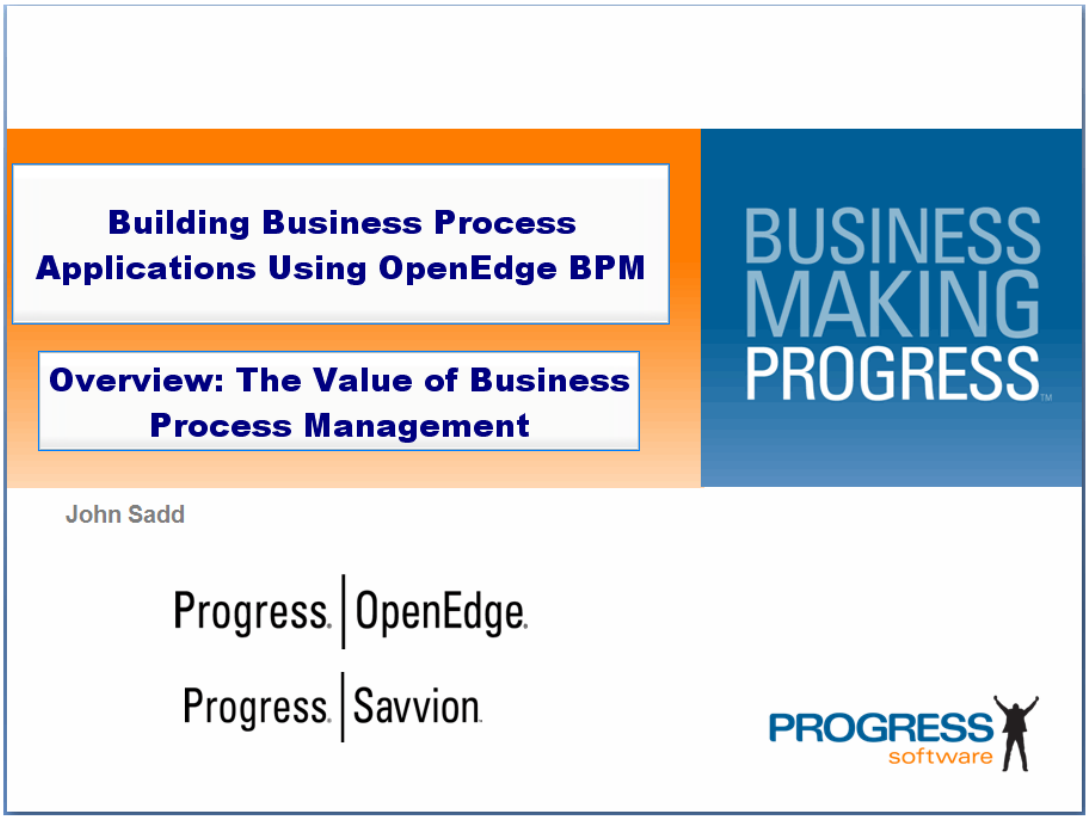
John Sadd
Fellow and OpenEdge Evangelist
Document Version 1.0
July 2011

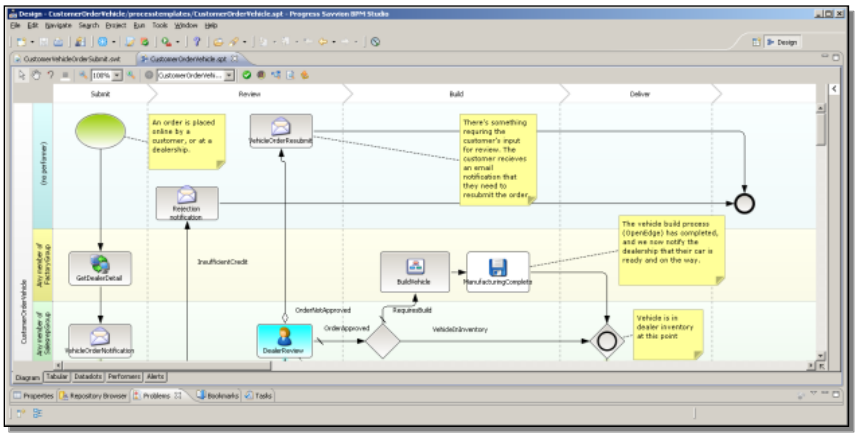


## DISCLAIMER

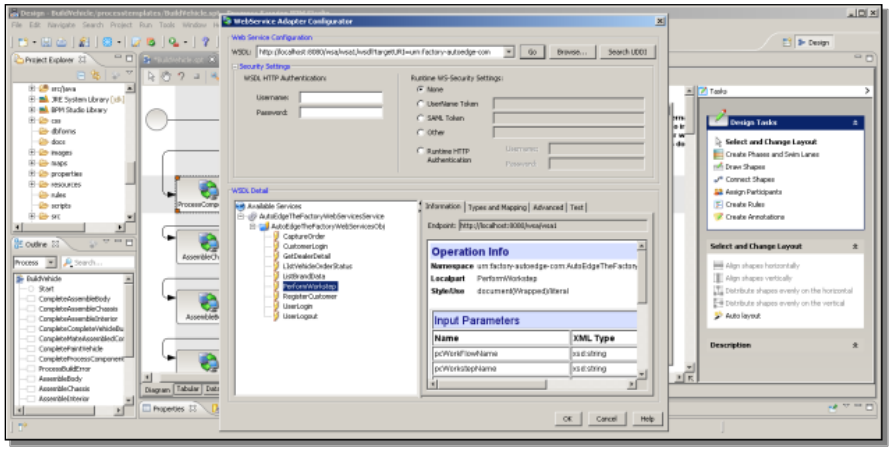


This paper accompanies the first in a series of presentations to introduce you to what we call **OpenEdge BPM**. BPM refers to **Business Process Management**, and these presentations will show you how to take advantage of the many capabilities of **Progress Savvion**, our Business Process Management software product, and how you can use it to extend and to provide process management, or what we have also called workflow support, for your OpenEdge application.

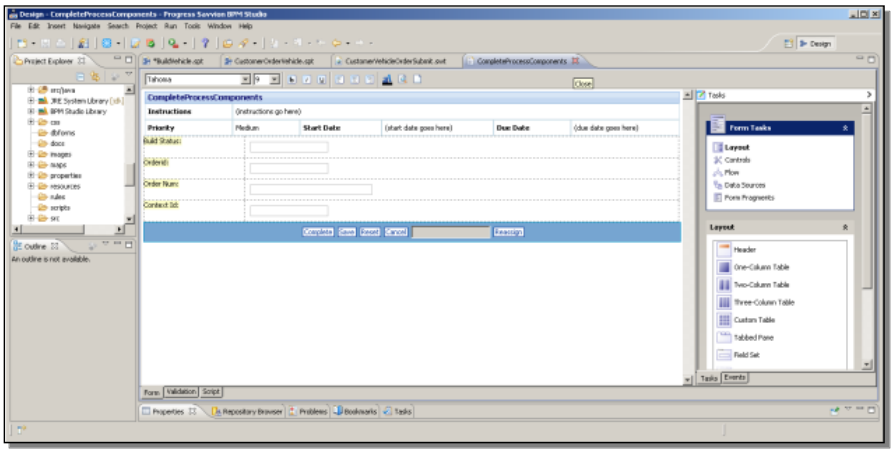
The Savvion toolset lets you define and communicate how the things that your business does operate, how they flow from one step to another. What happens when you process an order, from receiving it to shipping it? What happens when you set up a new customer contract? Having clear answers to these kinds of questions is central to running your business successfully, and those answers need to be encoded in your applications and the way people use them. Processes connect the people who do these jobs with the systems they use, using models like the one shown here:



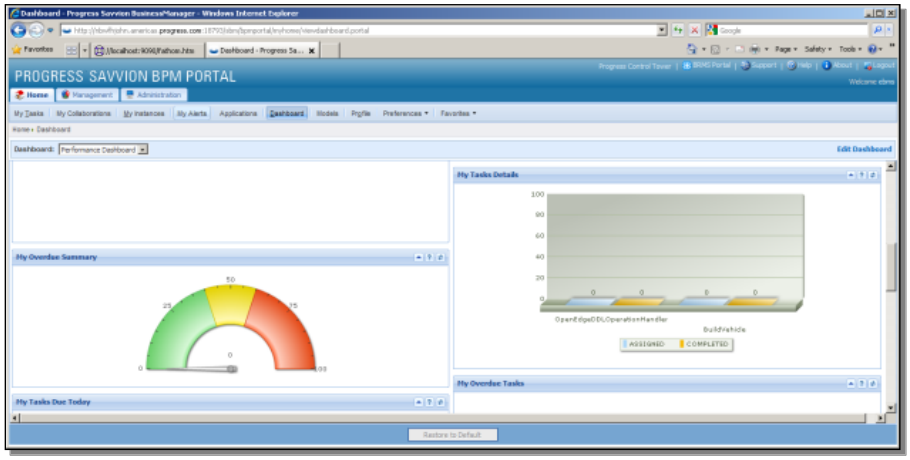

These models connect people and the systems they use together. This diagram shows a model of a process for ordering and building a car or truck. Some steps in the process access systems the company uses. Looking at the step named **Process Components**, for instance, you can see that it accesses a software application entry point as a Web service, passing various parameters to the service:




Here, on the other hand, is a step that represents something a person does, in this case, entering information in a form defined as part of the user interface of the model:

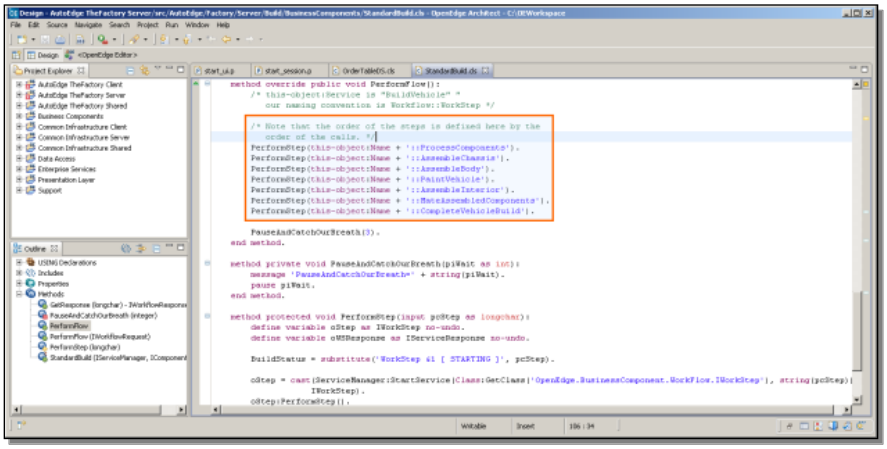


People and systems work together to accomplish the tasks that make up the company's business processes. Rules are another essential element of process models, such as rules that determine whether an order is approved or not, and whether a vehicle is in dealer stock or needs to be built to spec. And modeling benefits from the ability to use measurements and process metrics to optimize a company's business processes:




So why does process management warrant a purposed new software technology to make it happen? Well, probably you would all agree that understanding, and properly designing and maintaining your business processes, is one of the most important aspects of making your business successful. But how well designed and understood are most business processes, and how easy is it for you to understand how to improve and manage them?

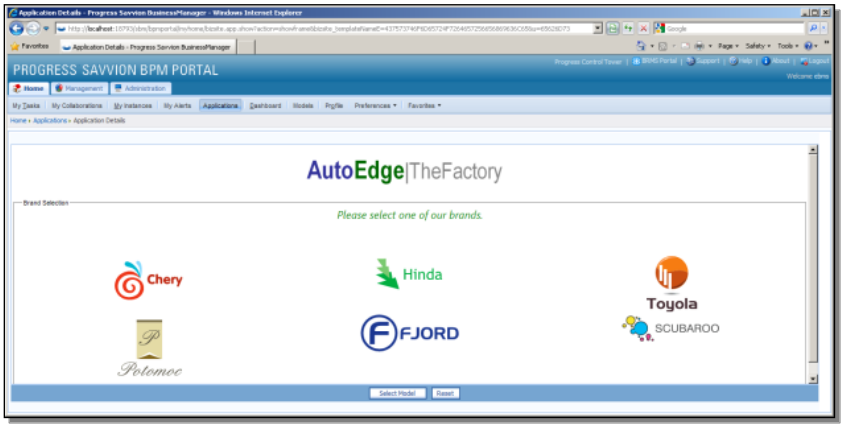
Process management can be hard-coded into the business application itself, as this sequence of steps in an ABL class illustrates:



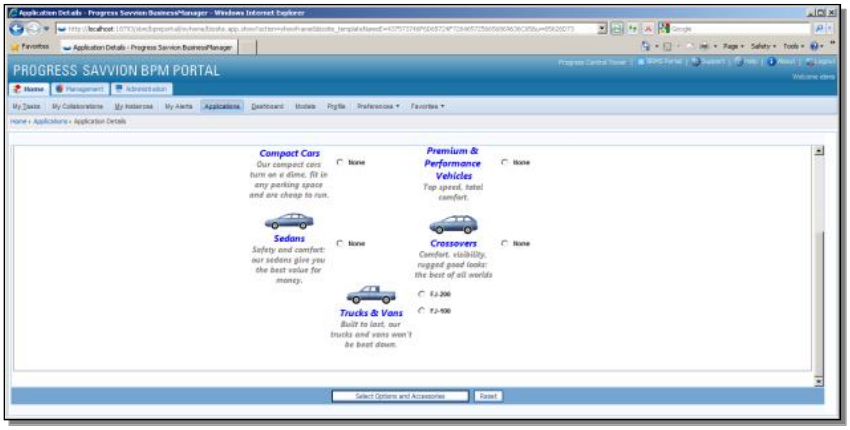
But this can lead to a lot of tangled if-then-else statements that lead a complex data-entry process from one screen to another, or that have various events defined to make tasks appear on someone's screen when something happens in some other part of the application. Most people would admit that this is difficult stuff to express clearly when it's embedded in the business logic of the application, and difficult to maintain. That's why Progress is offering the Savvion product, and why we're talking about how to leverage all the things you can do in Savvion to extend the range of an OpenEdge application, or a larger application with OpenEdge components. Consider a couple of simple examples of the value we're talking about here.

However you've defined (or not defined) process management and workflow within your application, it's probably OK until something goes wrong: management decides that some part of it has to change, or a key person who understands it leaves, or

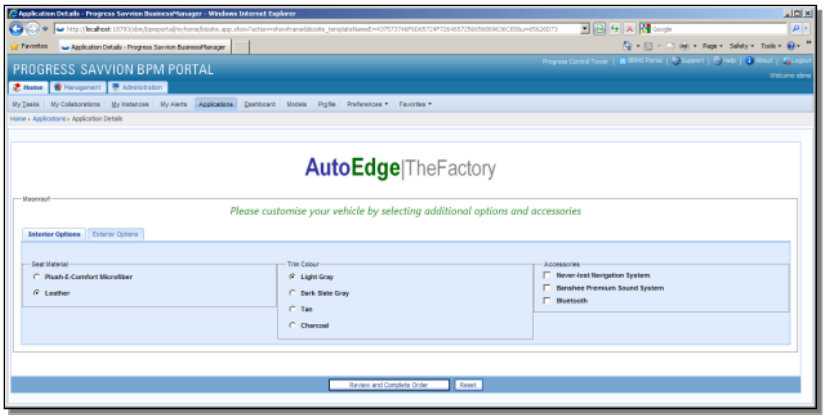
there's an error in the flow at runtime that's hard to track down. So look at just a bit of a sample application to illustrate what I'm getting at here. I can start up an instance of a Savvion business process called **CustomerOrderVehicle,** which leads a customer through the steps to order a new car. First the customer can select from a number of different makes supplied through different dealerships:



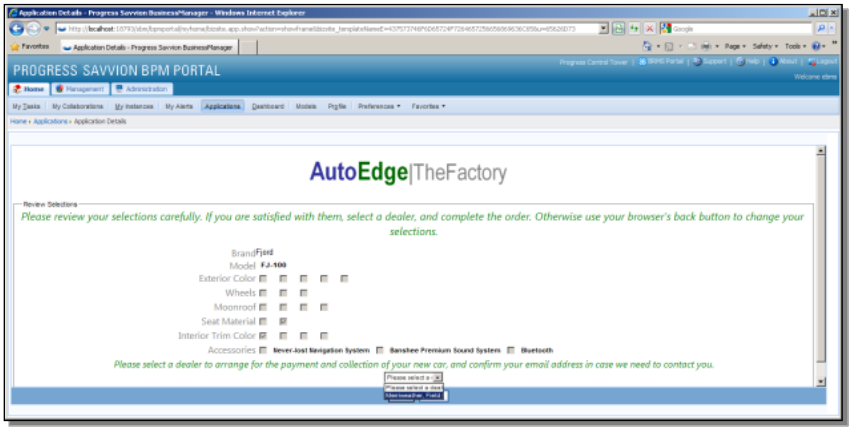

I'll select **Fjord** from among these made-up car brands. And I'll select a Model from among those offered. In this case a pickup truck called an **FJ-100**:



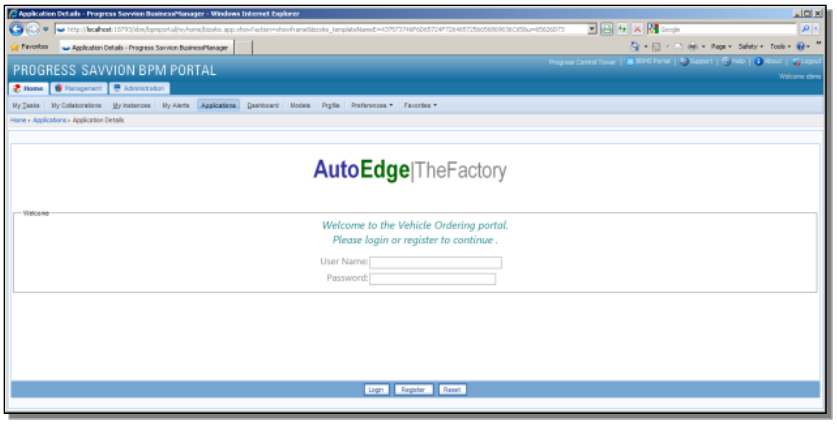

The next screen in the sequence lets me pick some options for the truck, including leather seating, and Light Gray interior trim:



The next step is to review my order, and select a dealer that can sell me the vehicle. So I select the one dealer available in this case:



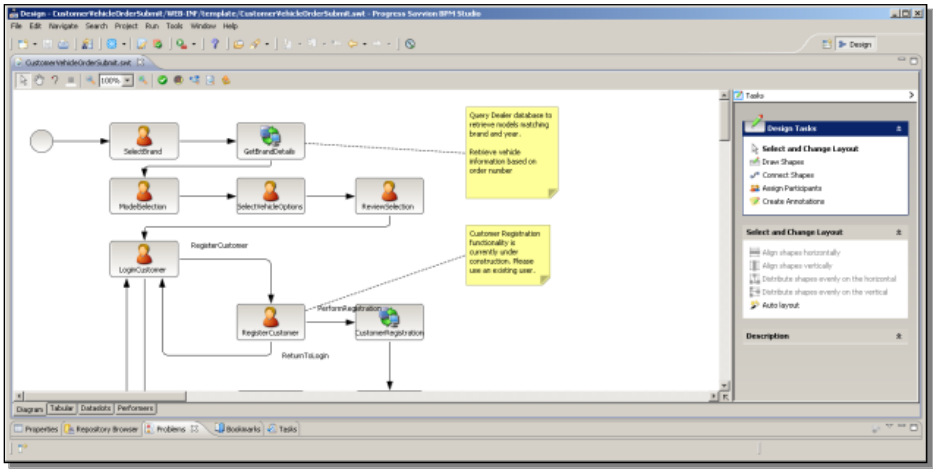

And finally, the sequence that's been defined wants me to log in to the customer order system:



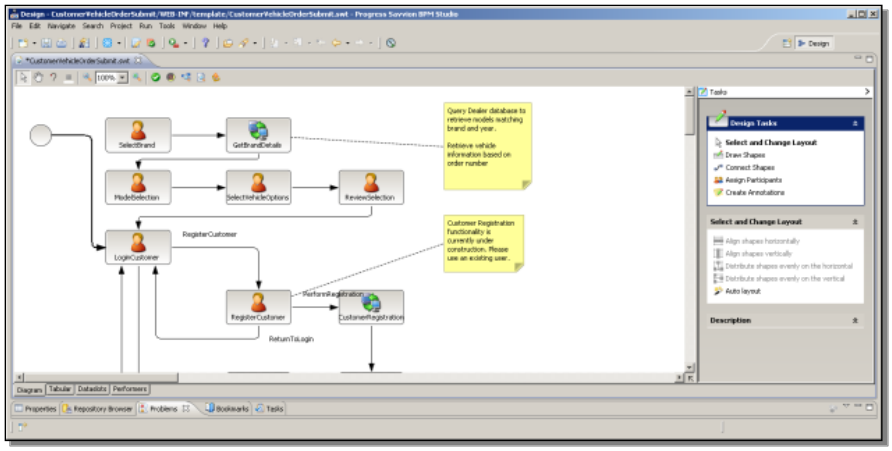

This simulates a lot of online stores where you can browse around, and you don't have to identify yourself until you're ready to buy something. I'll presume I've already registered with a username, and in this sample all I have to do is log in as a known user, and my car will be on its way.

This sequence of screens is defined somehow in the application this customer is using. But let's say that someone decides that it's really better to have the customer log in before doing anything else. Now as the application developer I'm faced with finding that place in all the application code where the logic directs me from screen to screen, and figure out how to change the code without breaking anything else.

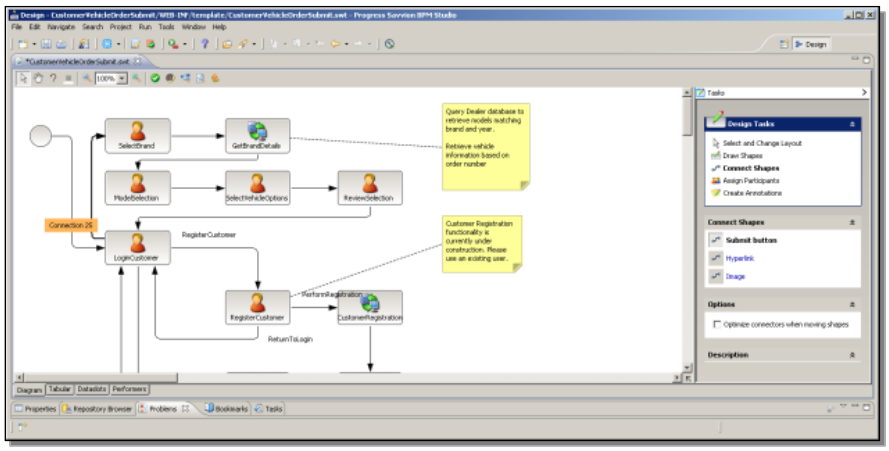
But wait! In my case, I don't have to search through all my code to find the spot to change. My process flow, in this case a screen flow through the customer's user interface experience, is defined as a diagram created using an industry standard process modeling language:



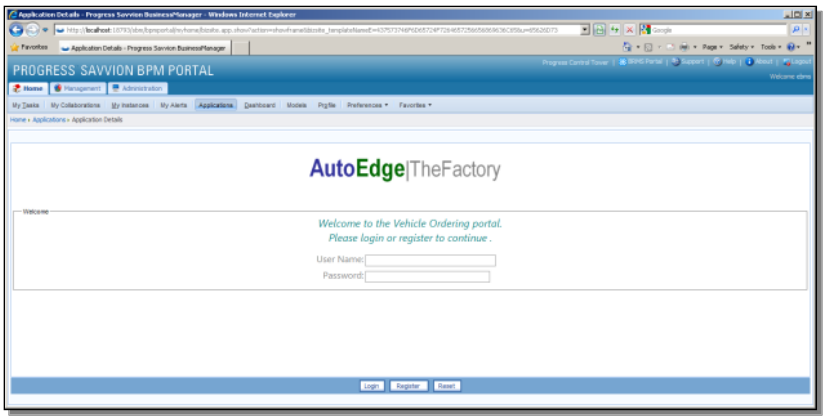
This isn't just a model of a process, it's actually an executable representation of that model. If I want the customer login to be the first thing that happens after the process starts up, I can just move the arrow that represents the link between steps in the process:



Then I select the **Connect Shapes** option in the modeling tool, and make a new connection from the login step back to what was the first thing the customer saw, the **SelectBrand** step in the process:

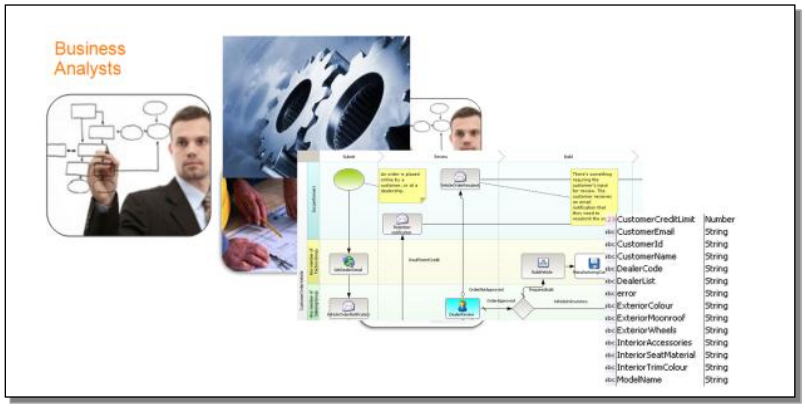


I can now save the model as an executable application. I click to redeploy the application to the server where it executes, and if I return to the tool that lets me start up the CustomerOrderVehicle application again to test it out, I can confirm that the first thing the customer will now see is the login screen:




So a change to a process model diagram that business analysts, developers, and managers can all understand and work from has been translated automatically into an executable process, without any coding.

In these videos I'll talk about some of the different roles that all have a part to play in designing, implementing, managing, and using the workflows that represent how your business functions. And different videos in this series will be targeted most specifically at one or another of these roles, although I expect most if not all of them should be interesting and useful to almost anyone who's affected by your process flows. One type of person we can think of as a **Business Analyst**.




This person  has to understand at a high level what your business does, what the workflows are, what kinds of data are entered or used at different steps in the process and so forth. We're not expecting that this person necessarily has a detailed technical background in software development, so it's important for them to be able to create process diagrams that describe workflows in a non-technical way that they can understand, and at the same time convey information clearly to others who need to implement and manage them. You'll see a little later how Savvion supports this kind of communication.

The second role type is the **Application Developer**.


This person understands how to design and use databases, user interfaces, and the logic that connects them. But they may not understand how the business operates, and they may have a hard time communicating with the analyst in a way that doesn't risk misinterpretation and poor implementation of the analyst's ideas. The developer needs to be able to use that same high-level design diagram that the analyst understands, but then to be able to translate that diagram into part of a working application without anything getting lost in translation.

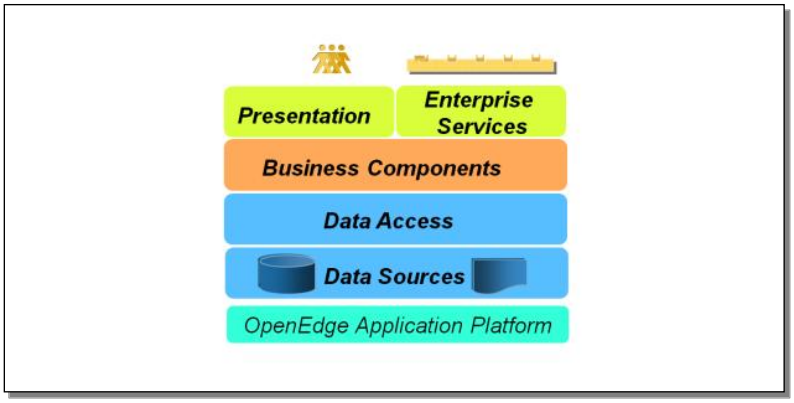
Then there's someone we can call the line of **Business Manager**.


This person supervises the people who do the everyday work of taking and processing orders from customers, or whatever other jobs your business engages in. They don't understand as well how the application software really works, but they need to be able to manage the people who use it, and they need to be able to play an important role in optimizing that software, dealing with problems in day-to-day operations, correcting errors, and generally improving how the business runs. They need their own tools to be able to do these jobs, and later videos in this series will show you how some of those tools work.

Finally, there are the people who do the visible work, the **Application End Users**.


They run the business application to be able to do the things the business does, taking orders, checking inventory, dealing with customer requests, whatever the nature of the business is. They are the prime users of the application software, though they can't be expected to understand how it's implemented. And they are the ones most affected by the business's process management, whether it's been explicitly defined or has simply evolved over time. Savvion makes it much easier for these end users to keep track of what their tasks are and what they should be doing at any given time. I'll show you a bit of how that works as well.

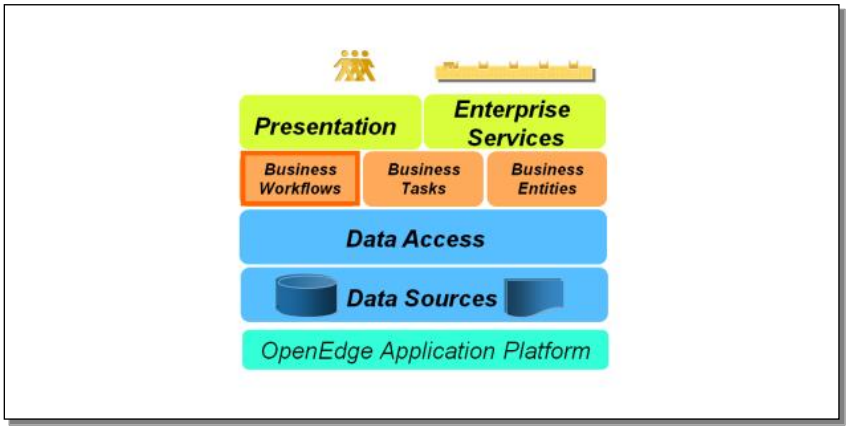
So what have we had to say about this where our own **OpenEdge Reference Architecture** is concerned? Well, the honest answer is, so far not much. Let me review a bit by drilling down into the **Business Components** layer of the Architecture.




The architecture element we call a **Business Entity** represents all the application code needed to manage a single business data component, which can be a complex relational object such as an Order and its Orderlines and so forth, represented typically in a ProDataSet. Database transactions can be defined at this level as updates are received from a client or through a service interface.

The notion of a **Business Task** means an operation that has to access and possibly update multiple Business Entities in a single operation. Perhaps you're updating a Customer's outstanding balance and Order detail information in the same request. This kind of operation can also be managed as a single database transaction, and considered a single request from the perspective of a client or other interface.

**Business Workflows** are more complicated.




Here the expectation is that you're defining an operation that has a number of discrete steps executed over perhaps some considerable period of time. There's no possibility of a single database transaction being kept open over that span, and the flow will require multiple calls into the workflow from different places at different times, with some notion of maintaining process state information over that span. We've never had a very specific example solution to defining a business operation of this kind because there's no straightforward and generally usable solution. This has pretty much been left at the level of, Insert complex code here.

This illustrates the value of Business Process Management with Progress Savvion. In the videos in this series, I'll show you how you can use process models defined in Savvion to tie together the pieces of an OpenEdge application to separate out your workflow from the business logic that is the heart of your ABL application. I'll show you how to execute ABL procedures and classes as steps in a process model, how to call back into a Savvion process from OpenEdge to trigger process events, and how you can use these two powerful products together to create what we call **OEBPM**, Business Process Management in an OpenEdge application. So stay tuned.