The web framework dream team

Johan Eltes

Johan.eltes@callistaenterprise.se
www.callistaenterprise.se

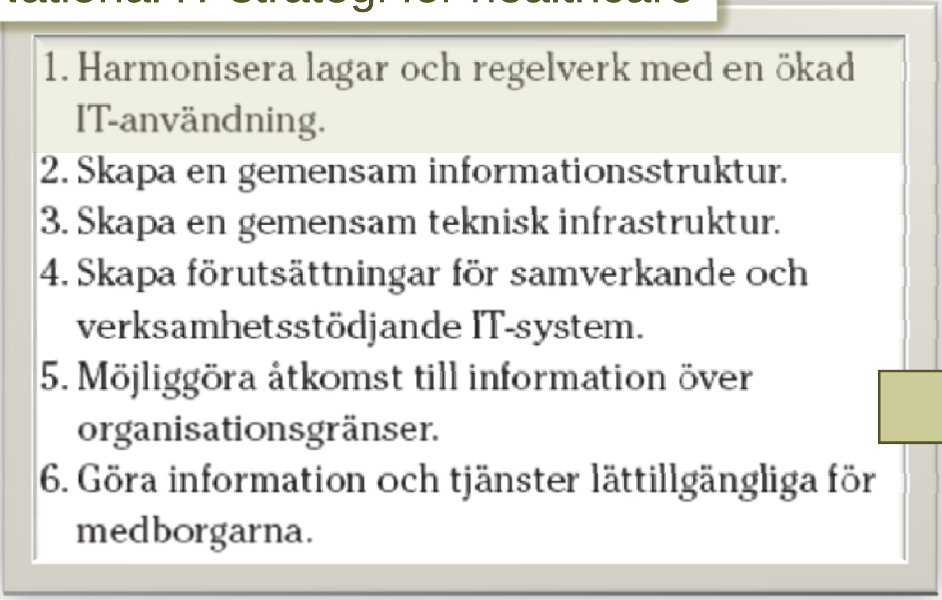
CALLISTA

Agenda

- Customer Case Background
- Requirements
- The reference architecture
- The frameworks and their contributions
- Lessons learned

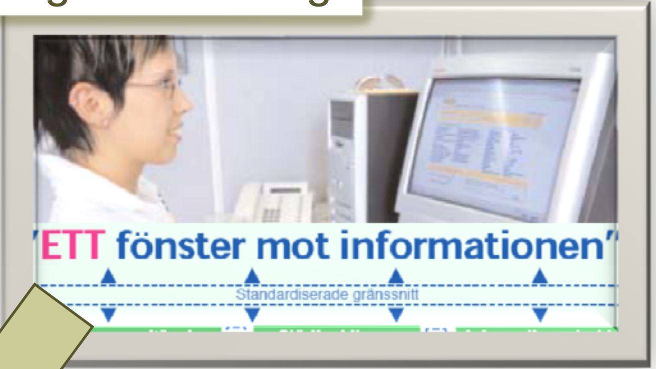
Strategic drivers behind reference architecture

National IT strategi for healthcare



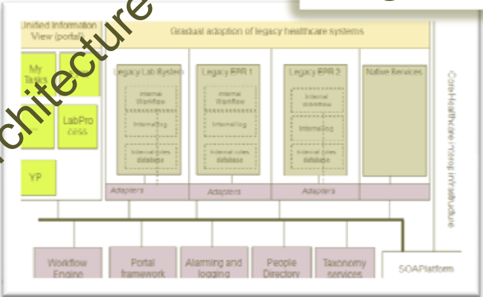
1. Harmonisera lagar och regelverk med en ökad IT-användning.
2. Skapa en gemensam informationsstruktur.
3. Skapa en gemensam teknisk infrastruktur.
4. Skapa förutsättningar för samverkande och verksamhetsstödjande IT-system.
5. Möjliggöra åtkomst till information över organisationsgränser.
6. Göra information och tjänster lättillgängliga för medborgarna.

Regional strategi

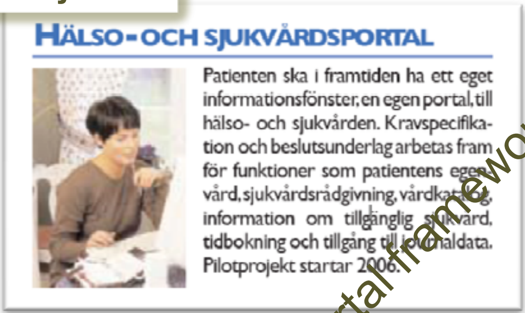


Regional projects

IT architecture



Portal framework

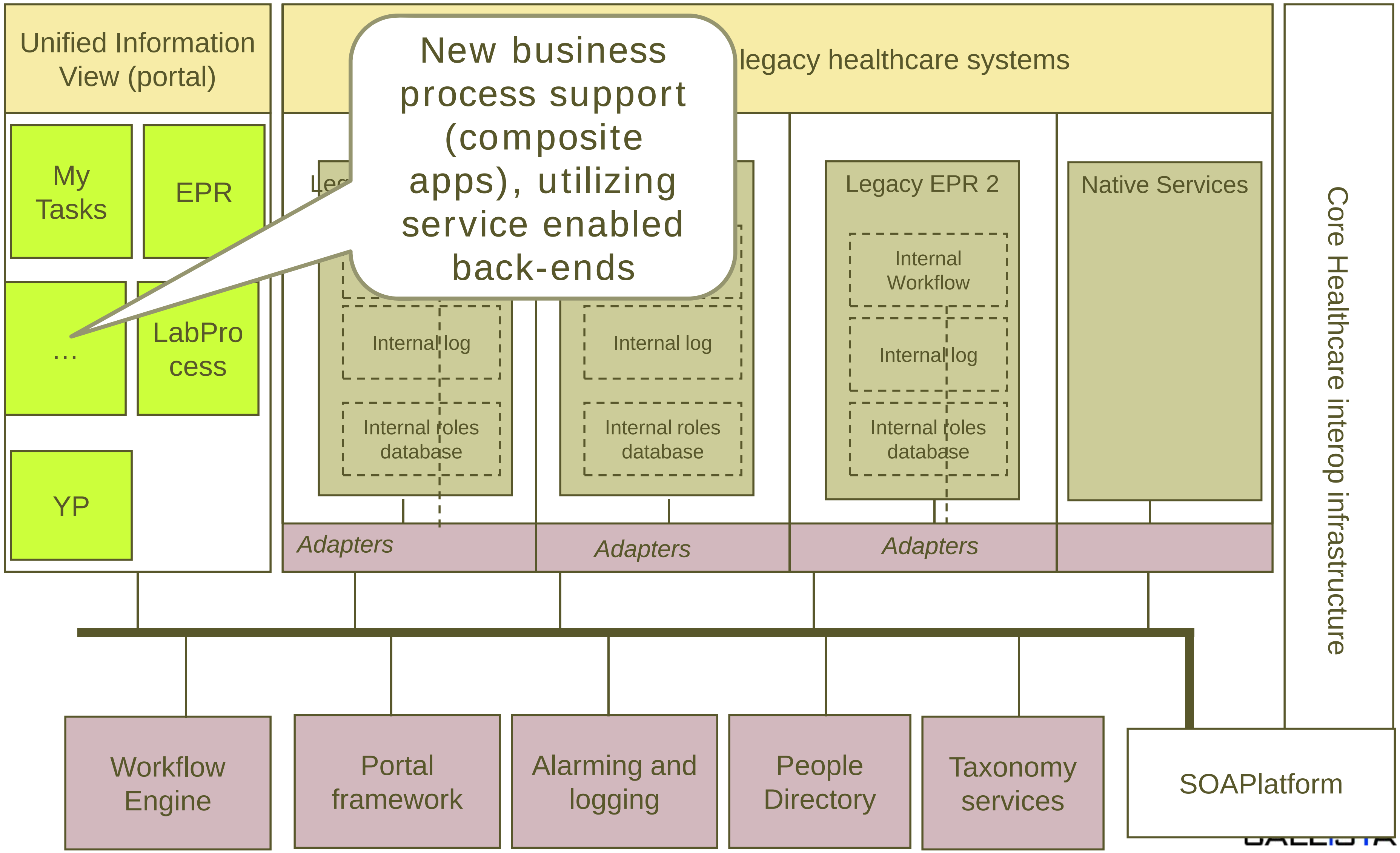


HÄLSO- OCH SJUKVÅRDSPORTAL

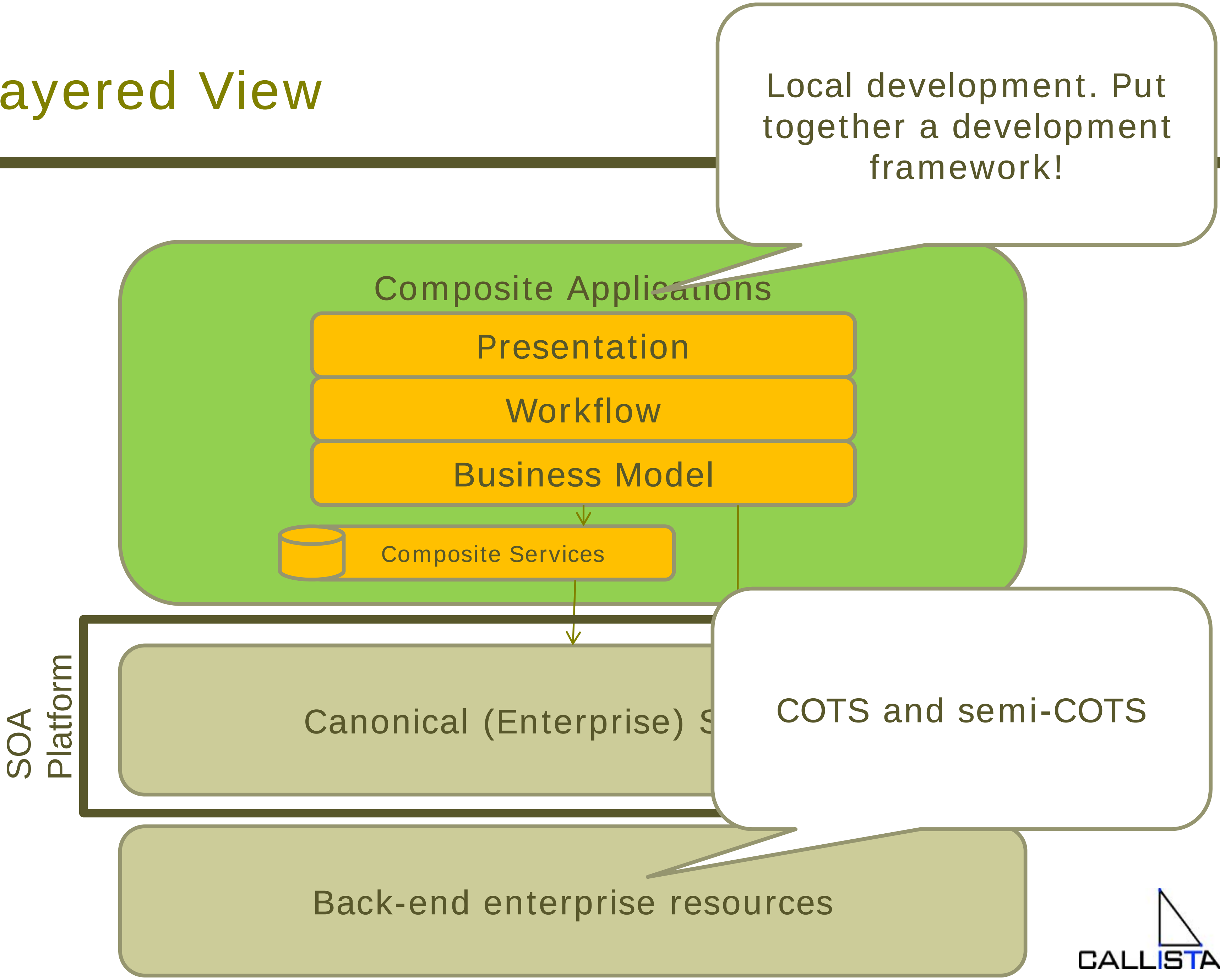
Patienten ska i framtiden ha ett eget informationsfönster, en egen portal, till hälso- och sjukvården. Kravspecifikation och beslutsunderlag arbetas fram för funktioner som patientens egen vård, sjukvårdsrådgivning, vårdkatalog, information om tillgänglig sjukvård, tidbokning och tillgång till journaldata. Pilotprojekt startar 2006.

…

IT Architecture for VGRegion Healthcare

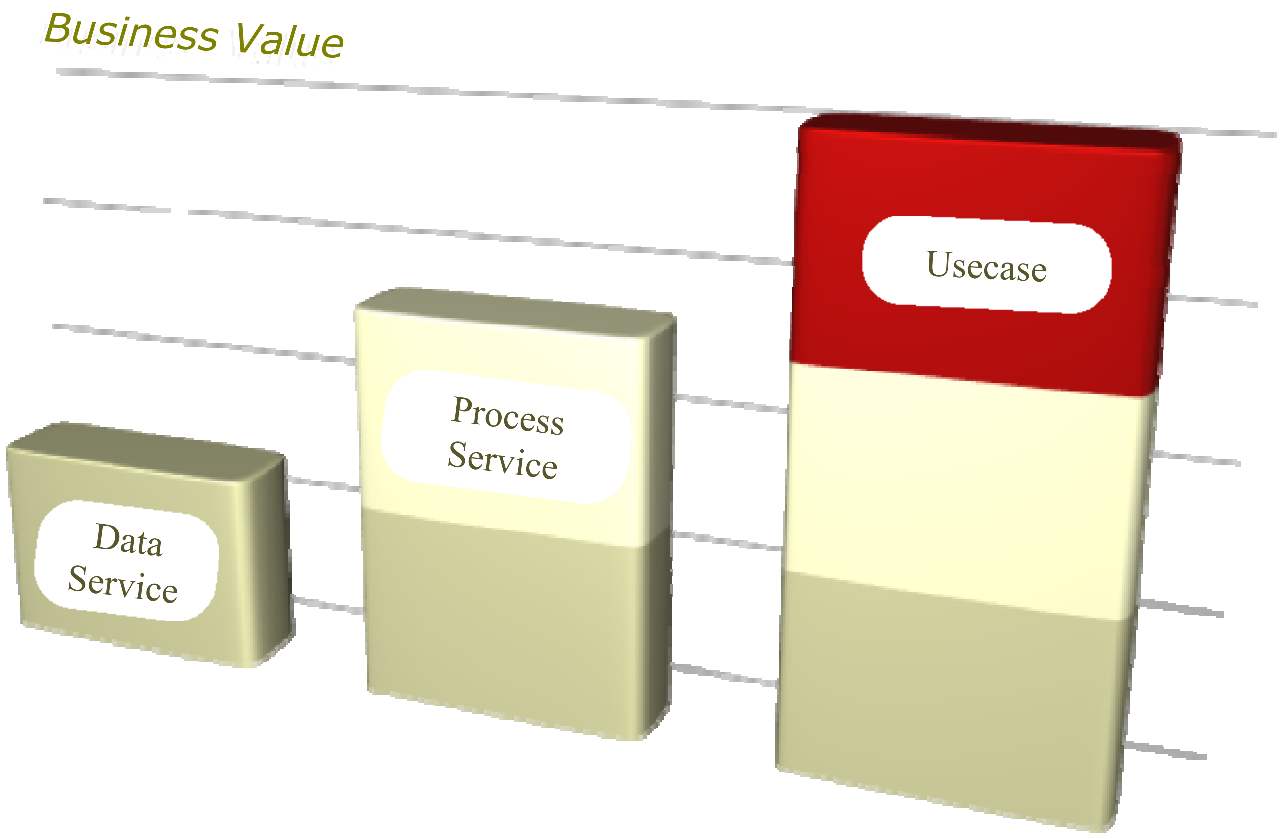


Layered View



CALLISTA

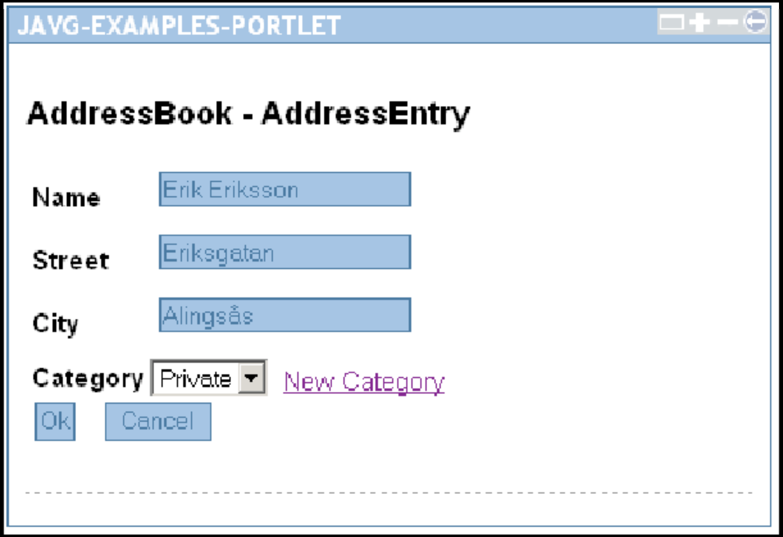
Requirement: “Use-Case-as-service”



CALLISTA

Requirement: Portability

Portlet

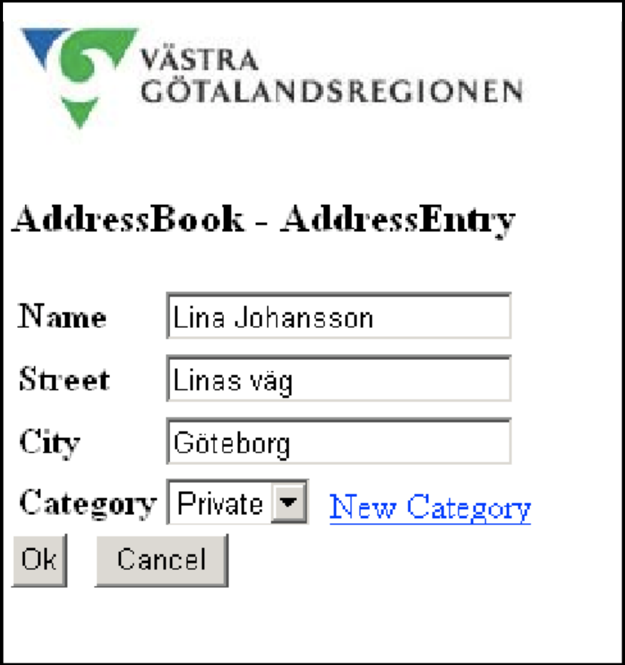


JAVG-EXAMPLES-PORTLET

AddressBook - AddressEntry

Name	Erik Eriksson
Street	Eriksgatan
City	Alingsås
Category	Private

Web-app



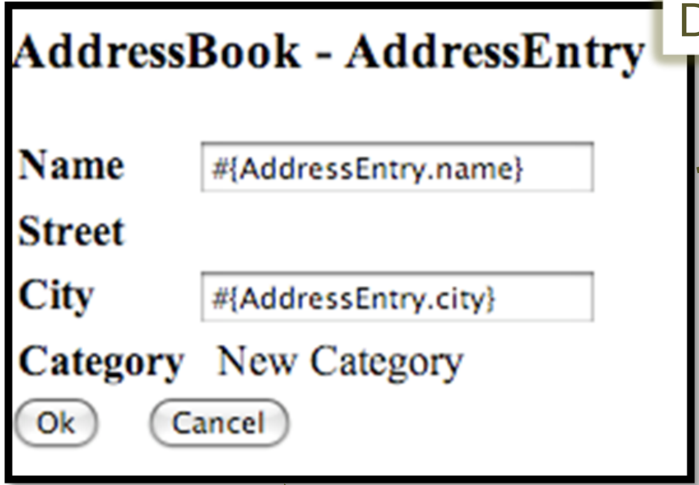
AddressBook - AddressEntry

Name	Lina Johansson
Street	Linas väg
City	Göteborg
Category	Private

Container portability (Java EE)
Portability across Web app and Portlet

CALLISTA

Requirement: Web designer friendly templating technology

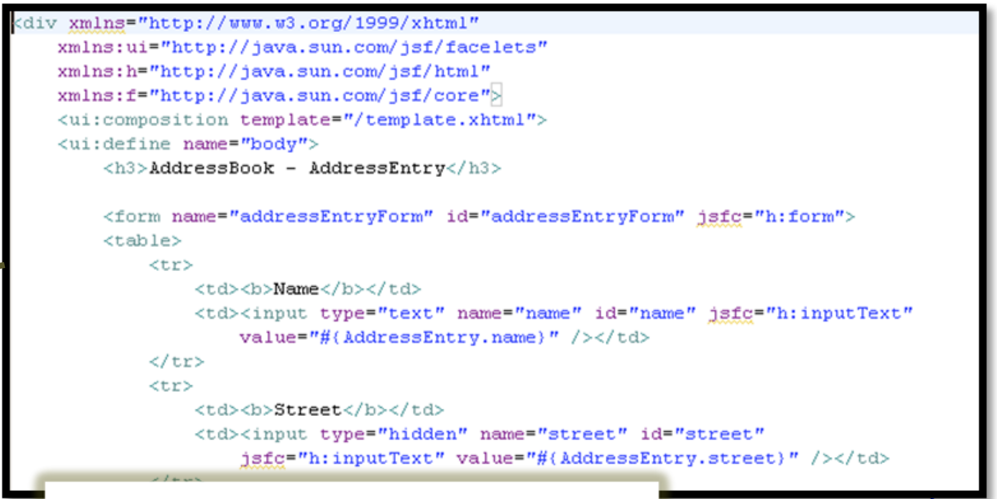


Design in DreamWeaver

AddressBook - AddressEntry

Name	#{AddressEntry.name}
Street	
City	#{AddressEntry.city}
Category	New Category

Ok Cancel



```
<div xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
    <ui:composition template="/template.xhtml">
    <ui:define name="body">
        <h3>AddressBook - AddressEntry</h3>

        <form name="addressEntryForm" id="addressEntryForm" jsfc="h:form">
        <table>
            <tr>
                <td><b>Name</b></td>
                <td><input type="text" name="name" id="name" jsfc="h:inputText"
                    value="#{AddressEntry.name}" /></td>
            </tr>
            <tr>
                <td><b>Street</b></td>
                <td><input type="hidden" name="street" id="street"
                    jsfc="h:inputText" value="#{AddressEntry.street}" /></td>
```

Development in XML editor

CALLISTA

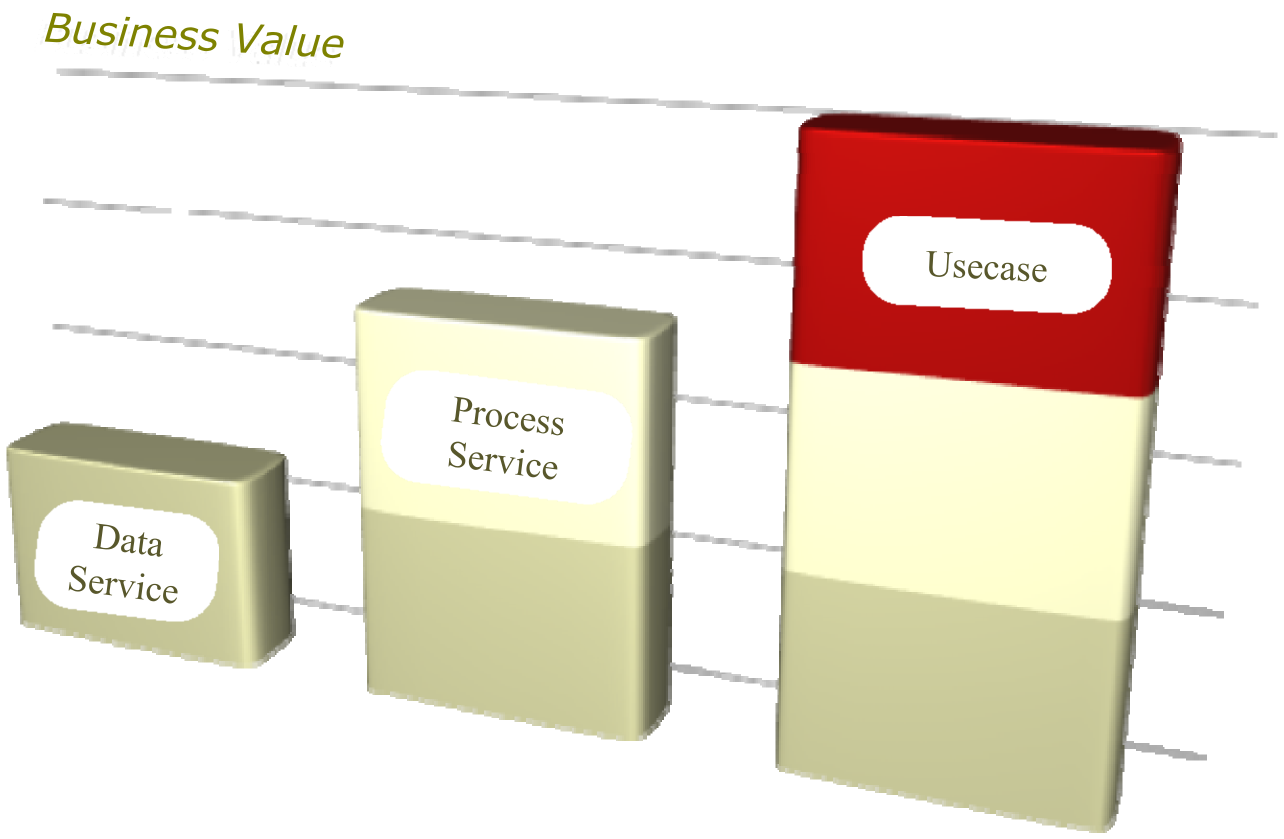
Requirement: Support Verva web design requirements

- Graceful degradation
  - Always functional
  - Usability proportional to browser capabilities
- Example
  - JavaScript disabled

Requirements: Agile Development environment

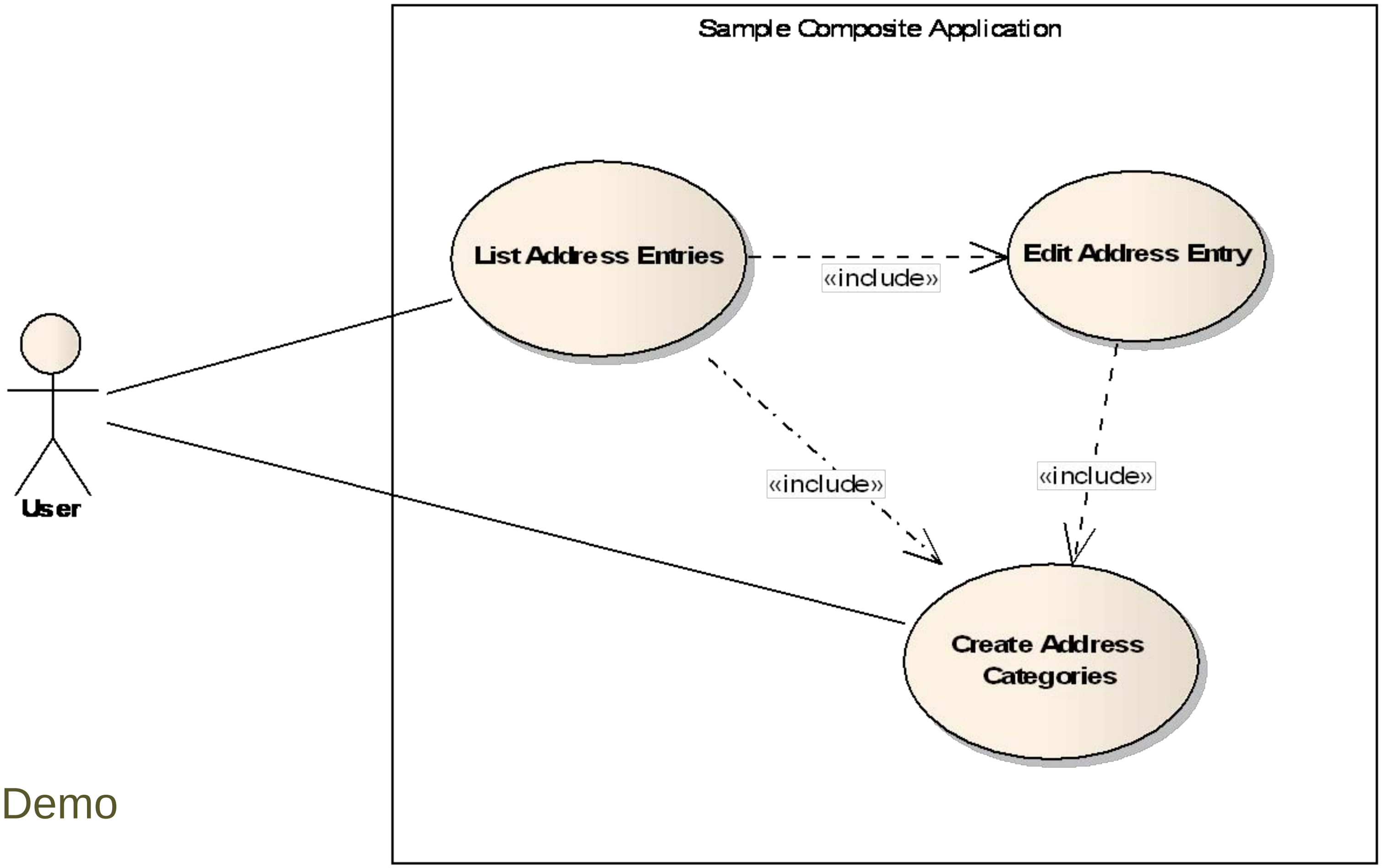
- No IDE / Developer set-up lock-in
  - Portable builds
  - IDE settings / projects generated from build files (“build file is master”)
- Reasonable hardware requirements
  - Run well in VMWare on mid-range laptop
- Broad, practical availability of tooling
  - Consultant-friendly
- Fast code-test-debug cycle
  - Do not require deploy in WebSphere Portal for developer testing
  - “Remove” inherent web programming issues / cost drivers

Major Challenge: Re-use at Use-Case-level



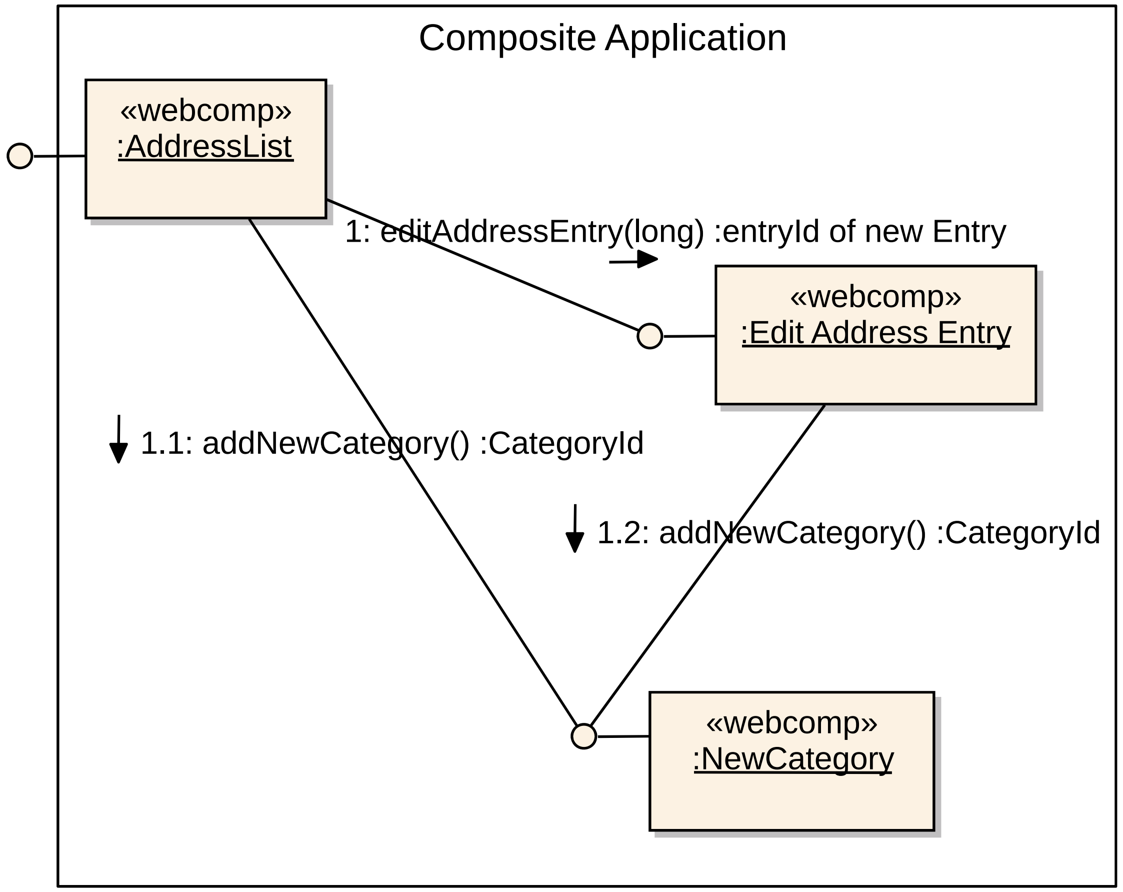
CALLISTA

How will Use-cases become components?



Demo

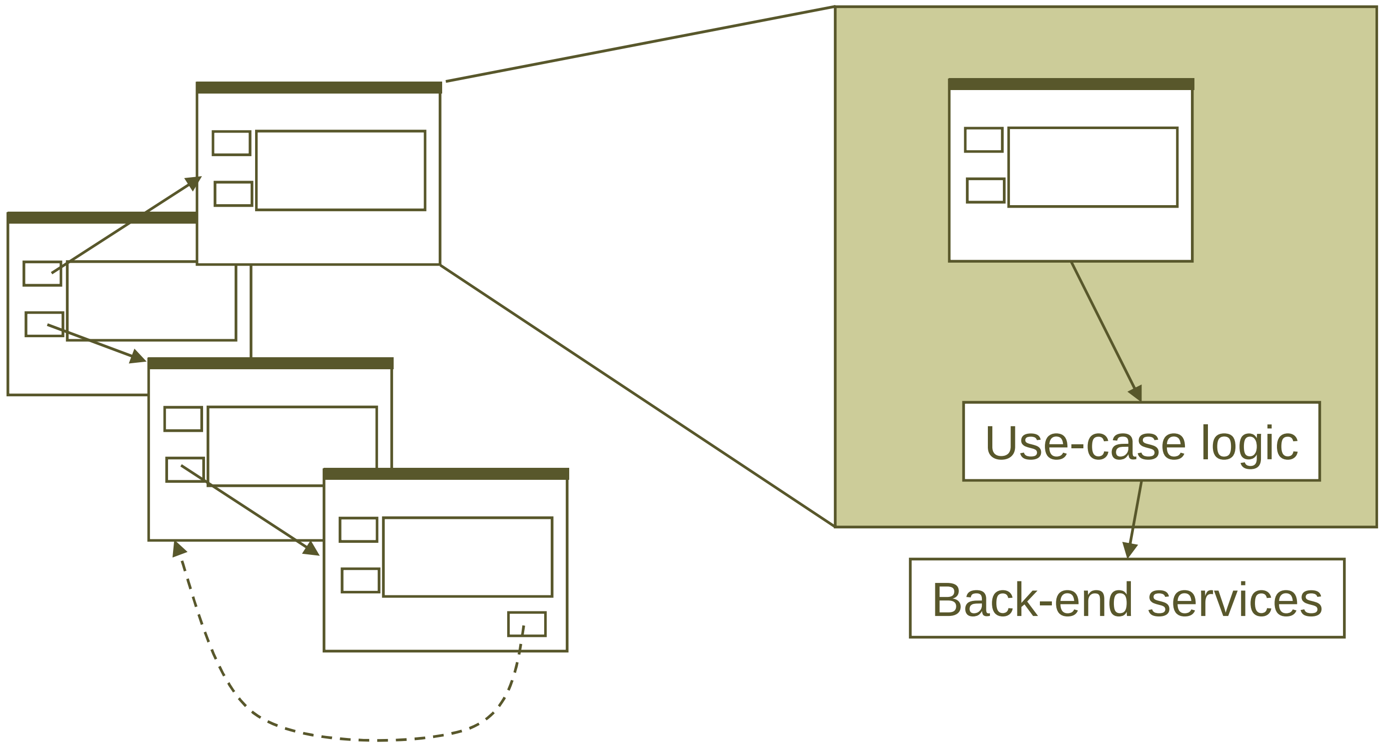
Use-cases as Components


Composite Application
«webcomp»
:AddressList
1: editAddressEntry(long) :entryId of new Entry
«webcomp»
:Edit Address Entry
1.1: addNewCategory() :CategoryId
1.2: addNewCategory() :CategoryId
«webcomp»
:NewCategory

Semantically equivalent to calling modal dialog in Swing!

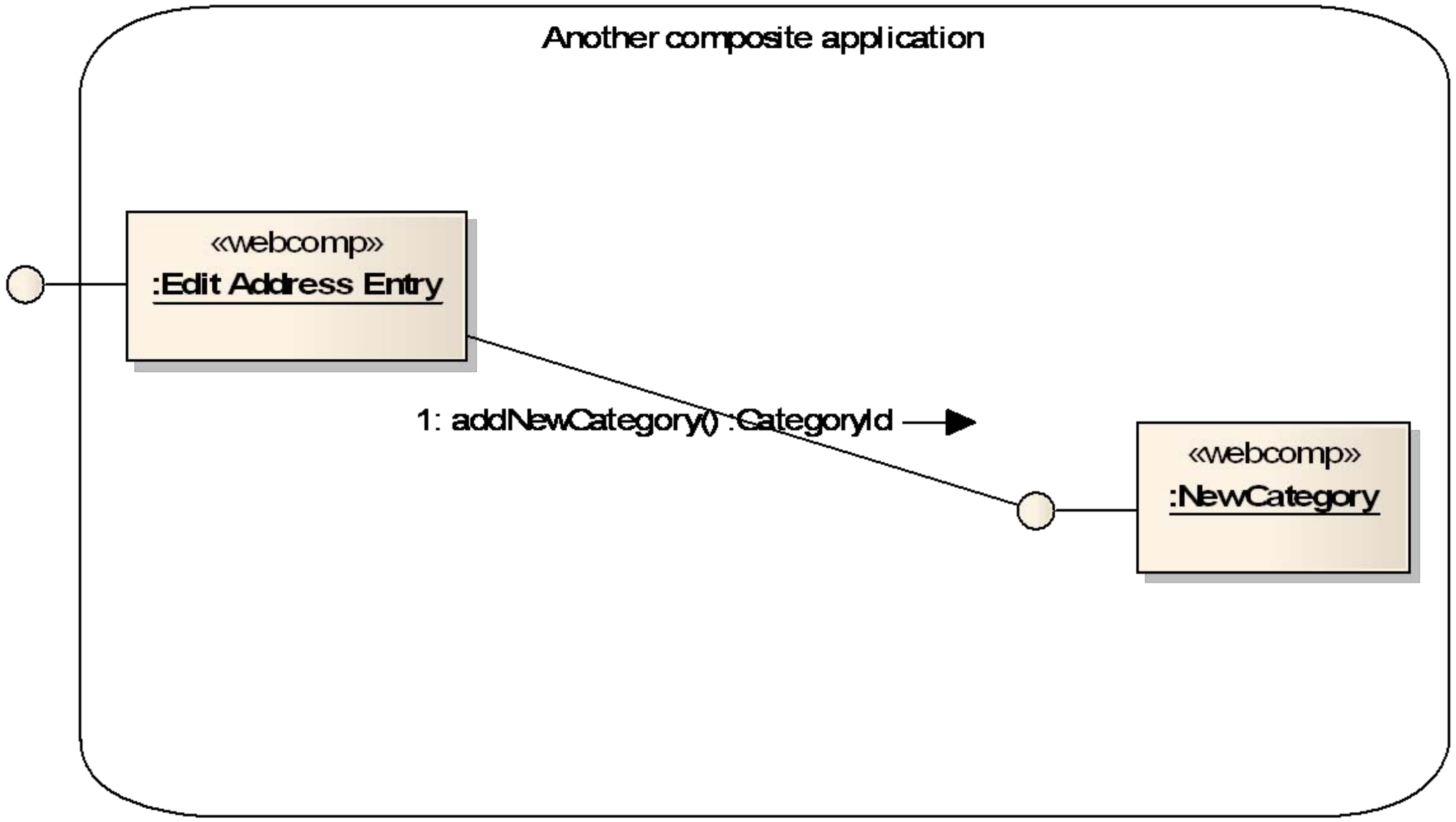
Web realization versus Rich Client

- Think ”Modal sub dialogs”
- Each dialog is tied to back-end logic



CALLISTA

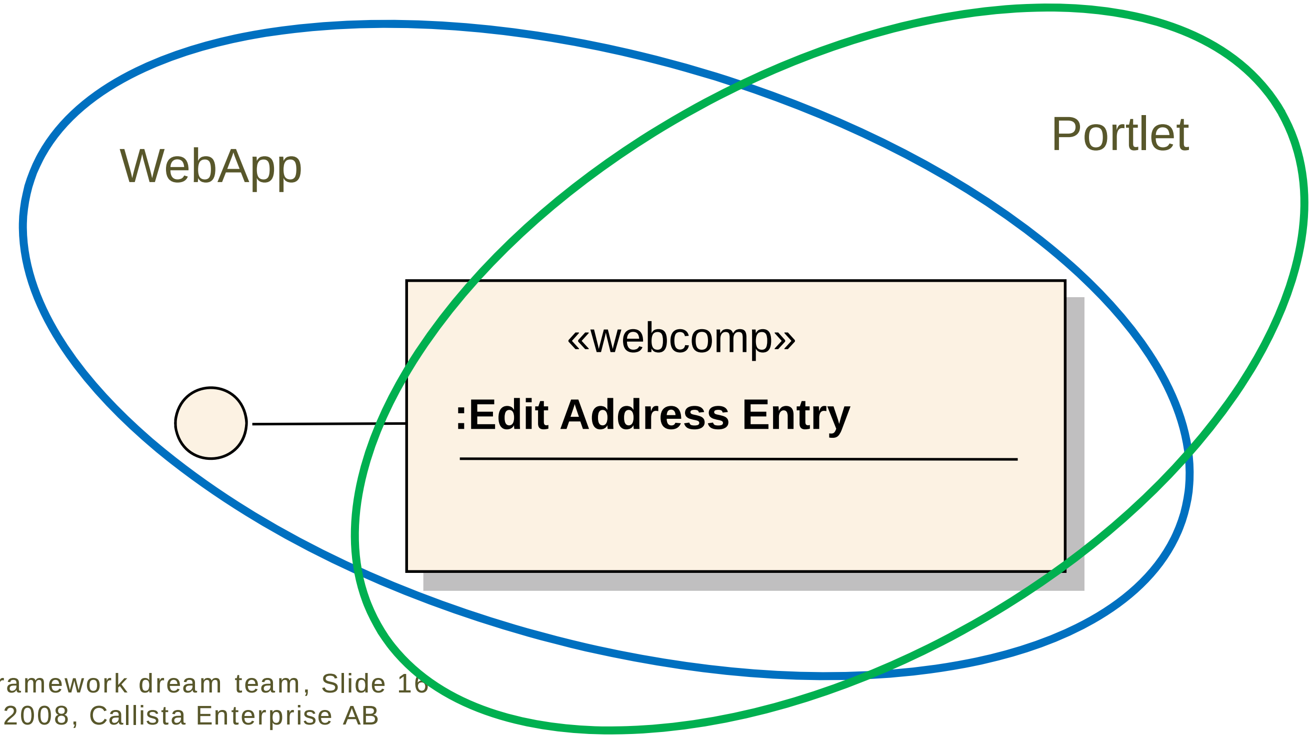
Re-use in another composite application


Another composite application
«webcomp»
:Edit Address Entry
1: addNewCategory() :CategoryId
«webcomp»
:NewCategory

CALLISTA

Adding portability...

- WEB-INF artifacts are not portable - nor composable
  - We can’t depend on JSPs
  - A use-case web component will have to be a jar files that works on WEB-INF/lib whether portlet or webapp (formulera om)



“Remove” inherent web programming issues

0. Get a random number between 0 and 100
1. Show the guess-form
2. Repeat until successful guess:
2.1 The user enters a value and submits the form
2.2. Validate the guess. I error (not between 0 and 100), continue at 2.
2.3. Unless successful guess, show whether the guess was under or above the answer.
2.4. Display guess-form.
3.0 Show congratulations screen with number of guesses.

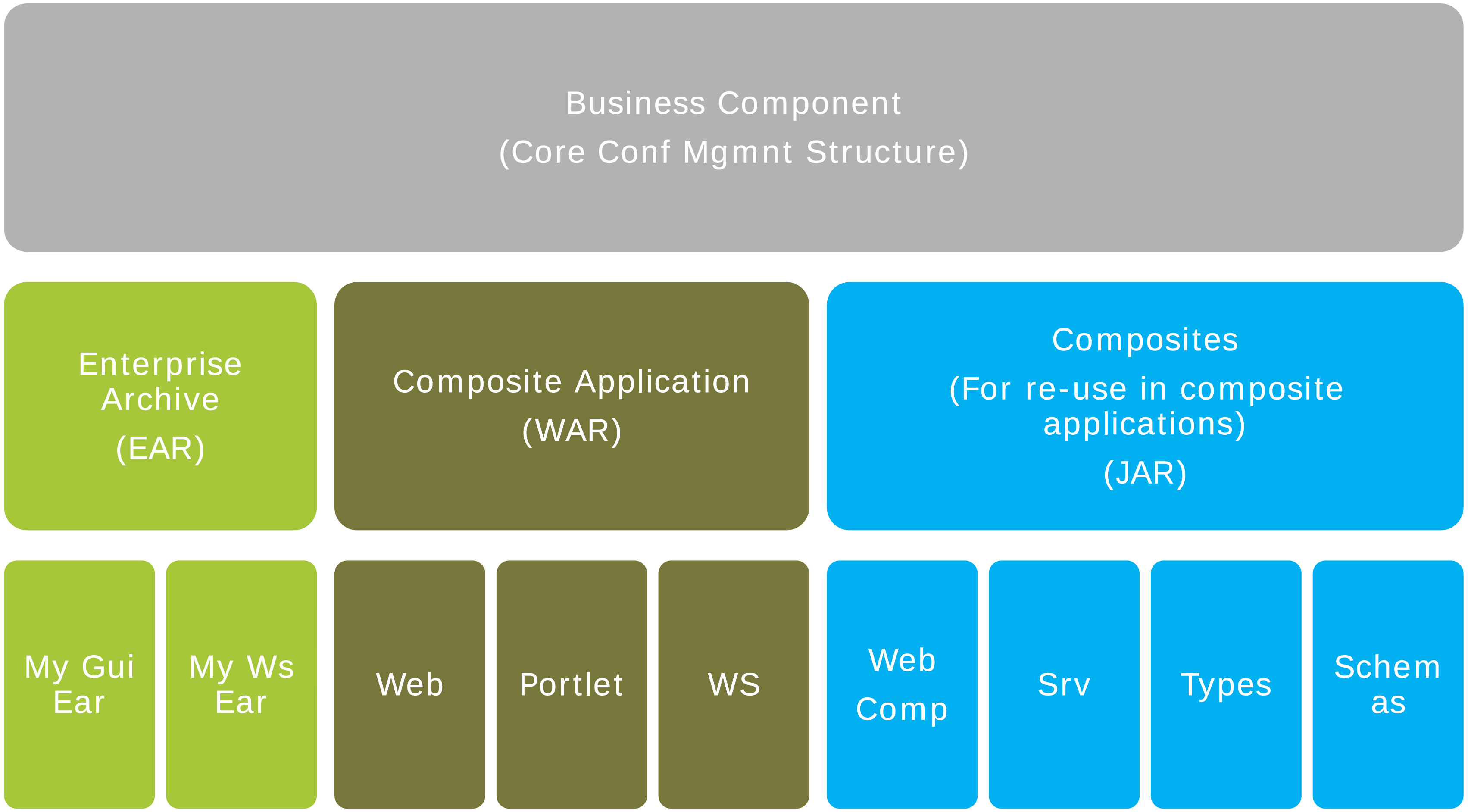
```
public class Game extends Element {
        private static Random randomNumbers = new Random();
        public void processElement() {

                Template template = getHtmlTemplate("game");
                int answer = 0, guesses = 0, guess = -1;

                answer = randomNumbers.nextInt(101);
                while (guess != answer) {
                        print(template);
                        pause();
                        template.clear();
                        guess = getParameterInt("guess", -1);
                        if (guess < 0 || guess > 100) {
                                template.setBlock("warning", "invalid");
                                continue;
                        }
                        guesses++;
                        if (answer < guess)     template.setBlock("msg", "lowe
                        else if (answer > guess) template.setBlock("msg", "hig
                }
                ContinuationContext.getActiveContext().removeContextTree();
                template = getHtmlTemplate("success");
                template.setValue("answer", answer);
                template.setValue("guesses", guesses);
                print(template);
        }
}
```

Reduce semantic gap: Flow of events

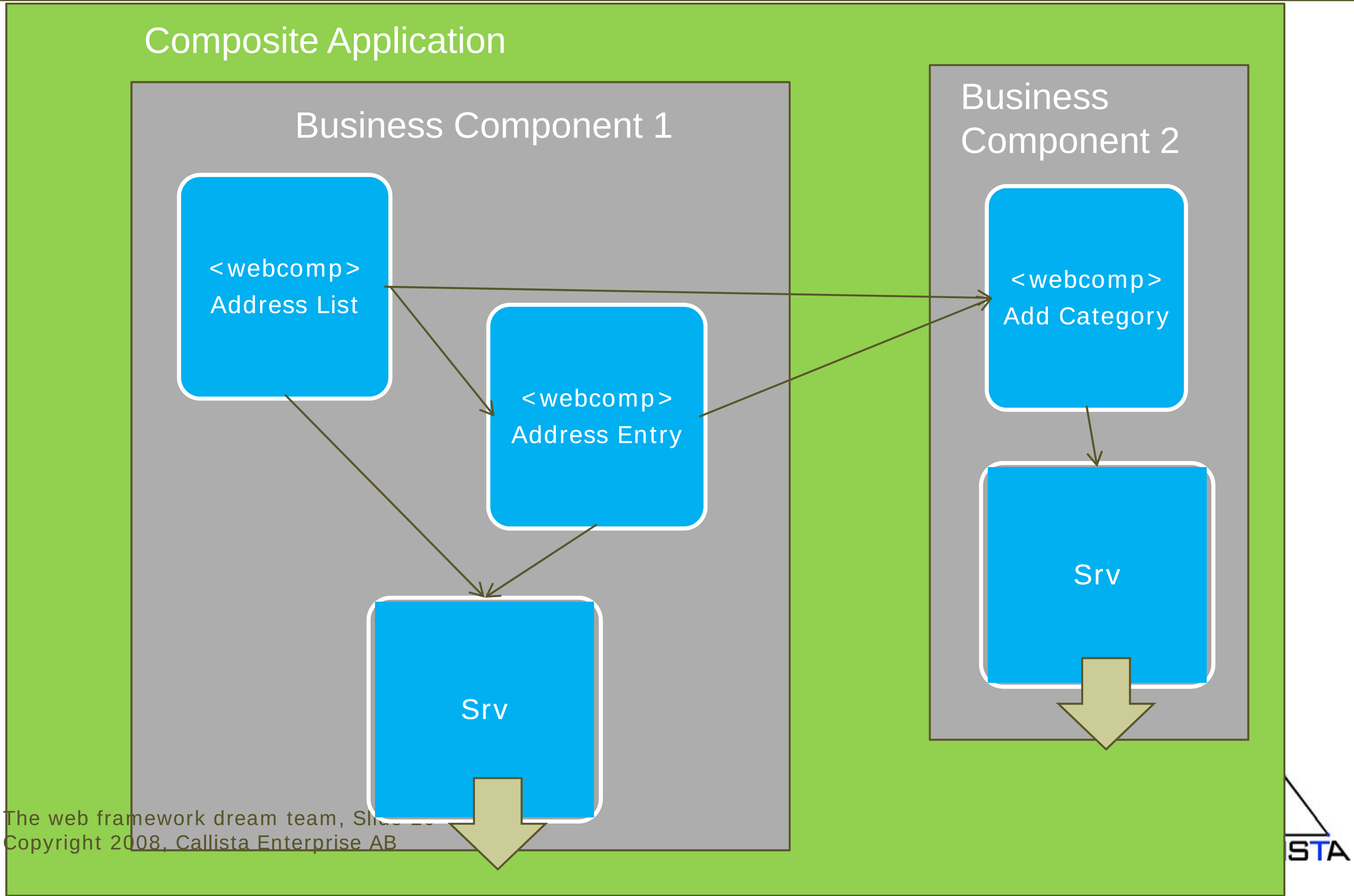
- Continuations
  - Code flow mirrors use-case flow
- Supporting technologies
  - Code in Java
    - The RIFE framework
  - Code in XML
    - Spring WebFlow

CALLISTA

The resulting reference architecture



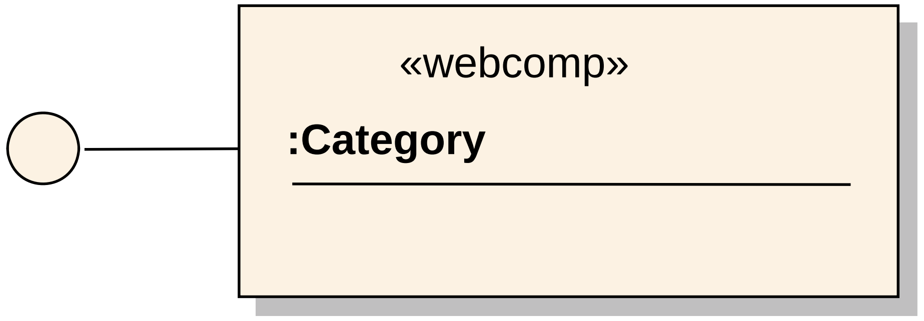
Dynamic view (runtime perspective)

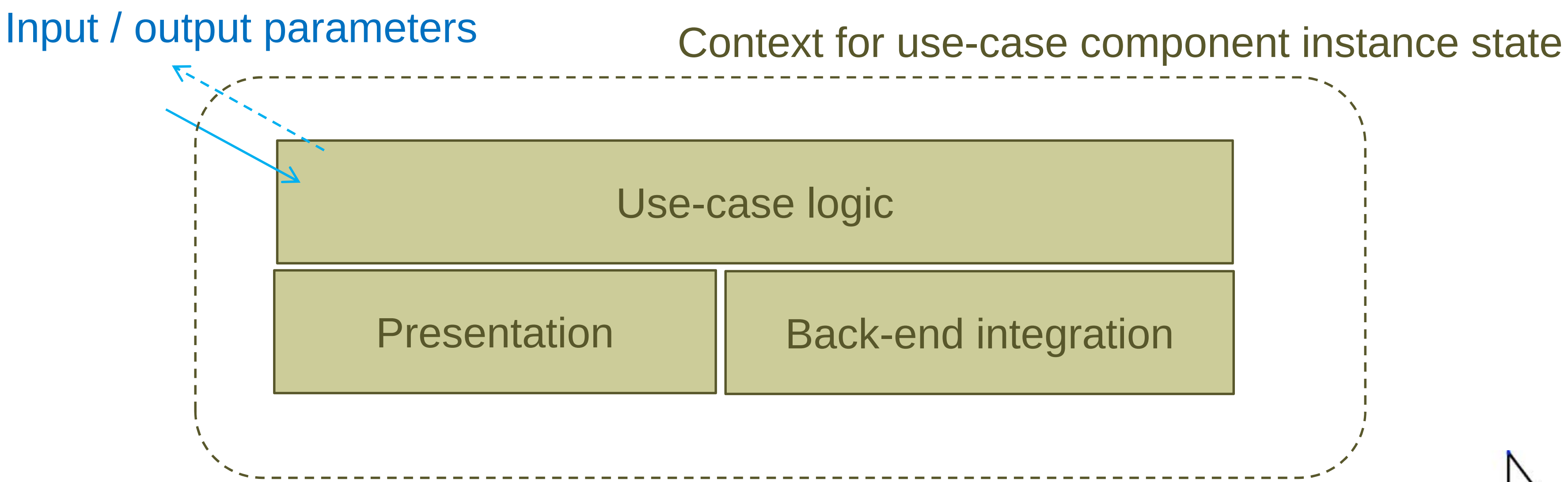


Selected frameworks

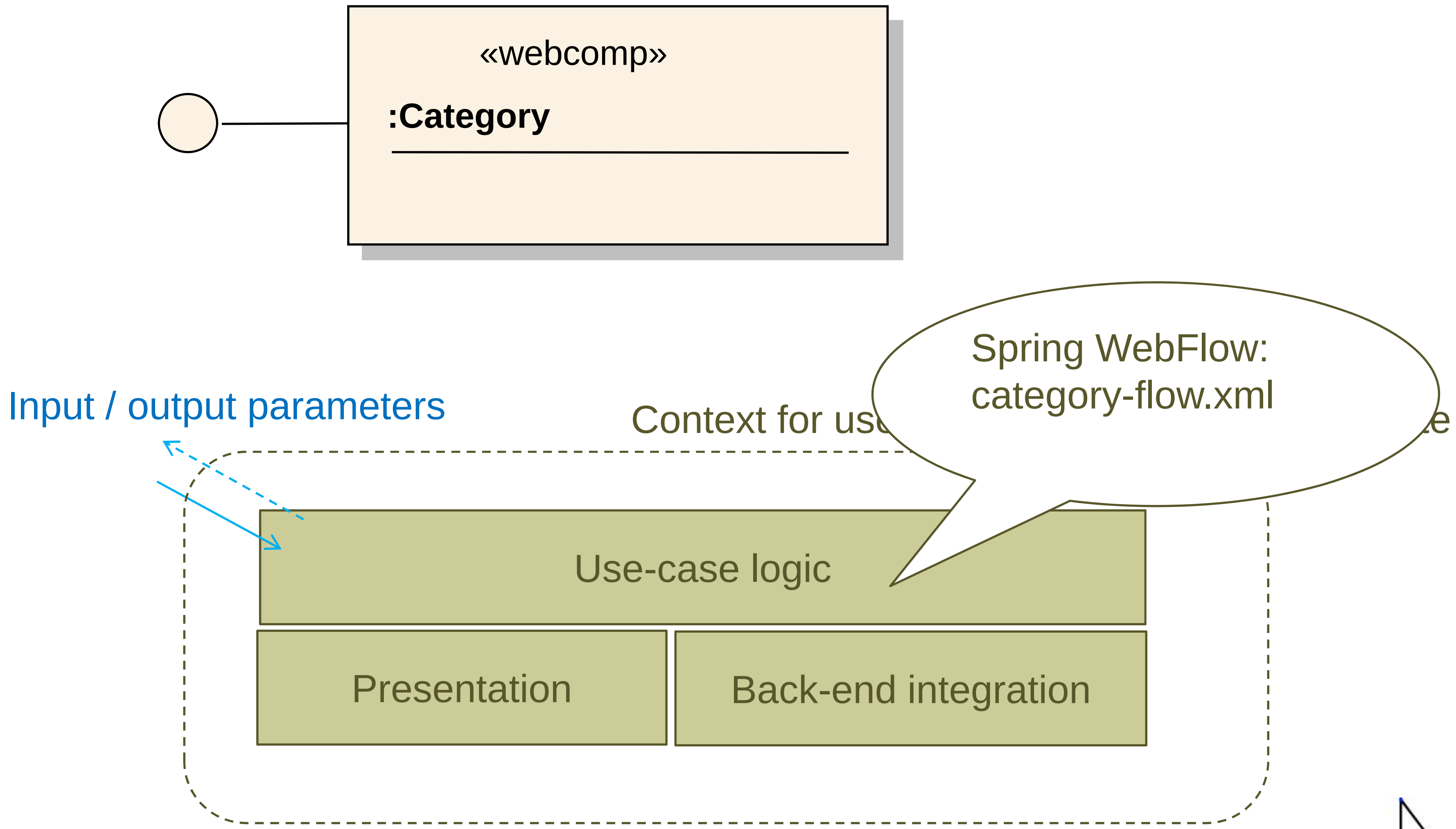
- Core requirements
  - Complete workflow in composable archives (jar files)
  - Empty web-inf (breaking the monolith web app)
  - Continuations (remove semantic gap)
- Spring WebFlow – Use-case logic
  - Clean separation of workflow logic
  - Powerful, expressive, simple, continuation-based
- Facelets - Presentation
  - None-intrusive to html
  - Supports jar packaging!
  - Made for JSF (unlike JSP)
- JSF – Request processing
  - Standard and does support jar-packaging
  - "Hidden" by Spring WebFlow

Anatomy of a web use-case component (web composite)

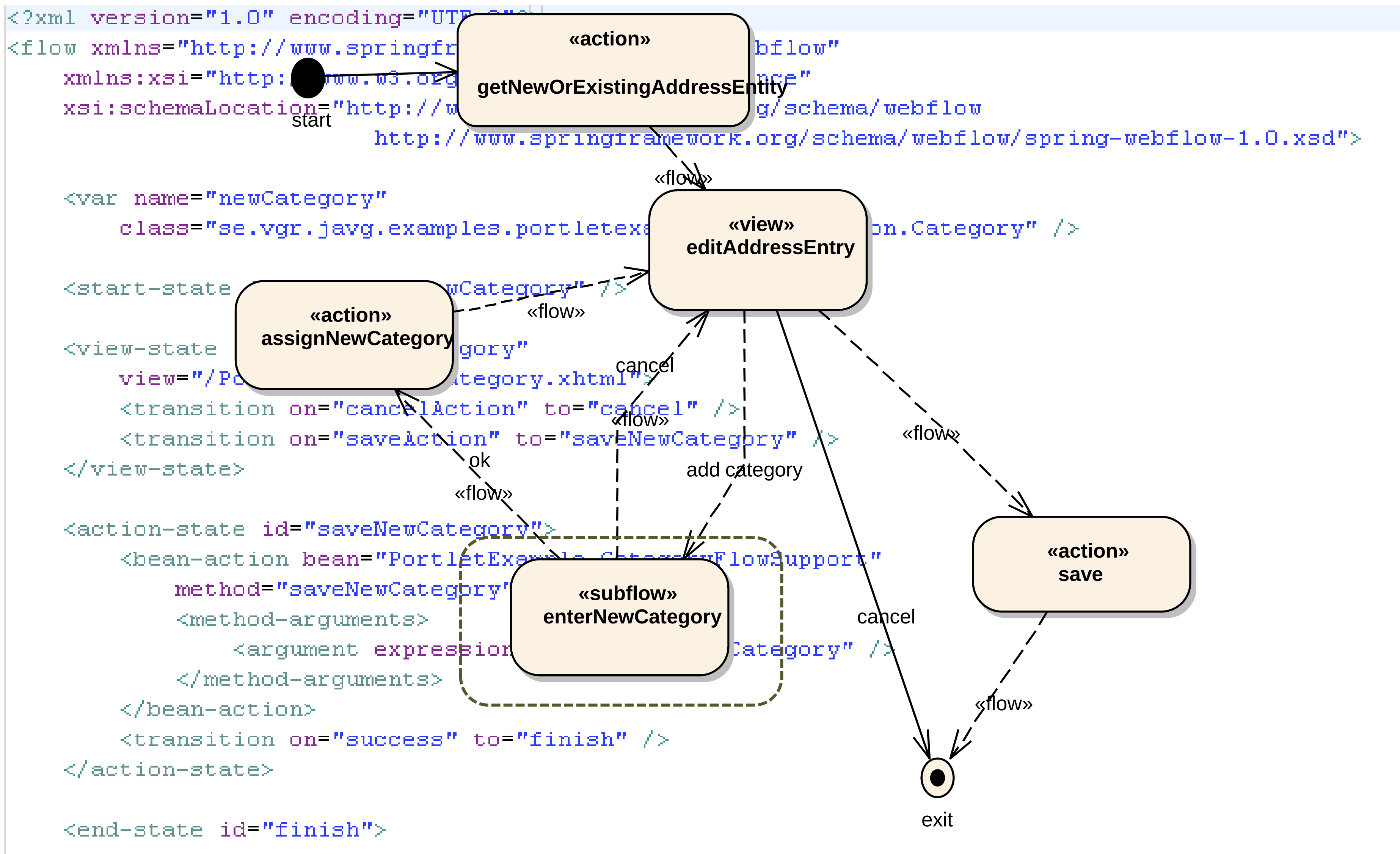



Input / output parameters
Context for use-case component instance state
Use-case logic
Presentation
Back-end integration

Anatomy of a web use-case component (web composite)

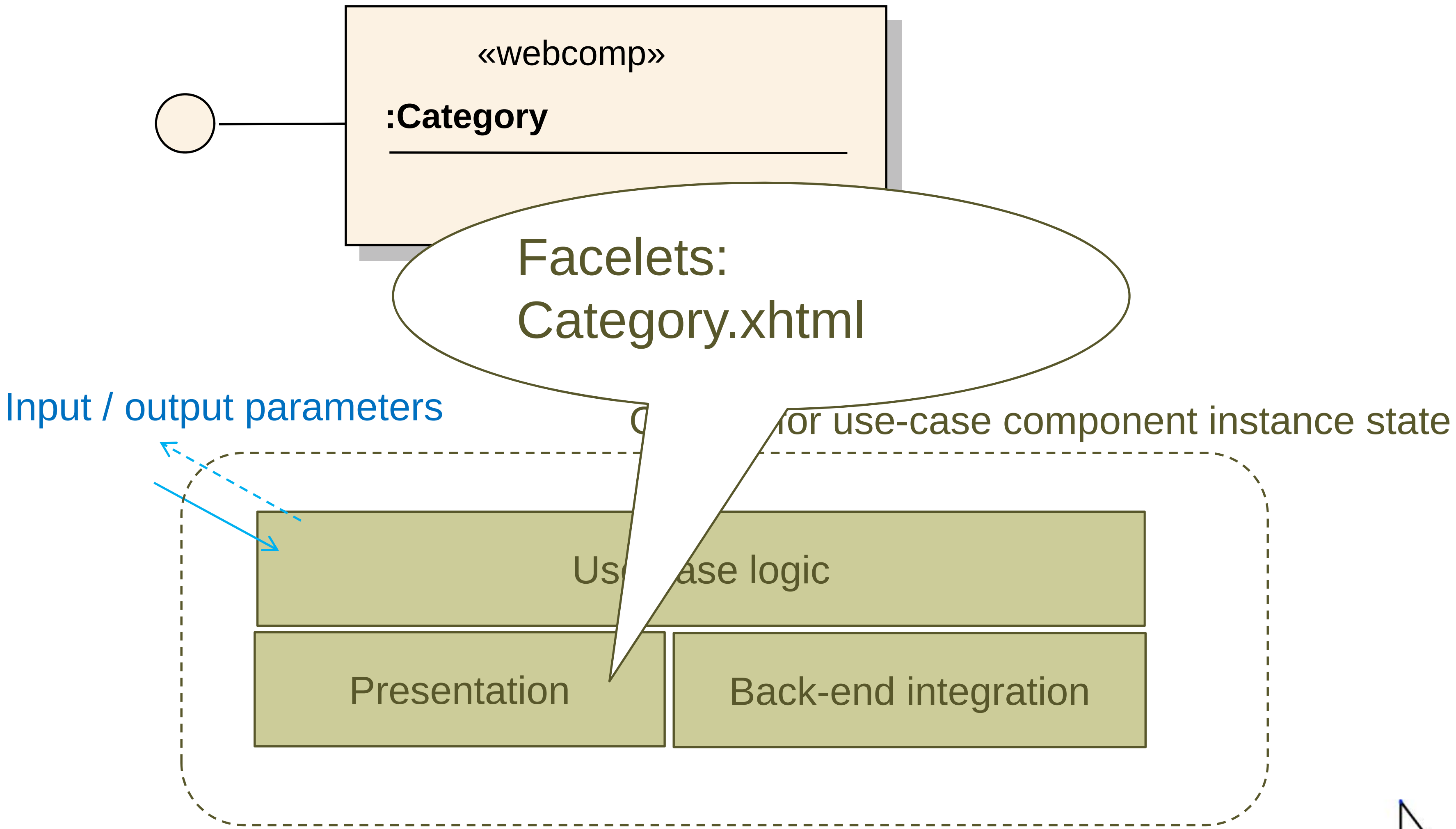


WebFlow – Sample – AddressEntryComposite



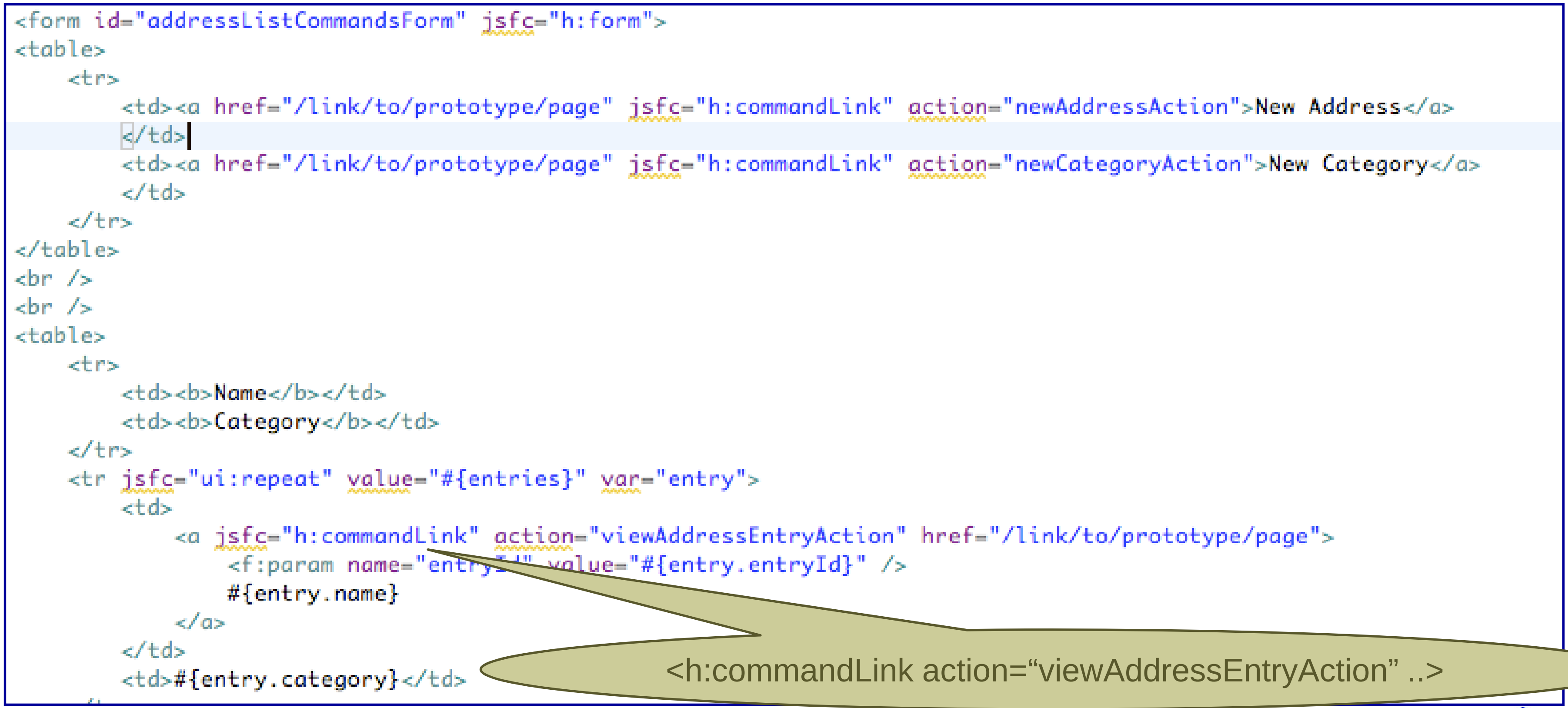
```
<?xml version="1.0" encoding="UTF
<flow xmlns="http://www.springfr                          bflow"
    xmlns:xsi="http:/  www.w3.org                         nce"
    xsi:schemaLocation="http://w                          g/schema/webflow
                 http://www.springframework.org/schema/webflow/spring-webflow-1.0.xsd">

    <var name="newCategory"
        class="se.vgr.javg.examples.portletexa                 on.Category" />

    <start-state                 wCategory" />

    <view-state                   gory"
        view="/Po                 tegory.xhtml">
        <transition on="cancelAction" to="cancel" />
        <transition on="saveAction" to="saveNewCategory" />
    </view-state>

    <action-state id="saveNewCategory">
        <bean-action bean="PortletExa           CategoryFlowSupport"
            method="saveNewCategory"
            <method-arguments>
                <argument expression                 Category" />
            </method-arguments>
        </bean-action>
        <transition on="success" to="finish" />
    </action-state>

    <end-state id="finish">
```

Anatomy of a web use-case component (web composite)



Facelets– Sample – Category Composite

- None-intrusive to html
- Supports jar packaging (loading of views from class-path)

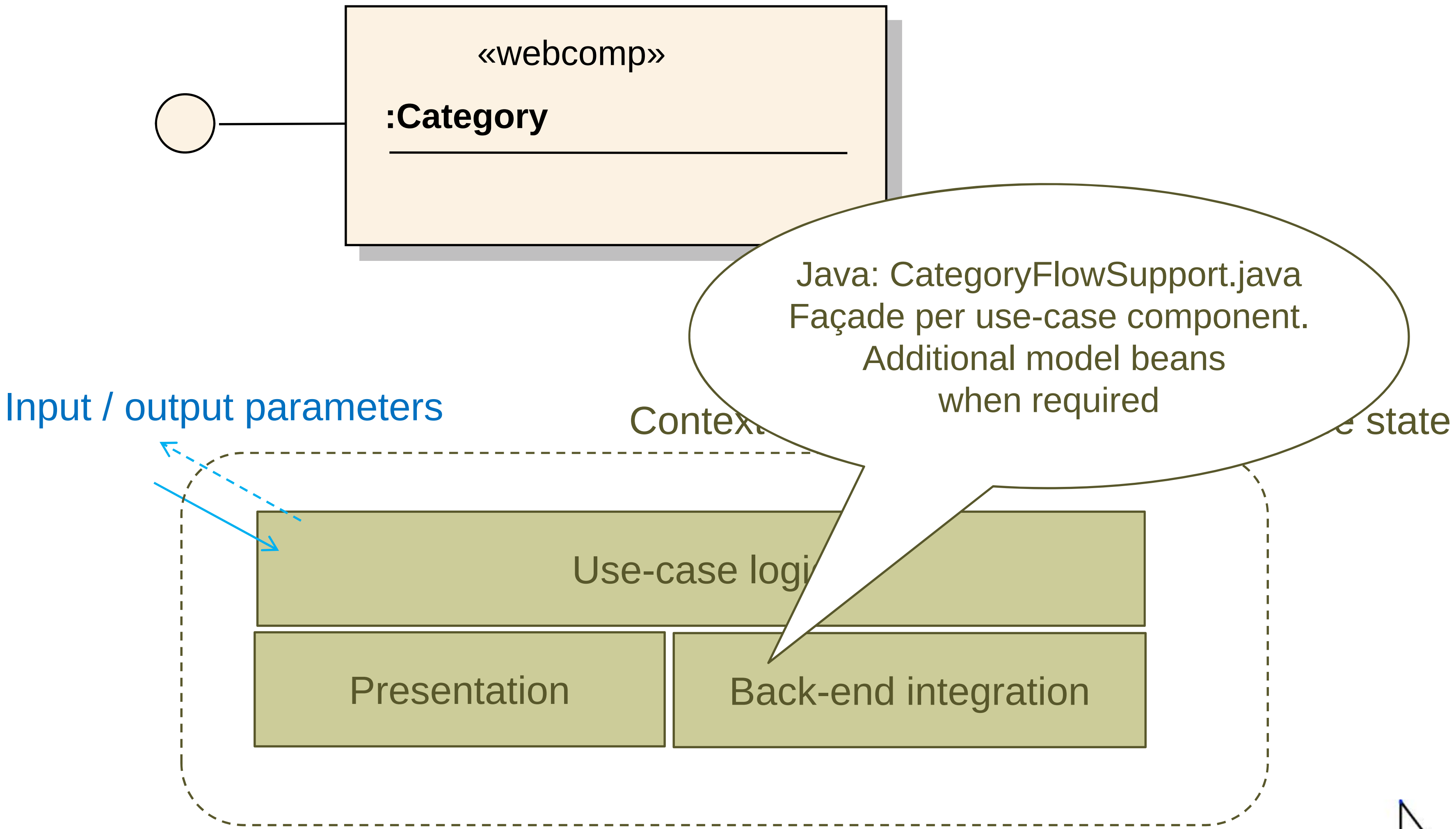


```
<form id="addressListCommandsForm" jsfc="h:form">
<table>
    <tr>
        <td><a href="/link/to/prototype/page" jsfc="h:commandLink" action="newAddressAction">New Address</a>
        </td>
        <td><a href="/link/to/prototype/page" jsfc="h:commandLink" action="newCategoryAction">New Category</a>
        </td>
    </tr>
</table>
<br />
<br />
<table>
    <tr>
        <td><b>Name</b></td>
        <td><b>Category</b></td>
    </tr>
    <tr jsfc="ui:repeat" value="#{entries}" var="entry">
        <td>
            <a jsfc="h:commandLink" action="viewAddressEntryAction" href="/link/to/prototype/page">
                <f:param name="entryId" value="#{entry.entryId}" />
                #{entry.name}
            </a>
        </td>
        <td>#{entry.category}</td>
```

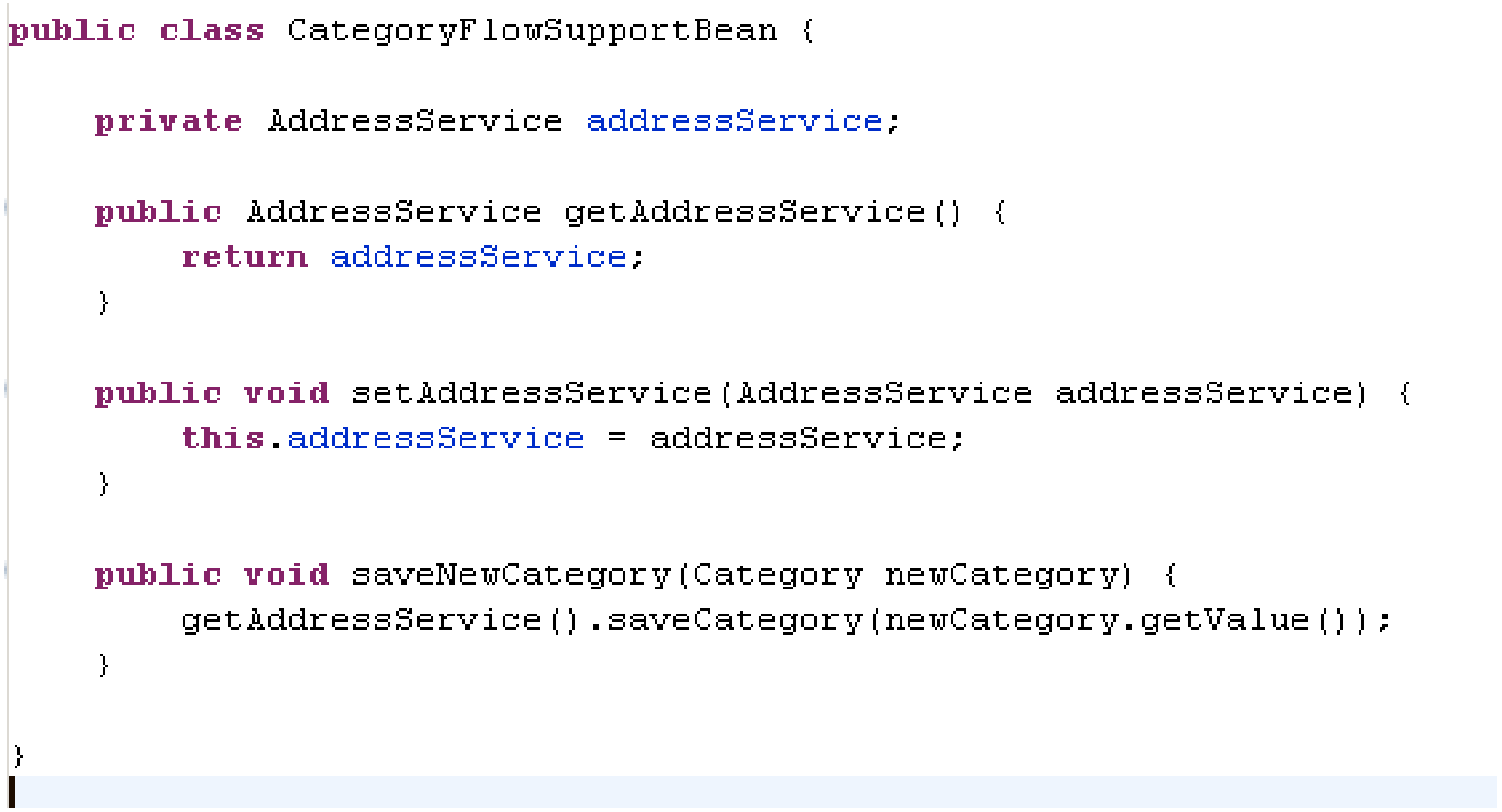
<h:commandLink action="viewAddressEntryAction" ..>

CALLISTA

Anatomy of a web use-case component (web composite)



Use-case façade



```
public class CategoryFlowSupportBean {

    private AddressService addressService;

    public AddressService getAddressService() {
        return addressService;
    }

    public void setAddressService(AddressService addressService) {
        this.addressService = addressService;
    }

    public void saveNewCategory(Category newCategory) {
        getAddressService().saveCategory(newCategory.getValue());
    }

}
```

But what about JSF?

- Just for bootstrapping Spring Webflow
- Generic faces-config in web-app
- No footprint in web composites

```
<?xml version="1.0"?>
<!DOCTYPE faces-config PUBLIC
  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">

<faces-config>
		<application>
				<navigation-handler>
						org.springframework.webflow.executor.jsf.FlowNavigationHandler
				</navigation-handler>
				<variable-resolver>
						org.springframework.webflow.executor.jsf.DelegatingFlowVariableResolver
				</variable-resolver>
				<view-handler>com.sun.facelets.FaceletViewHandler</view-handler>
		</application>

		<lifecycle>
				<phase-listener>
						org.springframework.webflow.executor.jsf.FlowPhaseListener
				</phase-listener>
		</lifecycle>
</faces-config>JSF
```

Environments

- Development
  - Apache Tomcat with Apache Pluto
  - Eclipse 3.3 (Europa) with WTP
- Deployment
  - WebSphere Application Server 6.1
  - WebSphere Portal Server 6.0
- Build system
    - Maven 2 – manages the component architecture
    - Generates Eclipse dependencies and project files

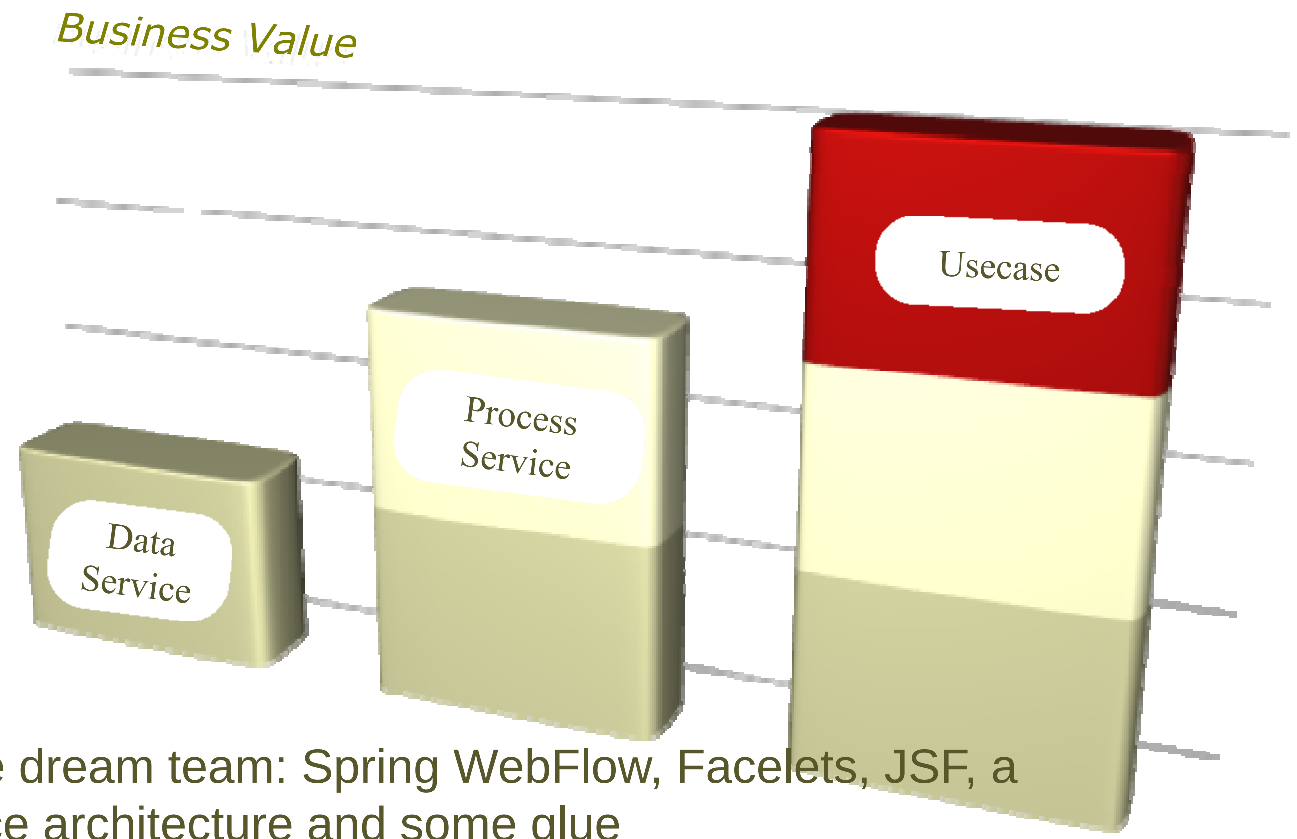
CALLISTA

Lessons learned

- Webflow
  - Revolution for html centric web applications
  - The foundation for use-case componentization
  - WebFlow validation framework tied into SpringMVC
    - On the agenda for next version…
- Facelets
  - Momentum depends on Seam
  - Servlet required for loading web resources (images etc) from jar files (web composites)
  - Intuitive
- Portability
  - Native JSF implementations are not fully portable. We ended up bundling MyFaces rather than deploying to WebSphere JSF implementation (Sun RI)

CALLISTA

Summary
Re-usable Use-case Components Delivered!



…by the dream team: Spring WebFlow, Facelets, JSF, a reference architecture and some glue

CALLISTA

Time (?) for Questions!