JavaFX many times!

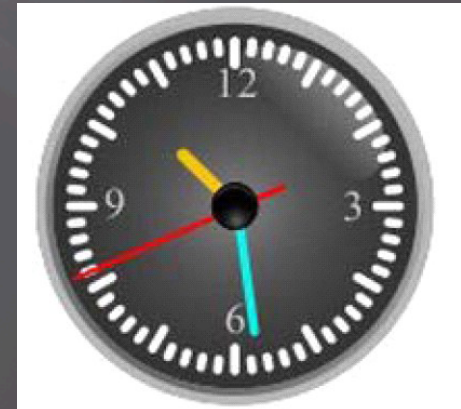
- Deployment modes
 - standalone / applet/ web start
 - native packaging (2.2)
- Major operating systems
 - Windows / Linux
 - Android / iOS
- Many platforms
 - desktops / laptops
 - phones / tablets
 - TVs

Selected JavaFX 2 Features [1,2]

- ▣ FXML
- ▣ A wide variety of UI controls
- ▣ A web component
- ▣ HTML5 support
 - Editable content
 - History maintenance
 - SVG

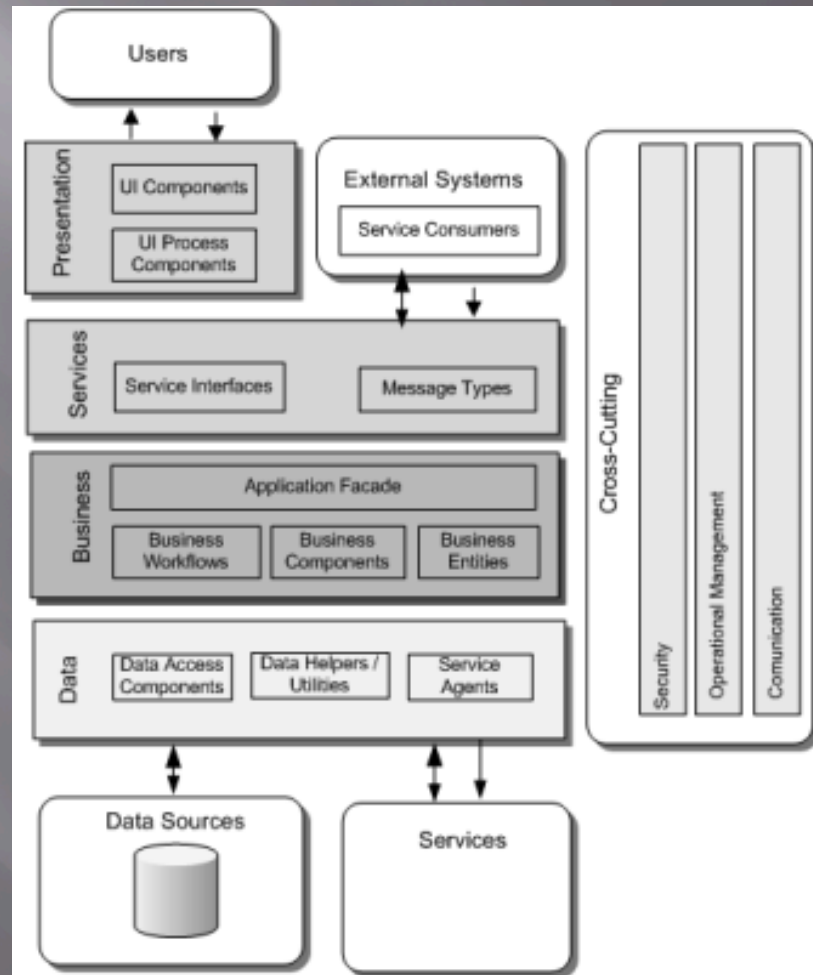


```
WebView browser = new WebView();  
WebEngine webEngine = browser.getEngine();  
webEngine.load("http://mySite.com");
```

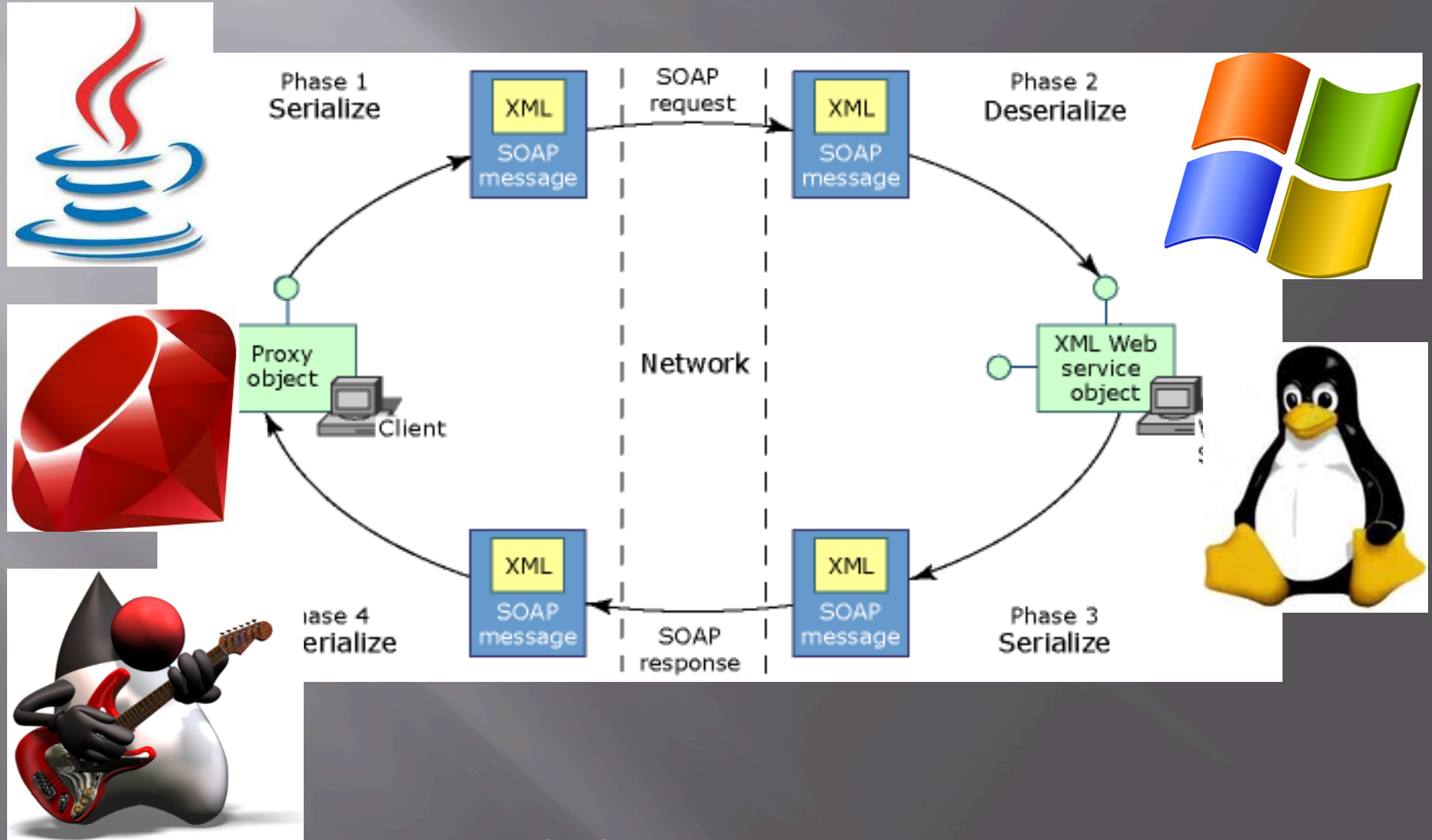


2. Architecture

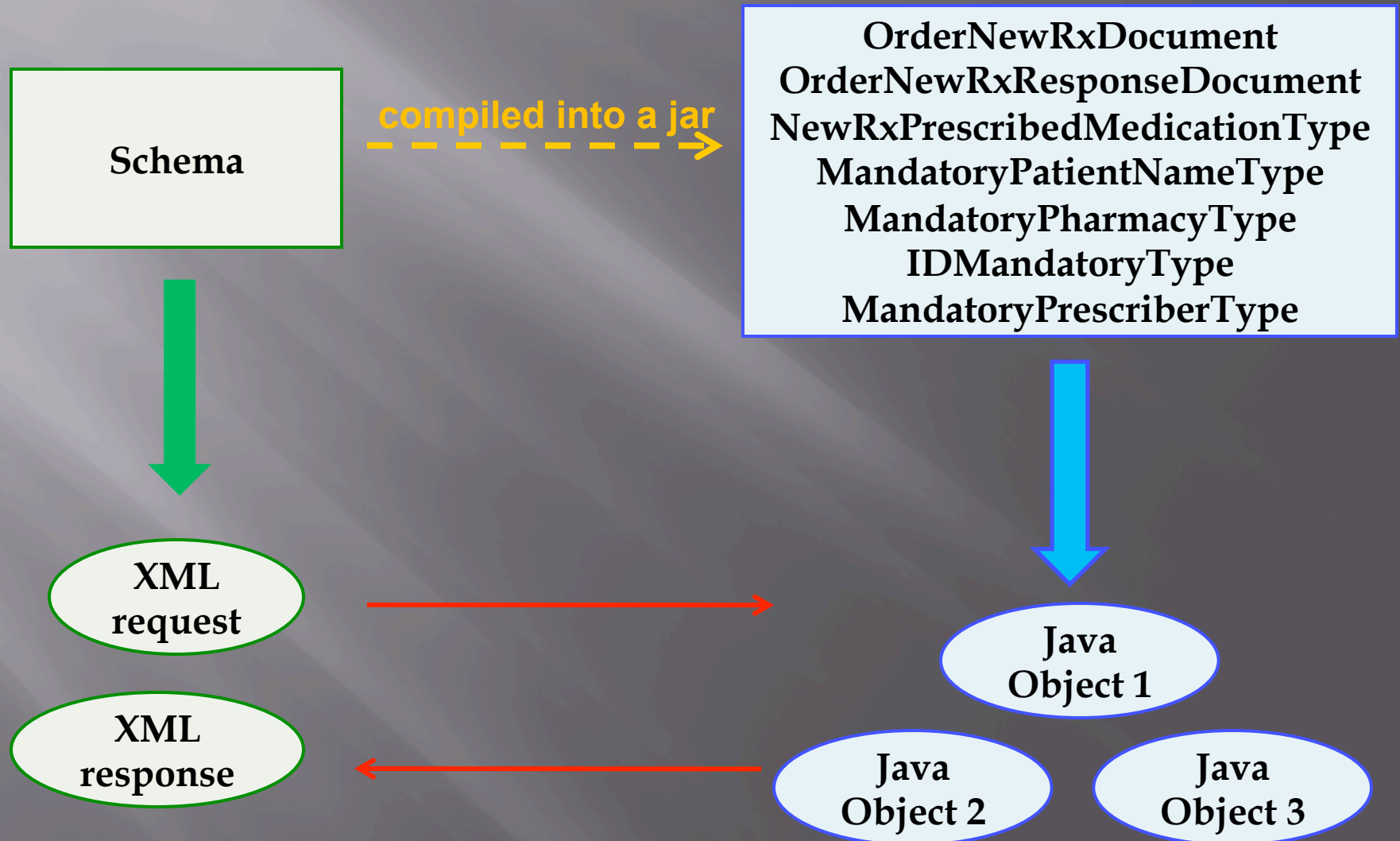
The Big Picture [3]



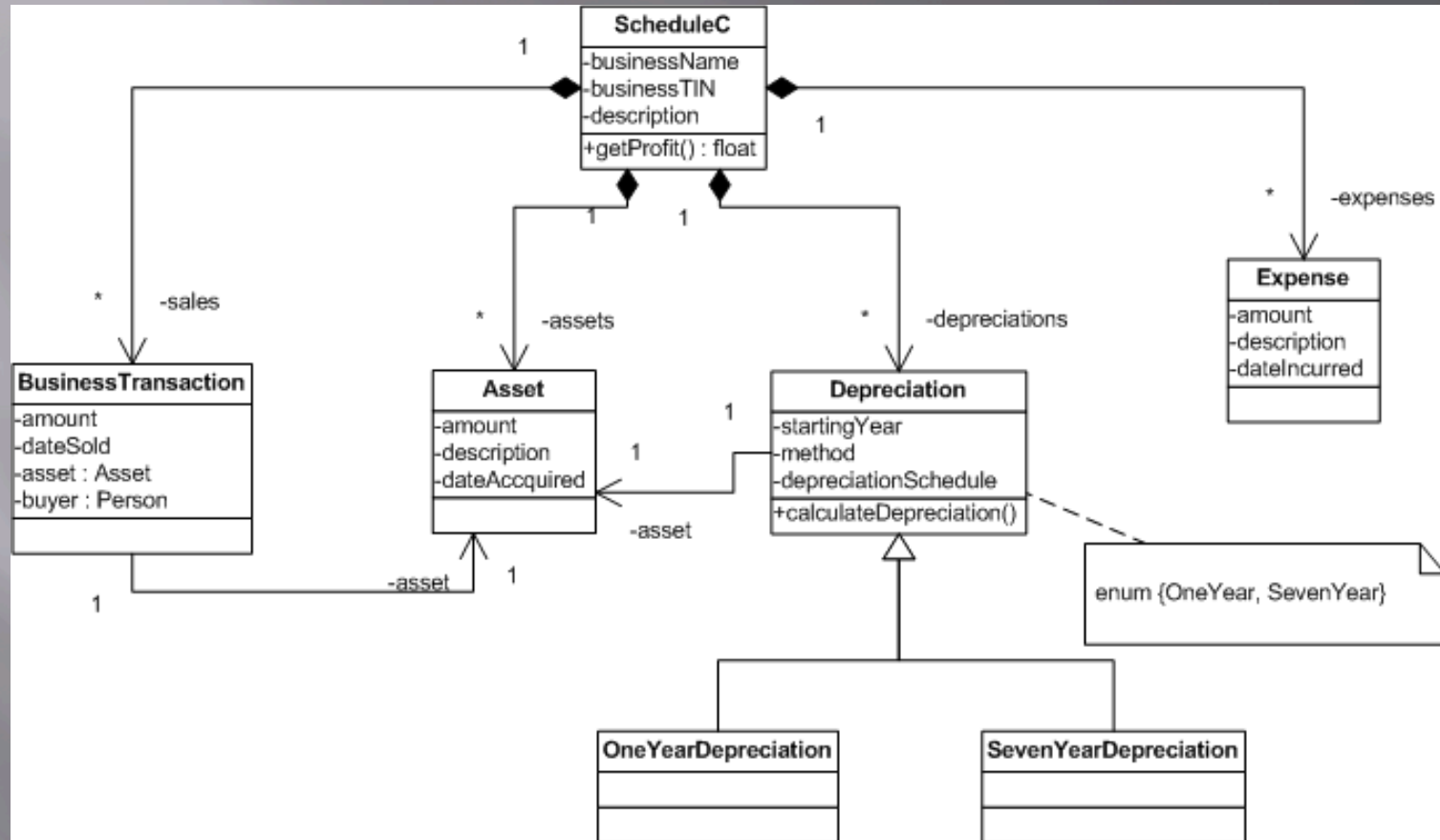
Service Layer – CXF [4]



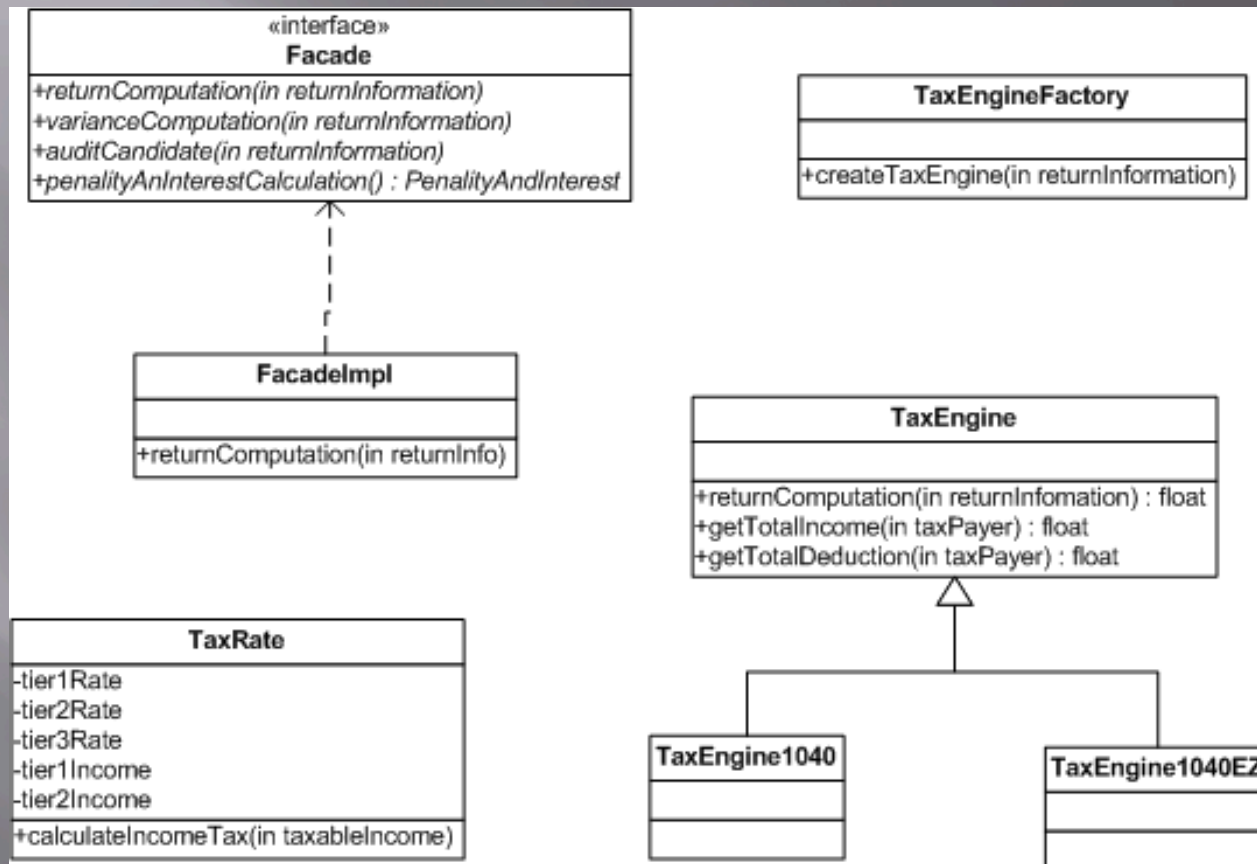
XMLBeans



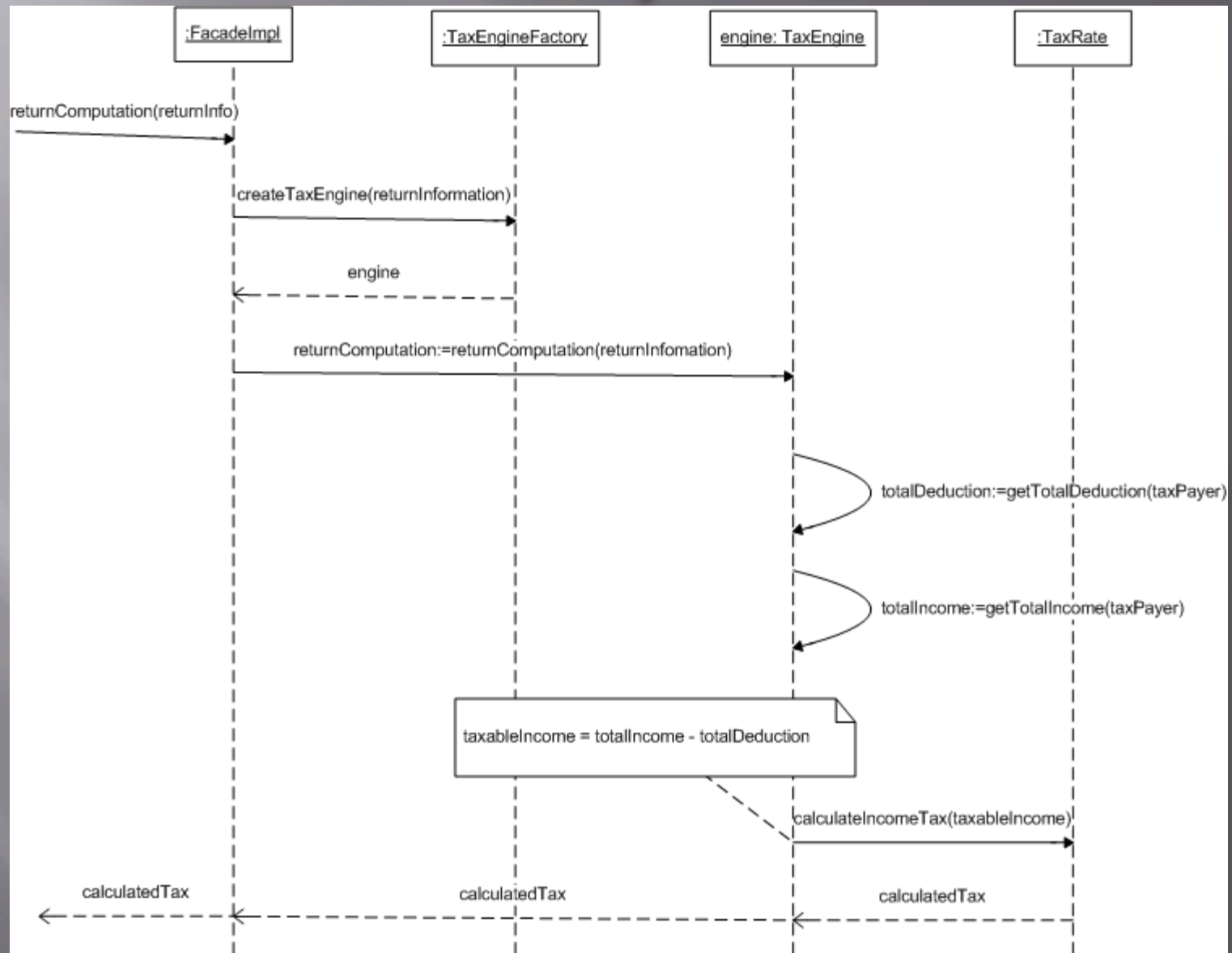
Business Layer – Domain Objects



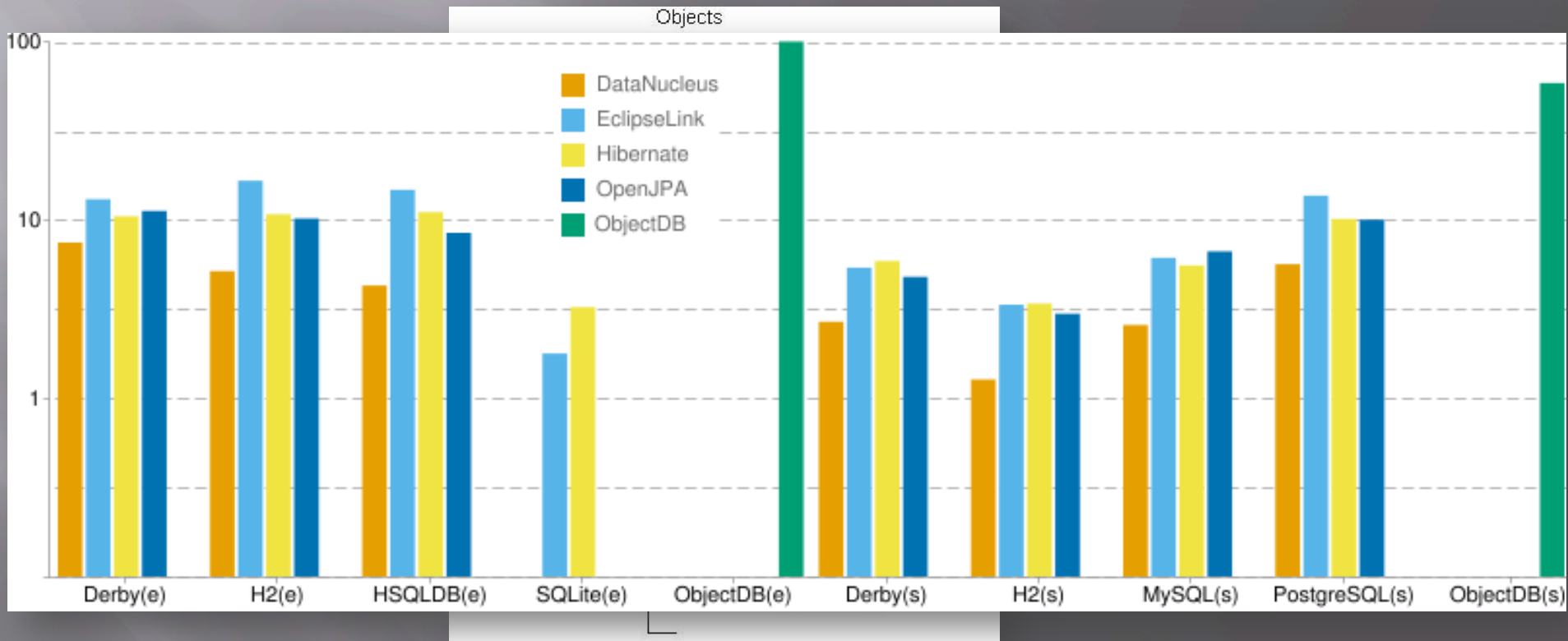
Business Layer – Server Objects



Return Computation SD



Persistence (Data) Layer – JPA [5,6]



3. JavaFX Client

Presentation Layer – JavaFX [7]

iCalculate

Payer Information: First Name Last Name Logout

SSN

Address: Street Calculate Federal Tax

City State Zip

Dependents Income Deduction Business Graph

Enter all dependant information here.

Relations...	First Name	Last Name	SSN	DOB	Same Ad...	
SPOUSE	Jane	Doe	222-22-2222	01/01/1980	true	
SON	Jack	Doe	333-33-3333	01/01/2000	true	

Relationship ☒ Same Address

Step-by-Step Guide

1. Define model or domain objects (IDE)
2. Design GUI screens (Scene Builder)
3. Create control objects (IDE)
 1. Create view objects
 2. Create event-handling method stubs
4. Wire screen components to view and control (SB)
5. Implement method stubs (IDE)
 1. to keep model and view in sync
 2. to invoke web services
6. Test JavaFX client (IDE)
7. Test deployment

1. Define Model Objects

The screenshot shows the 'Web Service Client' application window. In the background, the 'Web Services' tab is active, displaying a list of service definitions and a 'Client type' dropdown set to 'Java'. A modal dialog titled 'Current Assets Information' is open, featuring two tables for data entry. The first table, 'Current Assets Information', has columns for Amount, Date, and Description. The second table, 'Current Sales Information', has columns for Amount, Date Sold, Buyer, and Asset. Both tables have an 'Add' button to the right. The dialog also includes a 'Delete' button in the top right corner. At the bottom of the dialog, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The background window also shows a 'Monitor the W' checkbox and an 'Overwrite files without warning' checkbox.

Current Assets Information

Amount	Date	Description
500.0	2/2/2012	STV 600 violin
700.0	3/3/2012	STV 750 violin

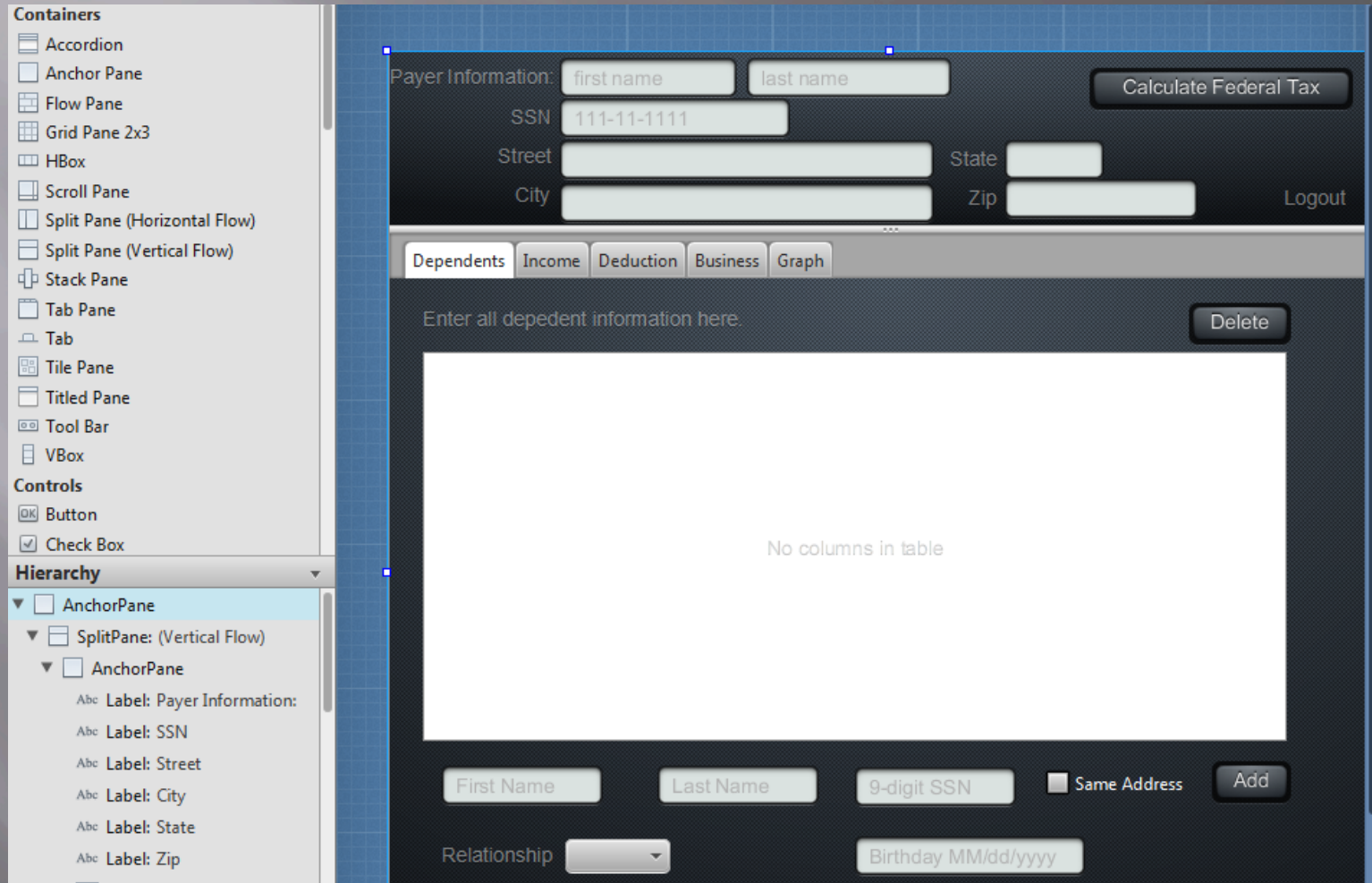
Current Sales Information

Amount	Date Sold	Buyer	Asset
700.0	7/7/2012	Jack Black	STV 600 violin acquired on 2/2/2012

ws.proxy.domain

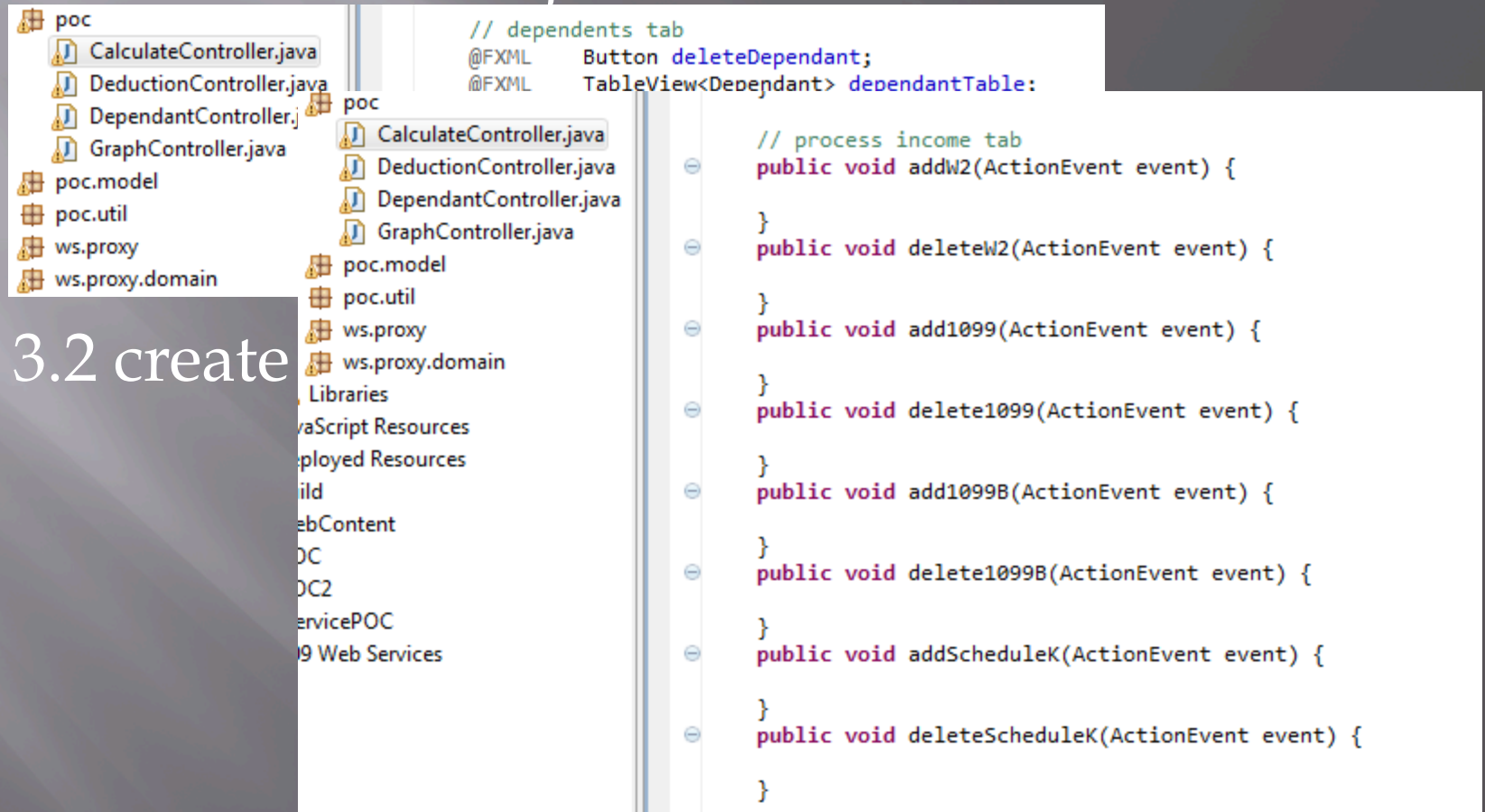
- Address.java
- Asset.java
- AuditCandidate.java
- PhoneType.java
- Relationship.java
- RetrieveReturnInfo.java
- RetrieveReturnInfoResponse.java
- ReturnComputation.java
- ReturnComputationResponse.java
- ReturnForm.java
- ReturnInformation.java

2. Design GUI Screens



3. Create Control Objects

3.1 create view objects



The screenshot displays an IDE with a project structure on the left and two code editors on the right. The project structure shows a package named 'poc' containing several Java files: CalculateController.java, DeductionController.java, DependantController.java, and GraphController.java. Below the 'poc' package are sub-packages: poc.model, poc.util, ws.proxy, and ws.proxy.domain. The code editors show the following content:

```
// depends tab
@FXML Button deleteDependant;
@FXML TableView<Dependant> dependantTable:

// process income tab
public void addW2(ActionEvent event) {
}
public void deleteW2(ActionEvent event) {
}
public void add1099(ActionEvent event) {
}
public void delete1099(ActionEvent event) {
}
public void add1099B(ActionEvent event) {
}
public void delete1099B(ActionEvent event) {
}
public void addScheduleK(ActionEvent event) {
}
public void deleteScheduleK(ActionEvent event) {
}
```

3.2 create

4. Wire Screen Components to View Objects

The image displays a software development environment with a wireframe of a 'Dependents' form and its associated properties.

Dependents Form Wireframe:

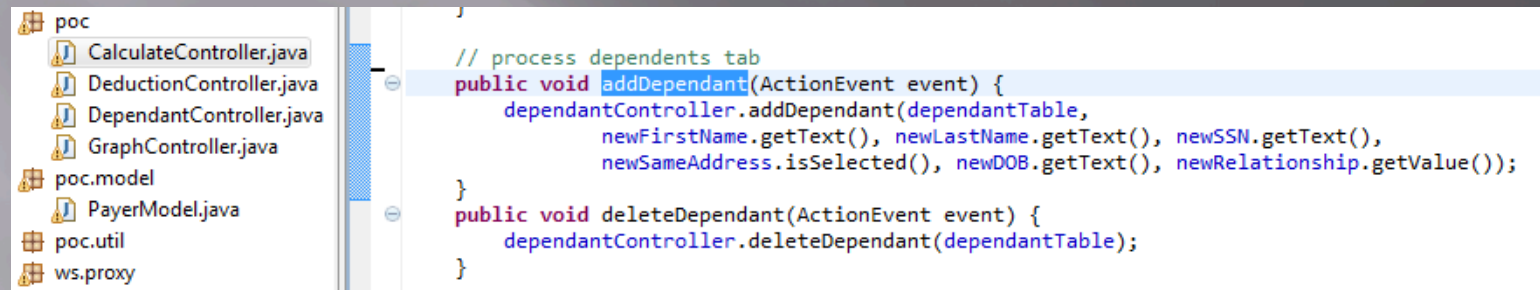
- Header:** Tabs for 'Dependents', 'Income', 'Deduction', 'Business', and 'Graph'.
- Text:** 'Enter all dependant information here.'
- Buttons:** 'Delete' (top right), 'Add' (bottom right, highlighted with a blue dashed border).
- Form Fields:** 'First Name' (text input), 'Last Name' (text input), '9-digit SSN' (text input), 'Same Address' (checkbox), 'Relationship' (dropdown), and 'Birthday MM/dd/yyyy' (text input).
- Table:** A large white rectangular area in the center with the text 'No columns in table'.

Properties Panels:

- Properties : TableView**
 - fx:id: dependantTable
 - Editable: ☒
 - Table Menu Butt...: ☐
 - Sort Order: []
- Properties : Button**
 - Layout : Button
 - Events : Button
 - On Action: #addDependant
 - Drag and Drop
 - On Drag Detected: Empty
 - On Drag Done: Empty
 - On Drag Dropped: Empty
 - On Drag Entered: Empty
 - On Drag Exited: Empty
 - On Drag Over: Empty

5. Implement Method Stubs

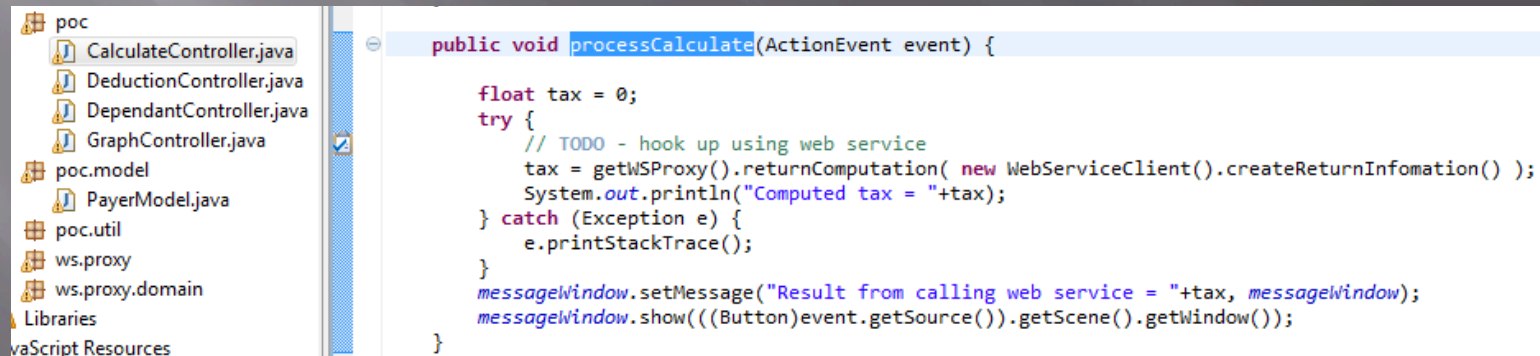
▣ 5.1 keep model and view in sync



```
// process dependants tab
public void addDependant(ActionEvent event) {
    dependantController.addDependant(dependantTable,
        newFirstName.getText(), newLastName.getText(), newSSN.getText(),
        newSameAddress.isSelected(), newDOB.getText(), newRelationship.getValue());
}

public void deleteDependant(ActionEvent event) {
    dependantController.deleteDependant(dependantTable);
}
```

▣ 5.2 invoke web services



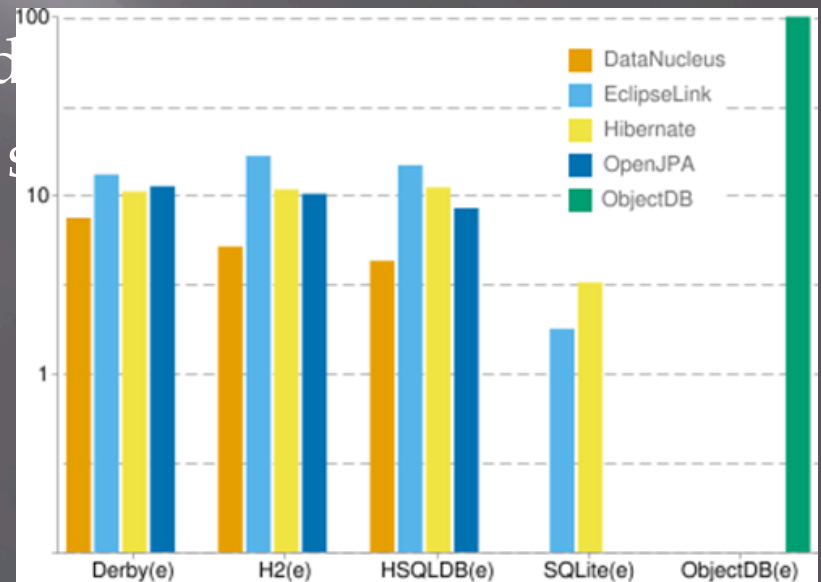
```
public void processCalculate(ActionEvent event) {

    float tax = 0;
    try {
        // TODO - hook up using web service
        tax = getWSProxy().returnComputation( new WebServiceClient().createReturnInfomation() );
        System.out.println("Computed tax = "+tax);
    } catch (Exception e) {
        e.printStackTrace();
    }
    messageWindow.setMessage("Result from calling web service = "+tax, messageWindow);
    messageWindow.show(((Button)event.getSource()).getScene().getWindow());
}
```


Demo

What's Next?

- ▣ Submit JavaFX POC for approval to use within the organization
- ▣ Persistence (Data) Layer
 - ▣ Java DB POC – embedded
 - ▣ ObjectDB POC – DB for s
- ▣ Business Layer
 - ▣ Drools POC



References

1. Cindy Castillo, [What is JavaFX?](#), Oracle, December 2011.
2. James Waeber, Weigi Gao, Stephen Chin, and Dean Iverson, Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology, Apress, February 2012.
3. J. D. Meier's Blog, [Application Architectures](#), 2008.
4. Thomas Liou, Comparing Axis2 and CXF Web Service Frameworks, JavaOne, 2010.
5. Douglas Barry, [Impedance mismatch when mapping from a relational database](#), www.server-architecture.com.
6. ObjectDB, [JPA Performance Benchmark](#), 2011.
7. [JavaFX Scene Builder Examples](#), Oracle, 2012.