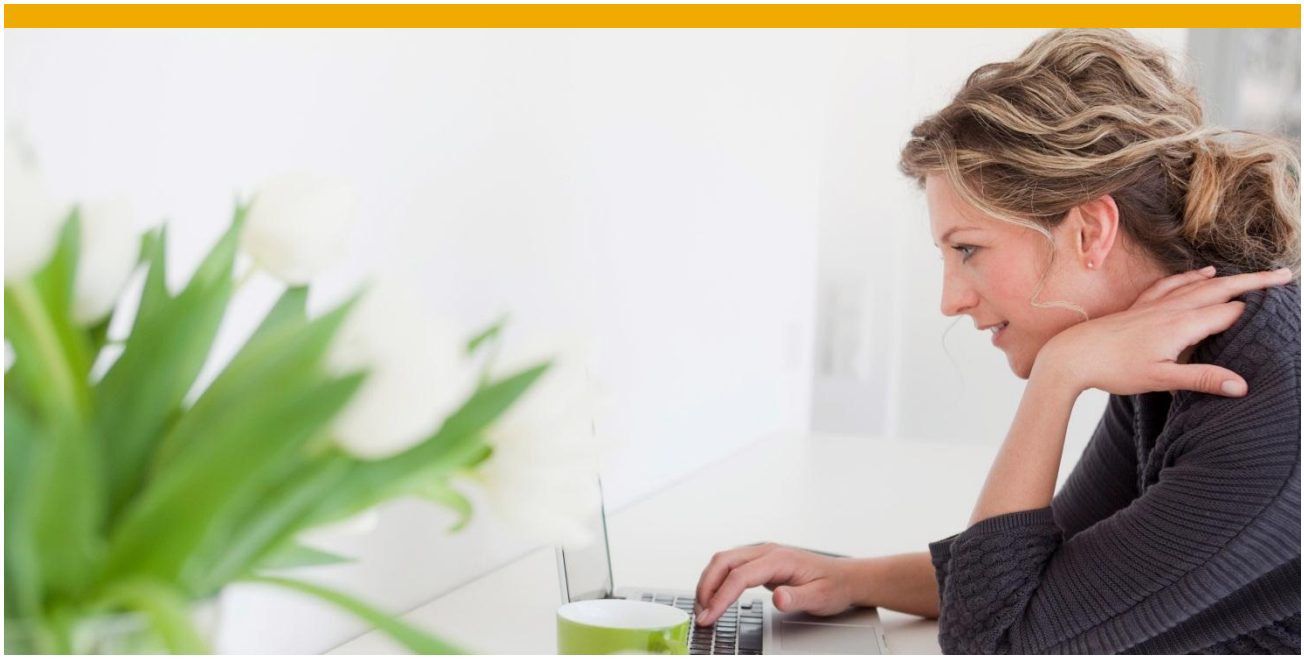

DEVELOPING SOA BASED APPLICATION IN BANKING USING NW VISUAL COMPOSER

GLOBAL BANKING ARCHITECTURE GROUP

TABLE OF CONTENTS

TABLE OF FIGURES

ABOUT THE GLOBAL BANKING ARCHITECTURE GROUP

The Global Banking Architecture Group is responsible for the banking solution architecture domain within the SAP Field Services Global Layer. The group mission is to streamline banking solution architecture practice by providing knowledge, tools and methods developed by us from direct involvement in banking projects around the world. For more info about the group contact Nitzan Levi, head of the group.

ABOUT THIS DOCUMENT

Authors

- Nitzan Levi, Head of Banking Solution Architecture Group (GBAG)
- Avi Malamud – Global Banking Solution Architect (GBAG)
- Efrat Ben David - Global Banking Solution Architect (GBAG)

Reviewers & Contributors

- Monika Eggers, Banking Solution Architect in Banking Solution Architecture Group
- Kobi Sasson, NetWeaver Visual Composer Development, SAP Labs Israel

Target Audience

- Banking Solution Architects
- Customer Solution Architects

Confidentiality
Customer
Creation Date
3-July-2013

OVERVIEW OF THE BANKING APPLICATION DEVELOPED BY GBAG

Composite applications is a concept introduced by SAP a few years back as part of the overall SOA strategy the company took , and it talked about the ability to develop focused/task oriented applications on top of a well-structured SOA foundation exposing the enterprise business software capabilities as web services. For that paradigm SAP has developed & refined both its business software (business suite, CRM, Banking , etc) to be true SOA enabled and Netweaver platform enables all sorts of technology capabilities to build SOA based solutions including the UI modeling tool called NetWeaver Visual Composer.
Global Banking Architecture Group has developed a real banking composite application that implements a Current Account creation task using the verity of capabilities SAP provides from both technology and banking solutions in order to validate the SOA architecture paradigm and the different components SAP has to implement it. The composite application was built on a real SAP Banking landscape includes SAP NetWeaver:

- Visual Composer for UI,
- PI for ESB,
- BPM for Service Orchestration,
- Portal for Runtime

And SAP Banking Services and CRM AO as the underlying core banking solutions.

The application enables the user to select a Business Partner, verify the eligibility of the Business Partner as well as his /her current total commitment and to create a current account by selecting different banking products. These products reflect different conditions for each customer type (Students, VIP's, etc.)
A second variant of this scenario also exists and it used the latest SAP Component for business process orchestration: SAP Process Objects Layer (POL)

DOCUMENT GOALS

To share the team's experience with the topic of developing banking applications using NetWeaver Visual Composer and following the SOA principles and to provide guidelines to develop similar solutions.
In addition, this document should be used as a basis for discussion with the development teams to provide inputs for further enhancements of the various tools used in this process.

CREATE ACCOUNT WEB APPLICATION – DEVELOPMENT APPROACH

As previously mentioned, the web application was developed following SOA principles as defined in our SOA for Banking Architecture Topic.
Based on this, the design of the application took into consideration all 5 architecture layers:

- The End-to-End Business Process
- The Organization Structure
- The Application Flow
- The Process Objects
- The Business Services

Bellow, we describe each of these layers in more details.

The Business Process

The business process is usually representing the way the bank process a specific business activity across the organization and underlying systems following the relevant business rules. A business process could be very simple or complicated and could be built from different steps each could be a process on its own. Moreover depends on the bank the business process could be done by 1 function in the bank or split across different functions using workflow. In our case we took the end to end current account origination process which could start from marketing activities up to creating and servicing the account, and our application fulfills the actual creation of the account which is one step in the overall process. As you can see in the following figure seeing the overall process and the specific step our application implemented which is also built as a process which we call the application flow:

Figure 1: The Business Process

The Organization Structure

The organizational structure represents the way the specific bank is organized in order to fulfill the business process. In our example the overall process could span across different departments from the marketing that responsible to generate the campaign and the field sales that responsible to bring leads and convince potential customers to open an account to the back office team that needs to enter all the info and create the account in the systems.
That said, in our process, we cover the step of creating the current account by a back office clerk based on the assumption that this back office clerk receives a request to create the account for a customer and once the account is created, the customer will be eventually notified and will be able to use the account.
This assumption is not groundless, as in various customer engagements we realized that this is the reality: employees receive a request to open an account (via fax, email, written form), they create the account and send a message back (fax, email, etc.) to notify the customer.
The following image shows the organization aspects of overall business process:

Figure 2: The Organization Aspect

The Application Flow

The application flow represents the “wizard” style application that guides the end user on the sequence of steps he should follow in order to fulfill the task. In our application the task is to create the current account in the core system however the steps are:

- Search the Business Partner (customer) data info if does not exist create one
- Check his total commitment to get a picture of his overall status
- Check eligibility (black list and credit scoring)
- Enter some data like type of account, and whether it requires a check book or not
- Send request to the core system to actually create the account

The application was developed following the design principles of SOA:

- Synchronous task done by an individual
- Agnostic of the organizational structure
- No business logics in the application
- Uses Enterprise Services for validation rules and data retrieval
- Manages UI and human interaction flows only on the activity level

We used the NetWeaver Visual Composer to implement the application flow and all the required UI components all in a modeling no code manner, consuming the logic and data services from the underlying systems via the ESB.
The following images show the application flow in which in our case we developed using the NetWeaver Visual Composer:

Figure 3 :The Application Flow

The Process Objects

The process objects are actually compound services representing a backend orchestration of single services that in a specific sequence provides a specific task. For example one could think that to create an account the a core system you just call one service that creates an account record however in real life such task requires several steps that have to happen in a specific sequence. For example you need to first create the customer record, only if ok create the account and with the account number create the check book. Also as the UI flow could take time, meaning the banker could do the first step and then wait for a few minutes and then continue things might change in between. So if you check the black list you might want to check it again just before actually creating the account, this could be added to the process object of the create account.
In our application we implemented the Process Objects in two different ways :

1. Using the NetWeaver BPM to orchestrate the sequence of calls
2. Using the SAP Process Object Builder , adding some more features of simulation and long lasting capabilities

Both at the end exposed as a single service call to the application which does not care about the underlying orchestration.

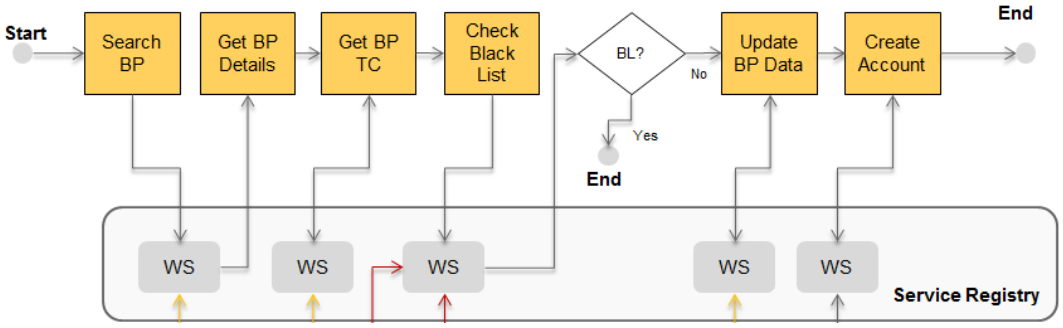

Figure 4 : The Process Objects

The Business Services

To implement such application you need to have the underlying business services that represent the different core systems that either provide the date for the process or fulfill the different transactions like create the BP, account and check book. Therefore the initial challenge was to identify the necessary services and understand their working mechanism. Once identified and according to the SOA design these services should be exposed through a central service bus and this is what was done for this application: all of our services were mediated through SAP NW PI.

In real life scenario like ours the exposure of services is not 1:1 in a trivial manner and sometimes require the ESB to expose different interface to the application and map it to the original underlying service , in our case our initial aim was to use the standard Enterprise Services as provided by SAP. That said, there are some things that need to be taken into account:

1. In some cases, some of the functionality needed is not exposed by Enterprise Services (for example, Total Commitment)
2. Due to their nature and requirements, some Enterprise Services are – by design – relatively complex and not all tools are able to consume them as is.

For functionality we didn't have services for, we actually built a Web Service using the Enterprise Service Builder (part of NW PI) and implemented a proxy that calls the appropriate RFC functions (BAPI's).
For the complex services that NetWeaver Visual Composer could not consume, we created a mediated service that simplified the original one so instead of hundreds of fields, we managed to reduce it to just the 3 – 4 fields needed for that functionality.
In addition to the web services exposed by the back-end systems (SAP BaS and SAP CRM), we also created additional services that expose rules (using SAP BRF+) and processes (using SAP NW BPM or SAP Process Object Builder). Needless to mention that Visual Composer consumed these services regardless of the functions they hold.
In addition to services provided by SAP components like CRM and BaS, the Create Account scenario also consumes non-SAP services such "External Black List Check" and "Credit Score Check". These 2 services are implemented using EJB's running on a NW Java server and they just imitate such functionality that banks usually call from an external provider. Beyond that, the services are also mediated in the ESB so we could reach a coherent service provisioning layer.
After creating all the services, we publish them all in the Service Registry of the Banking landscape. This enabled us to both imitate a real customer situation and also other applications can now consume them, ensuring Service Reusability

The following image shows the system in used and the services we consume for these systems

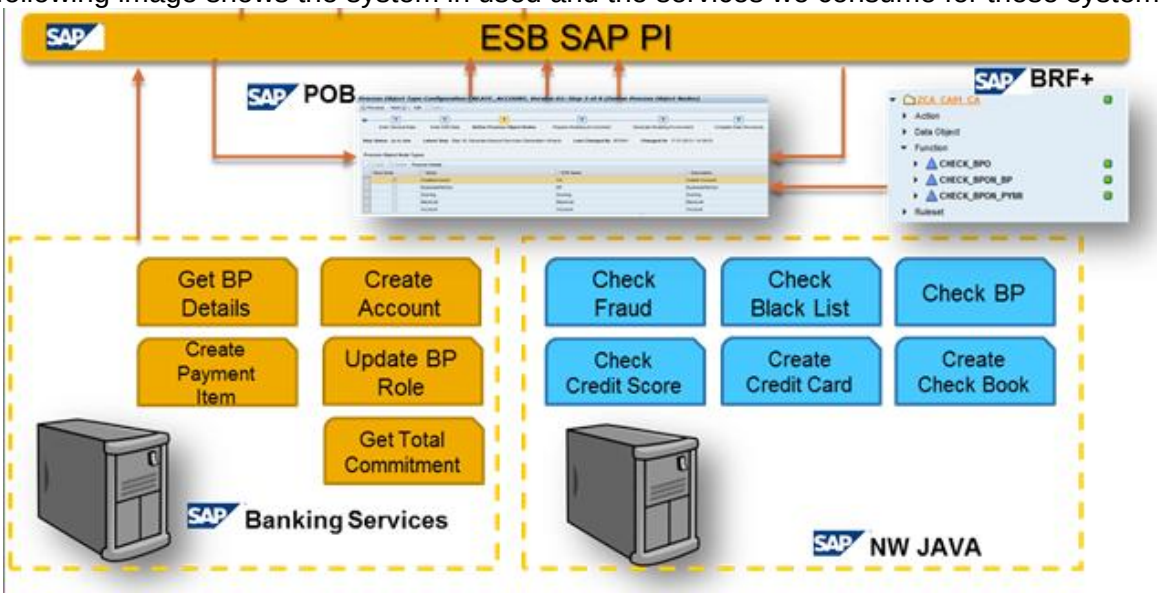

Figure 5 : The Business Services

THE NW VISUAL COMPOSER MODEL

The main requirement for this web application was to create a wizard-like process that will guide a bank clerk through the flow of creating a new account. For this purpose, we decided to use the Wizard component of NetWeaver Visual Composer. The rest of the model was created using the basic NetWeaver Visual Composer functions, as described below.
The Create Account Process is running on one single web application. This application was fully developed with SAP NW Visual Composer. In our case, we used Version 7.31 although the first application version was built on NW 7.01
As this is a web application, it runs from a web browser. This solution fits the typical bank approach: to develop web applications that are then embedded into the bank employee Portal allowing the bank to have one place for all their applications regardless to the back-hand systems they run on.
As we developed this web application using NetWeaver Visual Composer, the run-time of it is a WebDynpro for Java, therefore it must run on a NetWeaver Application Server. In our case, we expose the web application using a NW Enterprise Portal
NetWeaver Visual Composer, enabled us to focus on the process and not on the technical aspects of a web application development as NetWeaver Visual Composer provides all the controls, layout design options, personalization options, etc. so we actually didn't write a single line of code – as it is supposed to be done when NetWeaver Visual Composer is used for application development.
The NetWeaver Visual Composer application uses only Web Services that are published in the Service Registry of the banking landscape. This fact represents a realistic scenario in banks in which the bank provides web services to the various application development teams and these services are consumed by them.
In our scenario, NetWeaver Visual Composer consumes various types of web services:

- Queries
- Execution
- Rules

In addition, the web services used are both SAP Standard (Enterprise Services) and generic, so we are able to show a true realistic scenario.

NetWeaver Visual Composer Model Development Approach

The guidelines for the model development were clear and simple: use only the available functionality of NetWeaver Visual Composer, even if it forces us to limit the application capabilities. With this guideline in mind, we proceeded with the developing the model as usual:

- Identify all services – NetWeaver Visual Composer was pointed to the Service Registry located in the Banking Toolbox landscape (in our case, the Service Registry is run by SAP PI)
- For each service, add the necessary forms, tables, links, mapping, etc.
- Fort each form and table, hide the unnecessary fields, rename field names to be more friendly in the UI
- Arrange the UI layout

At the end, the NetWeaver Visual Composer model includes 9 services, 4 wizard steps and 18 forms and tables, in addition to panels and pop-ups components, to ensure a more suitable layout and application behavior.

We actually created 2 separate applications, each calling a different service that creates an account: one calling a service run by SAP NW BPM and one calling a service run by a fairly new ABAP functionality: SAP Process Object Layer. In both cases, NetWeaver Visual Composer is totally agnostic to the fact that the service is running a process.
The NetWeaver Visual Composer model also followed the guidelines of SOA development, and in our case, no logic was added to the application. All the business login, checks, processes, etc. are provided by the process components used (in our case, web services). This was a crucial fact for us, as we were able to create the same application (consuming the same services) in different technologies, like .Net.

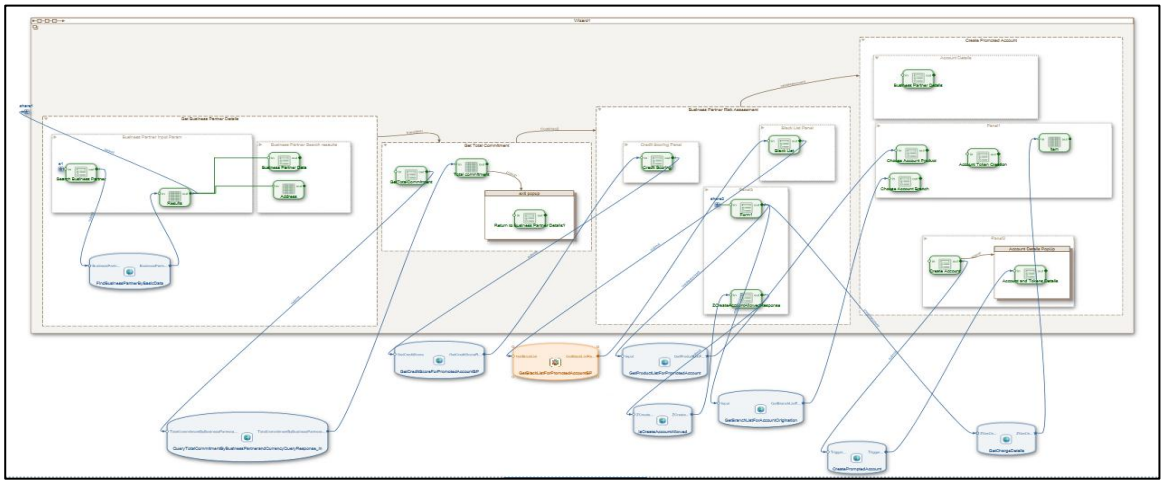

Figure 6 : The NetWeaver Visual Composer Model

The Development Environment

For this model, we decided to use the Web development environment that is available with the installation of SAP NW Java Stack.
The first model was created back in 2007. By then, the SAP NW CE was available and that was the version used.
The development environment provided as an Eclipse Plug-In (as part of SAP NW Development Studio) appeared only in 2011 and we decided not to use it since from our point of view, it's a more complicated environment and doesn't provide any additional benefits to the scenario. One of the biggest issues we found with using Sap NW Development Studio is the fact that there is a need to "find" the right version that matches the server on which the applications should be deployed.
Since the Global Banking Architecture Group deploys all applications on the One-Touch Banking Toolbox, and since the Banking Toolbox may include more than one NW version, we decided that using SAP NW Development Studio as the development environment of NetWeaver Visual Composer is not the right approach for us.

Developing with various developers

The whole development of the model was done by Banking Solution Architects from the Global Banking Architecture Group. As such, none of these architects is a NetWeaver Visual Composer expert, as we wanted to imitate a situation in which there are no developers but rather, business users. To some extent, we can say that this is a plausible situation.
Note: We are aware of the fact that the NW AS on which NetWeaver Visual Composer runs can be connected to a development management tool like perforce but in a real customer situation, this option may not be valid, therefore we think that the NW AS should include this capability and that a model should be shared.

Troubleshooting

As with any development tool, using NetWeaver Visual Composer was not always straight forward and we encountered cases in which the functionality of VC required us to use some workarounds. Some of the issues we encountered were:

- Ability of NetWeaver Visual Composer to deal with standard SAP Enterprise Services (complex structures, long WSDL files)
- Ability to generate a GUID by the application – this is critical for calling some services, although this should be a server side functionality
- Limited layout capabilities, ability to use different fonts, colors, objects positions, etc.
- As with any development tool, some solutions required a combination of features that have to be learned and exploited.

For these cases, we contacted the NetWeaver Visual Composer development team in SAP Labs Israel and we found that the team was extremely helpful and knowledgeable. That said, this fact raises our concerns for a case in which the development is done at a customer site.

SOLUTION ARCHITECTURE

The solution architecture of the banking application is presented in Figure 8.

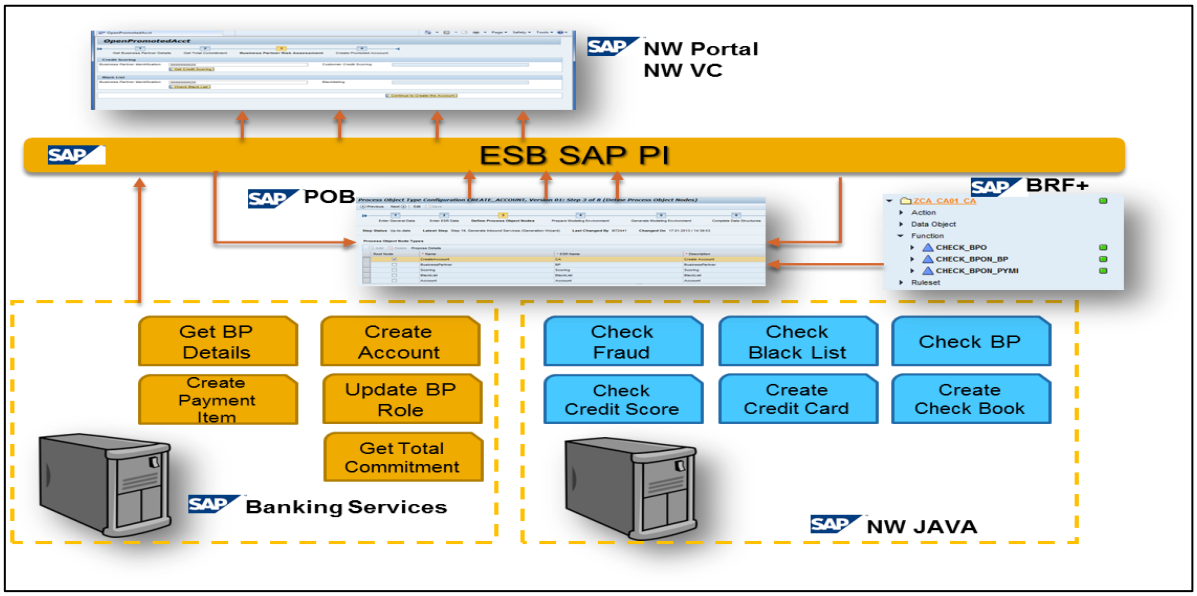

Figure 7 : Overall Solution Architecture

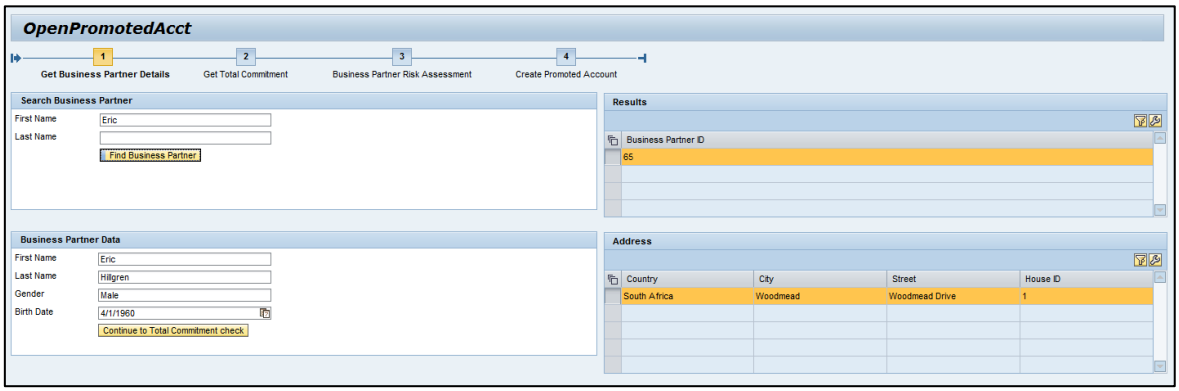

Figure 8 : The Application UI - Runtime

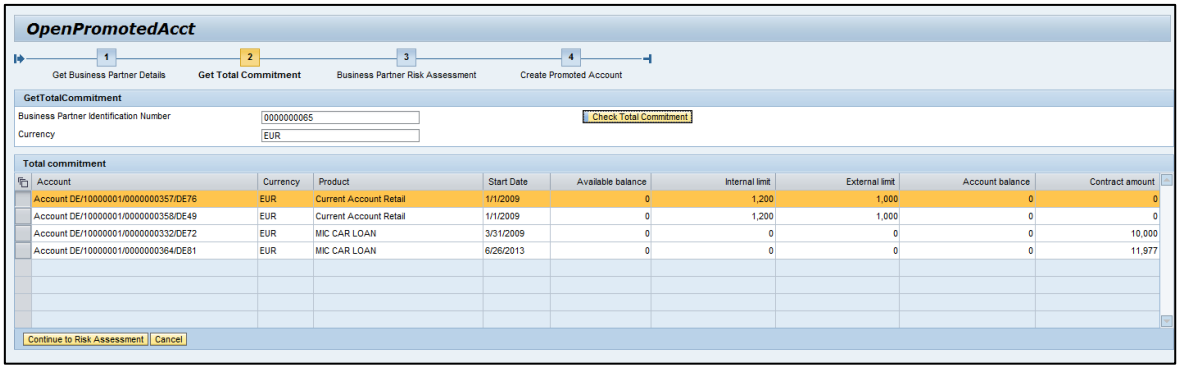

Figure 9 : The Application UI – Runtime 2

SUMMARY & CONCLUSIONS

One of our main objectives for developing such application was to check the capabilities of NetWeaver Visual Composer to be a viable development environment solution in a real customer project.
In general, we found that the answer is a YES, NetWeaver Visual Composer can certainly be a great solution to develop WebDynpro applications in a bank. That said, since NetWeaver Visual Composer is a mode driven development environment, and no code is written, we are aware of the limitations of such tool. We found the “limitations” of a model driven tool as strength point as it direct you to think simple, and stay within the boundaries of a well-defined layered application.
In most of the cases we saw in banking customers around the world the applications required to fulfill the different banking activities within the bank (not talking about apps for the bank customers) are quite simple and straightforward therefore using such development tool and such layered design could accelerate the development of new applications, and simplify their maintenance going forward.

The fact that NW Visual Composer is designed to fit the SOA paradigm makes him a good fit to banks that adopted the SOA and exposed their core banking functionality using web services published on an Enterprise Service Bas (ESB) have it be SAP (NW PI) or any other vendor.
In general, we can characterize the type of applications that can be developed using NetWeaver Visual Composer as follows:

- Single user application to perform specific tasks
- A well scoped application, supporting a pre-define guided procedure such as: customer verification, loan application etc.
- Apps that could be combined with any standard workflow engine to fulfill a bigger process cross users, cross business functions
- Wizard based UI’s with basic UI components for presenting data
- An application composed from a number of services been called from various systems.

For more information on the NetWeaver Platform in general and Visual Composer specifically we recommend going to SAP SCN

www.sap.com