

Rules Based Web Applications Development

Developing a highly dynamic web application for a large bank using rules-based technology

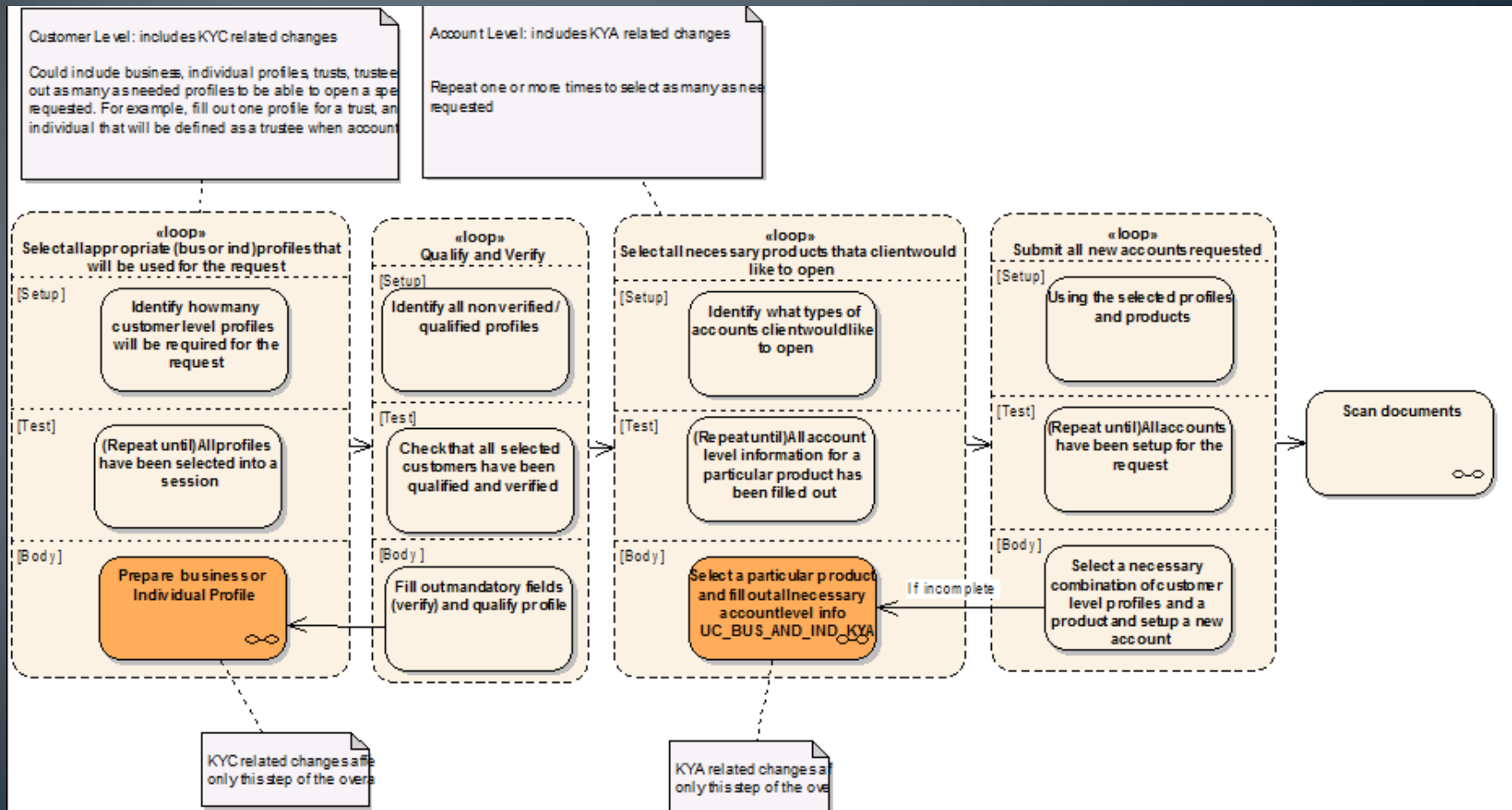
Topics

- Part 1: Requirements and zooming in on a solution
- Part 2: Design and development
- Part 3: Conclusion and lessons learned

Part 1: Requirements and zooming in on a development approach

- Requirements & Reference Architecture
- Changing directions
- Product / Vendor Options
- Selected Product and Reasons Why

As is: the original onboarding Process



New Requirements

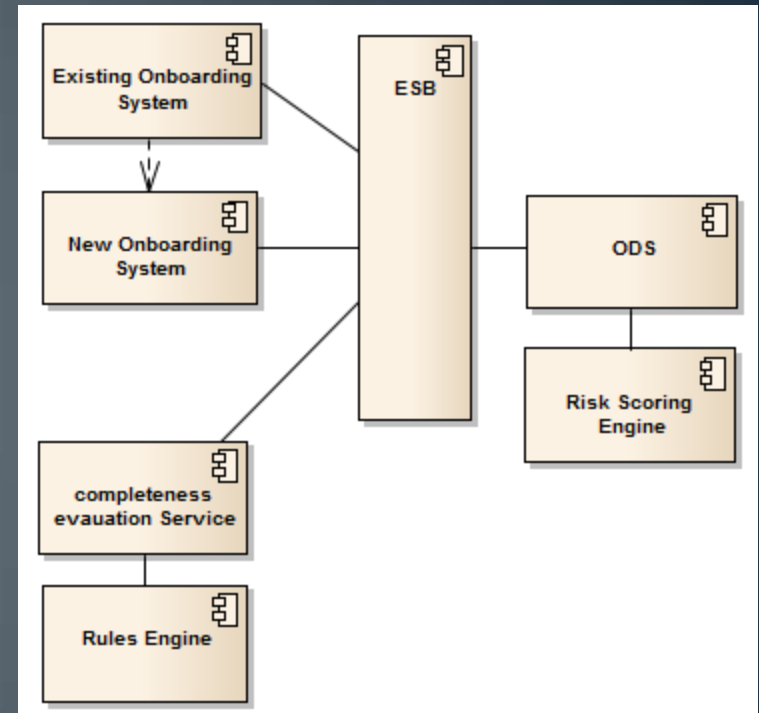
The new onboarding process:

- Has to seamlessly extend current onboarding process (the existing system) including matching UI experience
- Has over 300 new questions to ask depending on customer or account types, planned account activities and previously provided answers.
- Has to implement dynamic flows with overlaying complex UI Interactions:
 - Different UI operating modes
 - Conditional warning messages on many of the user actions.
 - Hard stops
 - Save / cancel behavior
- Not movable Delivery Deadline
 - Facing regulatory sanctions if not delivered.
 - Need to account for lead time needed to develop training materials and provide necessary training in 6000 + branches.

Design & Reference Architecture

The project involves 6 major components and several external vendors and systems:

- Existing onboarding System A (vendor #1)
- New System B used to extent the onboarding System A (vendor #2) that implements vast majority of the new questions / logic / complexity (over 90%).
- ESB / ODS as communication integration hubs / channels for data flow into a Risk Scoring Engine as the final destination (components 2,3 and 4)
- Rules based profile completeness evaluation service
- The new application is a separate application, but has to look and feel exactly like the existing one
- Has to integrate with ESB to receive / pass data.



Original Direction, Setback and looking for a solution

Upon finalizing design, cost and schedule and 3 Months before delivery date - a major set back from vendor #2:

- Some of the “must have” requirements cannot be met.
- Overall cost and schedule is longer than originally estimated

Need to find a solution that will:

- Implement functionality AND Meet all of the must have requirements that the current vendor cannot meet
- Return project back to original cost and schedule

Looking for Options

Based on:

- sheer amount of logic required for the dynamic application to function
- Short project timeline left
- Requirements to use rules engine as one of the components anyways

The call is made to use rules driven UI framework to try to build the new application.

Go/No Go Decision:

- Quick POC to prove that it might be a viable approach:
 - Must have requirements to be implemented as part of the POC
 - Most complex section of the dynamic forms must be implemented as part of the POC

Other Major (must have requirements) for the framework for the Go/No Go decision.

- Robust Rules Management UI – too many to manage otherwise (over a thousand that needs to be built within a few months)
- Cost - there are more than 6000 thousand regular users, so seat licenses or any other complex licensing requirements may impair the project progress
- Dependency on other components, availability of ready to start resources, or inflexible development lifecycle is a MAJOR risk— only 2 months to deliver.

The rules based web frameworks considered:

- Appian
- IBM ILOG
- OpenRules ORD.

Selection of OpenRules ORD

OpenRules Framework was selected based on combination of all factors:

- Cost and Schedule
- A completed POC to prove the ability to meet business the business requirements
- Excel based UI for entering rules
- Simple rules configuration logic
- Simple licensing requirements
- Positive reference checks

Part 2: Design and Development

TOC:

- **Section A:** Running OOTB solution based on templates as a starting point for new application development and structure of typical apps
- **Section B:** Summary of framework and support provided by OpenRules to build the dynamic web applications
 - ORD Templates
 - Data Binding and Special Tags
- **Section C:** Design and Development to specific requirements:
 - Rules based web forms design
 - UI Look and Feel
 - Back End Integration
 - Any Other Customizations

Section A: Running OOTB solution based on templates as a starting point for new application development

Setup of an OOTB Solution based on a Dialog Credit Card App and structure of typical app

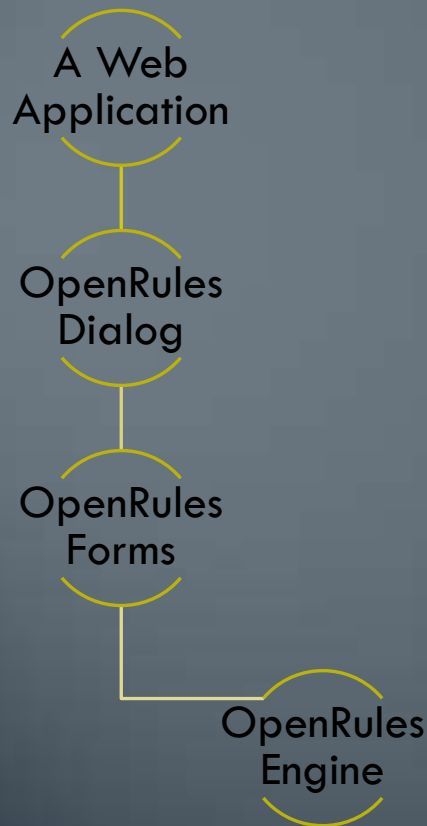
- Required Software:
 - Java
 - Tomcat
 - Ant
 - OpenRules libraries (openRules.config)
 - Sample Template: Dialog Credit Card
- **Demo:** Installation and Deployment of a complete OOTB solution
 - Configure deploy settings
 - Start tomcat
 - Run deploy.bat
- **Demo:** A working dynamic web Application
 - Navigation
 - Dynamic Question / Answers
 - Automated pre-fills based on answers

The screenshot displays a web browser window with the address bar showing 'localhost:8080/dc/'. The browser title is 'DialogCreditCard'. The page contains a form with two main sections: 'Applicant Name and Address' and 'Applicant Data'. The 'Applicant Name and Address' section includes input fields for First Name, Middle Initial, Last Name, Address, City, State (a dropdown menu currently showing 'MA'), and Zip Code (pre-filled with '11371'). The 'Applicant Data' section includes input fields for Home Phone, Home Email, Date of Birth (mm/dd/yy), Social Security Number, Annual Household Income (pre-filled with '100000'), and Employment Type (a dropdown menu currently showing 'Employed'). There are 'Prev' and 'Next' buttons at the top of the form. A small 'OpenRules' logo is visible in the bottom right corner of the form area.

Section B: Summary of framework and support provided by OpenRules to build the dynamic web applications

Summary of the framework and features provided by OpenRules for building apps

- Summary of the OpenRules based Web Application architecture.



- **Demo:** Rules for defining structure and dynamic aspects of the web forms (ORD based):
 - Static definition of Pages, Sections, Questions, Answers, Auto-Responses, Custom Controls
 - Dynamic aspects: defining navigation (pages or tabs) templates, hiding/showing sections, questions children of questions, resetting of sections, answers, defining and processing events.
- Underlying Forms Support (**Example – Next Page**):
 - `<F>` tag for data binding and actions
 - `<C>` tag for including any code
 - Layout marker to create any HTML content
 - Method marker to write any java based code right within the excel

Section B: Example of built-in Web Forms Support Features

- Layout marker
 - To create any HTML content
- Method marker
 - To write any java based code right within the excel
- `<F>` tag
 - Data binding controls
- `<C>` tag
 - including any code

What is the source of incoming domestic electronic transactions? (Check all that apply)

Select

Accounts Receivable (i.e., money due to the business from its clients)

Transfer from other accounts

Other

Method `TableLayout multiSelect(String id)`

```
Question q = dialog().getQuestion(id);
return multiSelectContainerLayout(q);
```

Layout `TableLayout multiSelectContainerLayout(Question q)`

q.name	<pre><div> <C>required(q);</C> <C> TableLayout ms = multiSelectLayout(q); ms; </C> <C> TableLayout hidden=hiddenFieldLayout(q);hidden; </C> </div></pre>
--------	--

Layout `TableLayout hiddenFieldLayout(Question q)`

```
<F style="display:none">[q.getAnswer()]][updateWithEvent(q,p0)]</F>
```

Section C: Design and Development to specific requirements

Building Your own Dynamic Web Application

TOC:

- **Extending User Interface: Using HTML / JavaScript / CSS, and OpenRules templates to create reach user interface**
 - Using / Modifying default look and feel using css and page, section, question templates
 - Extending existing or building new Question/Answer Templates
 - Adding reach GUI elements and interactions
- **Back end integration activities and customizations: building connectors into external systems.**
 - Integration with Vendor A
 - Integration with Enterprise Service Bus (ESB)
- **Extending default capabilities of ORD.**
 - Support for multiple questionnaires in a session
 - Support for ability to copy a portion of answers from another questionnaire in the session
 - Support for tabs rather than pages
 - Adding client / server side logic as per requirements to control conditional actions, modes, hard stops

Extending User Interface: Look and Feel

- **Summary:**
 - html / css touch up to existing default templates to have a required look and feel
- **Example:**
 - Appearance made to match the existing requirements
 - Removed regular header and replaced it with tabs
 - Added “Ok / Cancel” Footer
 - Indented parent / child questions
 - Different operating modes (required more work):
 - Prospecting (questions are not required)
 - Required (the same questions become required)

The screenshot displays a web browser window with the address bar showing 'localhost:8080/dcg/'. The browser's title bar reads 'DialogCreditCard'. The page features a blue header with 'Prev' and 'Next' navigation buttons. The main content area is titled 'Applicant Data' and is organized into two columns: 'Applicant Name and Address' and 'Applicant Other Information'. The 'Applicant Name and Address' column includes input fields for 'First Name', 'Middle Initial', 'Last Name', 'Gender', 'Address', 'City', 'State', and 'Zip Code'. The 'Applicant Other Information' column contains a 'Home Phone' field. Below these columns, there are two sections: 'NON-BANKING FINANCIAL INSTITUTIONS (NBFI)' and 'MONEY SERVICE BUSINESS (MSB)'. Each section contains several questions with radio buttons for 'Yes' and 'No', and dropdown menus for 'Date' and 'Select'. At the bottom right, there are 'OK' and 'CANCEL' buttons.

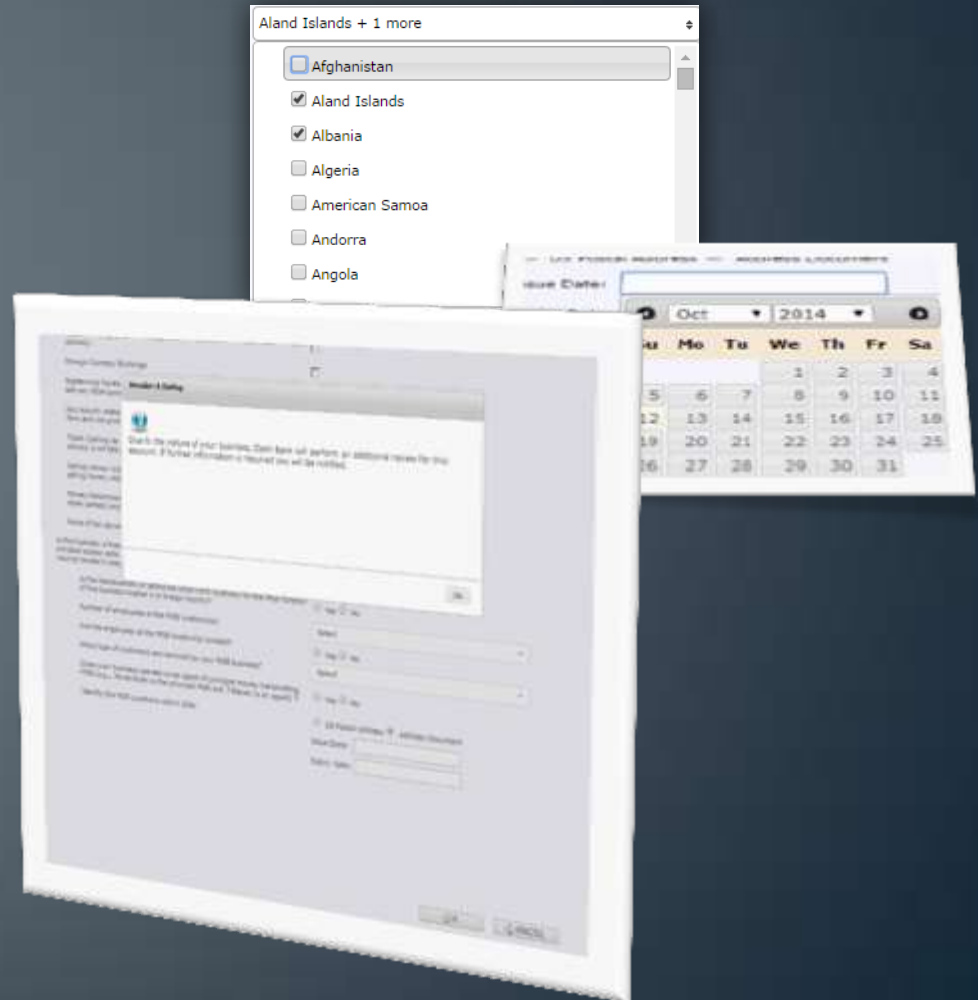
User Interface: Rich GUI Elements

Summary:

- Using more JavaScript, CSS, ORD templates create rich GUI: different type of controls, additional dialog boxes for alerts, confirmations.

Highlights:

- Use ANY js/css frameworks: jquery ui, tw bootstrap, etc.



Example: Extending existing templates

There are dozens of pre-built templates:

- Demo of the question templates
- Demo: extending template as Date Picker:
 - Use existing template (TextBox)
 - Define the hook class in Questions section
 - Configure control behavior in JavaScript



```

setDatePickers : function() {
    $(".question-date").datepicker({
        changeMonth : true,
        changeYear : true,
        yearRange : '1914:2124',
        buttonImage : "calendar.gif",
        onClose: function(dt,obj){
            $(cdd.form).submit();
        }
    }).change(function(){
        cdd.submitForm();
    });
},
    
```

Rules questions extends questionsTemplate										
Question Id	Question Name	Question Type	Size	Hidden	Validation	Required	Unique Tag	UI Class: chose one or several separated by a space :question-date	Hard Stop if answer is	Parent Question
MSBProgram10	<div class="cdd-question-label" style="float:right">Request training materials</div>	CheckBox		Yes	Yes		MSB/ComplianceTrainingMaterialsProvided			MSBProgram7
MSBProgram11	<div class="cdd-question-label" style="float:right">Issue Date </div>	TextBox		Yes	Yes		MSB/ComplianceTrainingMaterials/IssueDt	question-date		MSBProgram10
MSBProgram12	<div class="cdd-question-label" style="float:right">Expiry Date </div>	TextBox		Yes	Yes		MSB/ComplianceTrainingMaterials/ExpDt	question-date		MSBProgram10
MSBProgram13	<div class="cdd-question-label" style="float:right">Is an independent review performed on your business to monitor your BSA/AML program?</div>	RadioButtonSubmit		Yes	Yes		MSB/ComplianceProgramReviewed			
MSBProgram14	<div class="cdd-question-label" style="float:right">Request date of last review</div>	TextBox		Yes	Yes		MSB/LastComplianceReviewDt	question-date		MSBProgram13
MSBProgram15	<div class="cdd-question-label" style="float:right">Request exemption report of available</div>	CheckBox		Yes	Yes		MSB/ExamReportProvided			MSBProgram13

Example: building a completely new Question/Answer Control Template

If not enough, steps to create your own: Multiselect Control example

- Requirements:
 - Ability to select more 1 entry
 - Ability to open / hide sections / questions based on values selected.
- Steps to build
 - Define a new template
 - Call it using configuration
 - Enhance with JS/CSS behavior - just as any other template.

```
Method TableLayout multiSelect(String id)
Question q = dialog().getQuestion(id);
return multiSelectContainerLayout(q);

Layout TableLayout multiSelectContainerLayout(Question q)
<div>
<C>required(q);</C>
<C>
TableLayout ms = multiSelectLayout(q); ms;
</C>
<C>
TableLayout hidden=hiddenFieldLayout(q).hidden;
</C>
</div>

Layout TableLayout hiddenFieldLayout(Question q)
<F style="display:none">[q.getAnswer()]updateWithEvent(q.p0}</F>

Layout TableLayout multiSelectLayout(Question q)
System.out.println("inside multiselect");
String[] answers = q.getPossibleAnswers();
//System.out.println("pos answers: " + answers);
String output="<select class='multiselect'>";
int len= answers.length;
for (int i = 0; i < len; i++)
{
output += "<option ";
if (q.answer != null) {
if (q.answer.toLowerCase().contains(answers[i].toLowerCase())
output += " selected";
}
output += " value=" + answers[i] + ">" + answers[i] + "</option>";
}
output += "</select>";
output;
```

Aland Islands + 1 more

- Afghanistan
- Aland Islands
- Albania
- Algeria
- American Samoa
- Andorra
- Angola

Demo of rules breakdown to support the final structure, flow of the application

- Demo of using rules outside of ORD templates
 - hard Stops, high risk checks, NAICS codes

FYI: Keeping code clean using rules...

Externalize rules out of java code when possible.

Example/Demo: NAICS categories

Rules void defineCategoriesByNAICS(Customer customer)

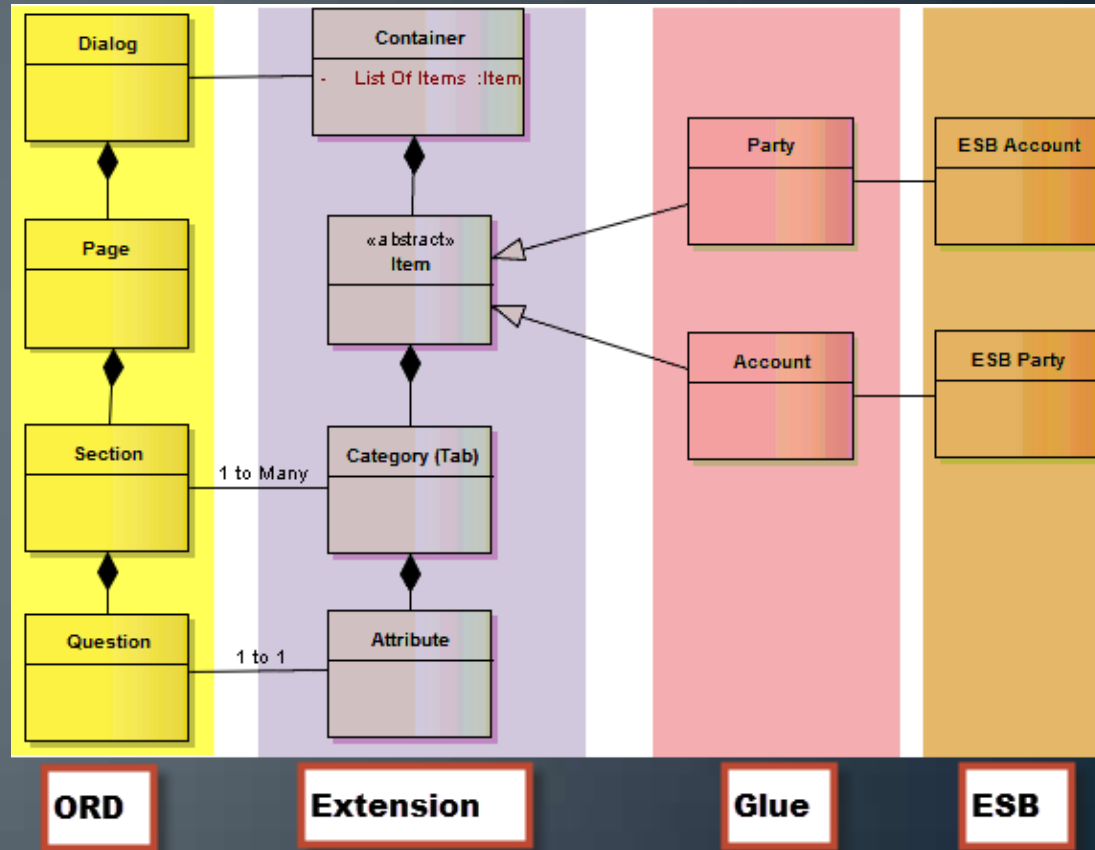
C1	A1	A2	A3	AA	A5
customer.getNaicsCodes().contains(code)		customer.getScore(score)	customer.addCategory(SCOLUMN_TITLE)	customer.addCategory(SCOLUMN_TITLE)	customer.addCategory(SCOLUMN_TITLE)
String code			String a	String a	String a

```
public class Customer extends Item {  
    private List<String> naicsCodes;  
  
    public void loadCategories(){  
        this.container.getDialog().getEngine().  
            .run("defineCategoriesByNAICS", this);  
    }  
}
```

NAICS CODE	NAICS CODE	NAICS CODE	NAICS CODE	NAICS CODE	NAICS CODE
423940	Lev				
441110	Ne				
441120	Us				
441210	Re				
441222	Bo				
441228	Mt				
441310	Auto/Motive Parts and accessories Stores				
445110	Supermarkets and Other Grocery (except Convenience)				
445120	Convenience Stores				
445310	Beer, Wine and Liquor Stores				
446110	Pharmacies and Drug Stores				
446120	Cosmetics, Beauty Supplies and Perfume Stores				
446191	Food (Health) Supplement Stores				
446199	All Other Health and Personal Care Stores				
447110	Gasoline Stations with Convenience Stores				
447180	Other Gasoline Stations				

Extending default behavior of ORD

- By default ORD handled
 - One object in session at a time
 - Multiple pages but not multiple sections
- Our requirements:
 - Use tabs, not pages
 - Define tabs at run time based on objects loaded
 - Handle different types of objects
 - Handle multiple objects and switch between them on a fly
 - In case of multiple accounts, we should be able to copy information category by category



Conclusions and lessons learned

Conclusion:

- Very powerful yet intuitive rules and template architecture
- Short run / test cycles of building web forms using rules dramatically reduce SDLC
- All rules defined declaratively, externalized out of the application code

Suggestions:

- Consider splitting work into separate but parallel tracks using the OOTB template and independently working on UI, back end integration, structure of the web forms
- Building your rules:
 - Rules become as simple as they look ONLY for minds that are analytical in nature.
 - Have people with analytical mind to understand business requirements and translate them into rules.

Appendix (if time permits)

TOC:

- Demo: building forms for entering more than one row
- Demo: dealing with auto-responses.