A Native Approach to Service Oriented Modeling and Implementing of Business Processes

Jacek Gruber, Tomasz Brzozowski

Institute of Applied Informatics
Wroclaw University of Technology
Wybrzeze Wyspianskiego 27
50-370 Wroclaw, Poland
Jacek.Gruber@pwr.wroc.pl
Brzozowski.Tomasz@gmail.com

Abstract: For the last few years Sybase Company [SYB] proved its commitment to improving business modeling by creating integrated infrastructure for modeling enterprise businesses. Thanks to the incorporation of Service Oriented Architecture (SOA), Web services and business process execution languages, Sybase has significantly extended business modeling in almost every direction. This article is intended to provide a short survey of Sybase approach, examining its usage and related key aspects - among them: defining static and dynamic view of business organization, implementing business processes by specifying external services and deploying processes to a target execution platform, which allows orchestration and execution simulation.

1 Introduction

Historically, the primary goal in Business Process Modeling, as one of the disciplines in Software Development Lifecycle, was to create common communication platform between system developers and people strictly related to the business. Business modeling languages, graphical notations, business UML profiles – they all were genuinely meant to provide infrastructure for expressing business processes (workflows, business objects, rules), in a way, which could make them understandable for IT architects. Along the evolution of business modeling both sides of the contract formulated a few new expectations, which changed a bit the primary purpose of business modeling. Stakeholders and end-users expect to have a tool for orchestration, management and simulation of different business organizations to be able to quickly choose enterprise configuration, which would support company end-goals in most efficient manner. On the other hand, developers demand business modeling architecture, which could get most out of the latest IT technologies and which could be semi-automatically transformed into executable system architecture by implementing processes using service-oriented approach.

Nowadays there are a lot of different modeling environments, which support enterprise business modeling. One of them was proposed by Sybase and the thing, which seems to make it slightly different from all the others is business driven approach to software development. Implementing this approach forced Sybase to put business modeling in the very foreground and at the time being Sybase is one of the top leaders in business process modeling and implementation. Attempts to fulfill developers' and stakeholders' expectations pushed Sybase to integrate business modeling with SOA. This resulted in creating executable business modeling environment, which remained completely independent from application considerations and which at the same time enables rapid and seamless integration within the enterprise.

This short article is intended to describe key aspects of business modeling approach presented by Sybase, mostly by presenting, how to design analytic and executable business process model using PowerDesigner platform and then how to orchestrate and execute business processes using Unwired Orchestrator. We also decided to present an academic case study to better visualize the whole idea, as well as to highlight couple very specific aspects, especially concerning implementation of business processes using Web services.

2 Business modelling according to Sybase – quick overview

Business Process Modeling on Sybase platform is divided into two methodological aspects:

- Analysis Business Process Modeling – this aspect is responsible for providing 'common language' between people strictly related to the business and people not familiar with it. It provides graphical notation for describing enterprise business organization on a very high level of abstraction.

- Executable and Orchestration Business Process Modeling – this aspect covers implementation of previously defined and structured processes by orchestrating activities and executing services according to Service Oriented Architecture. This aspect is also responsible for supervision and simulation of process execution.

Typical strategy for business modeling assumes starting with analysis model, which is perfect for discovering existing processes and data flows within an enterprise organization. Moreover even a non-technical analyst without any IT knowledge is capable of defining company organization by building Analysis Business Process Model (Analysis BPM). Next we can generate the analysis model into a new BPM for SOA executable language. Choosing SOA for the executable language still gives us very high level of abstraction and still doesn't include any technical considerations. This allows developers and business people to work together on execution business architecture.

Once we have valid Execution BPM (EBPM) we can very easily generate it into another EBPM, but this time choosing less abstract execution language. In practice we can choose between "Sybase Unwired Orchestrator Language" (using Orchestrator as a target execution engine) and "Business Process Execution Language for Web Services" – BPEL4WS (to drive the implementation using application server like BEA@WebLogic). After checking, if newly created model was generated in accordance with the target language, we can move further and start importing WSDL files and implementing processes utilizing imported Web services. But this is not the end, we can step even further by exporting our model to Unwired Orchestrator project, which makes it possible to deploy implemented processes and to refine all the aspects related to processes execution. Figure 1 shows a quick overview of business modeling approach proposed by Sybase.

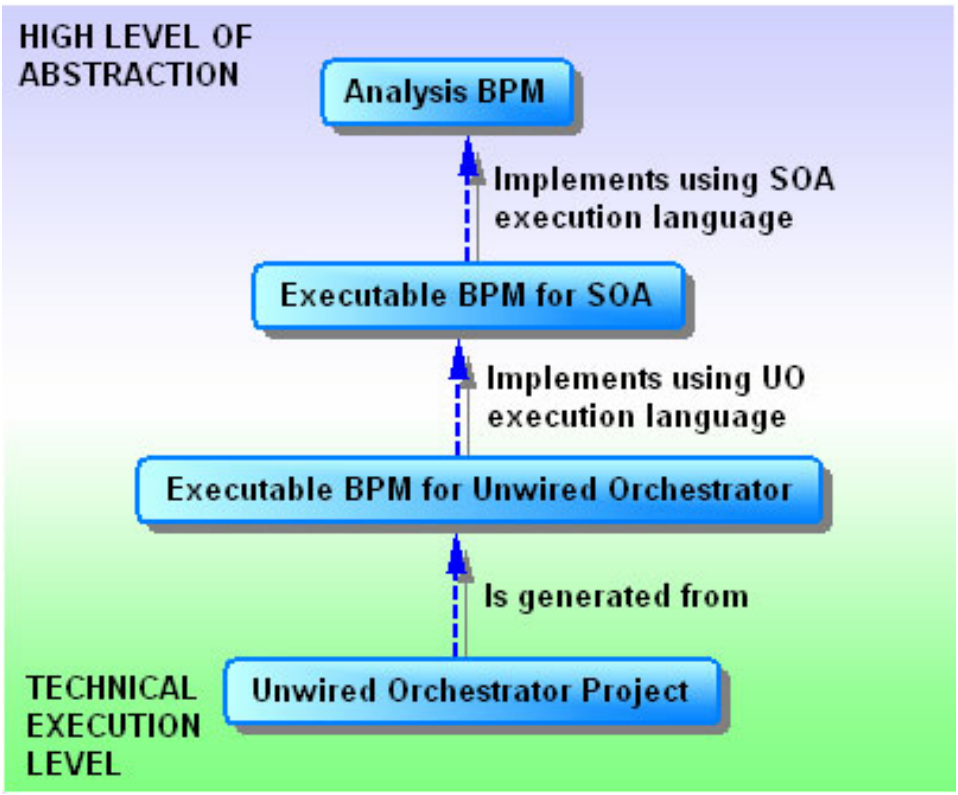

Figure 1: Relations and dependencies between business modeling artifacts

3 Analysis BPM

Since the very beginning of business modeling IT developers, who are used to technical description, had a huge problem with capturing business requirements as well as capturing the current and future business processes using non-technical description. Analysis BPM is one of the business modeling elements, which appeared when IT people realized that business process modeling is something completely different than application modeling and it requires a lot more abstract approach.

At this stage we should answer three questions:

- What are our business requirements?
- How do existing business processes work?
- How should the new system work?

First step in designing analysis BPM is discovering top-level processes representing top-level business functions. Each top-level business process can be defined as set of activities, which realize common business goal or common business function. In the next steps previously identified top-level processes have to be decomposed (into set of sub-processes) down to the elementary level. Elementary level process is defined as 'single activity performed by a single person at single point in time and place, which brings specific business value and leaves company data in a consistent state' [PDUG]. Discovered processes and their decompositions can be defined in a hierarchical business process diagram, which becomes the first artifact of business modeling. This diagram defines static structure of our business model using only two types of elements: processes and decomposition links displayed as branches in the diagram (lines that join the process symbols) – Figure 2 presents hierarchical business process diagram, which defines the scope of our example, which is in this case 'order processing system'.

Figure 2: Hierarchic diagram representing decomposition for a composite process

By looking at the Figure 2 we can easily say, which process is composite, and which sub-processes is it made up from. However we can't say anything about execution order for sub-processes and interactions between them. Hierarchic structure is far from satisfactory when it is about expressing dynamic aspect of business process modeling. That's why after finishing hierarchic diagram we must move to analysis of separate processes. For every composite process we need to design business process flow diagram, which can reflect such dynamic aspects as: control flow, data flow, interactions between processes and message formats (Figure 3 presents business process flow diagram for 'take order' process from previous example).

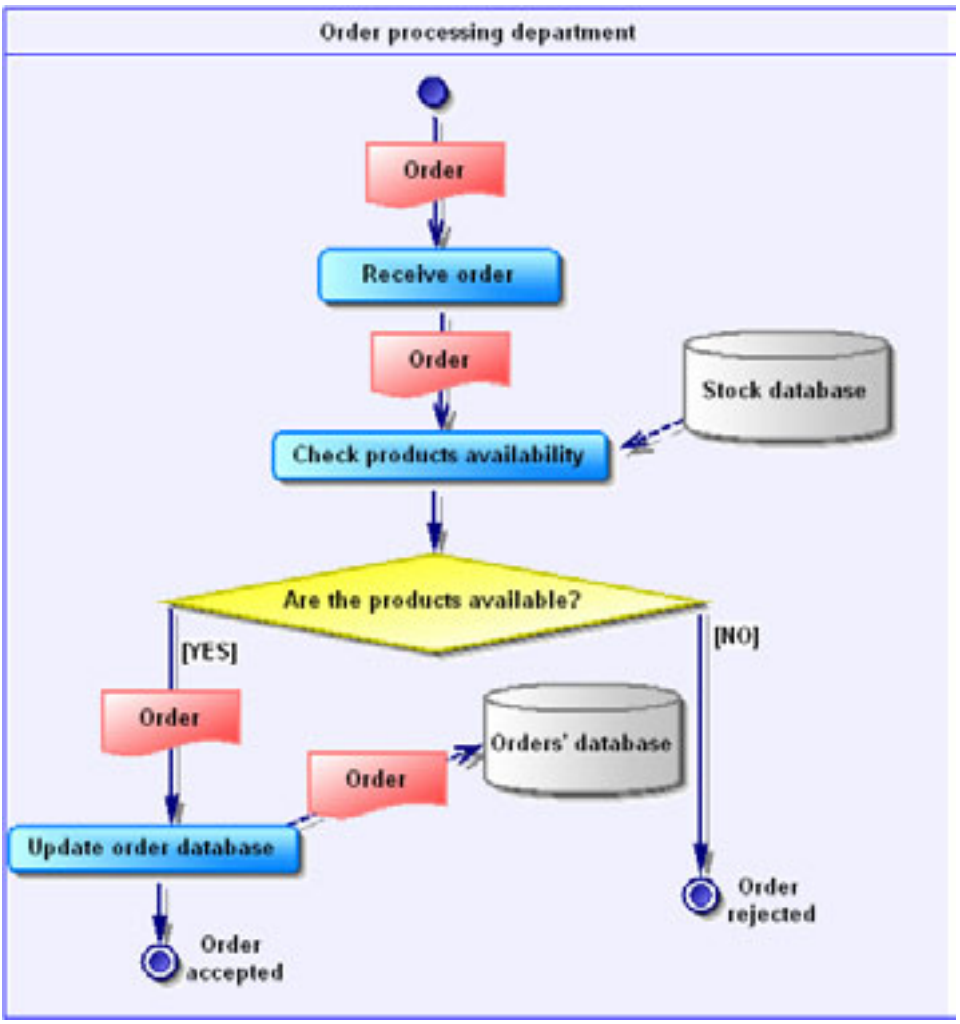

Figure 3: Business process flow diagram for 'take order' process

Even a very quick glance at the Figure 3 gives us much more information, than it was in case of hierarchic diagram. When defining process flow we have much bigger palette of objects, which allow representing dynamic aspect of any kind of business process. Within these objects we can find many elements typical for standard flow diagrams, among them: organization units (representing company department or single person, who is responsible for executing specific process), starting and ending points, decision points, branches and synchronizations, and of course process objects. There are also two additional elements: message formats and resources. The latter is quite obvious – as a resource we can understand any data storage or data supplying system (mostly databases but not necessarily). But the most important are message formats, which can be defined as 'format of a piece of information exchanged between business processes. The format depends on the nature of the exchanged information and allows objects to find an agreement to communicate' [PDUG]. Why is it so important? The answer is simple – this element is mainly responsible for getting SOA into action. To be able to use Service Oriented Approach we need to define format of messages, which are exchanged between a process and its service provider (object which delivers services that are used for process implementation). This can't be done properly without specifying messages exchanged between processes first - a process can't provide its service provider with data, which isn't delivered to the process from one of the resources or from some other process.

Message formats can be specified in several ways, e.g. by defining message parts using simple data types or by defining a company form, which is well known inside the organization. However the easiest way of doing this seems to be defining XML documents, such as XML Schema or DTD (Data Type Definition), which are great for describing data structure.

To sum up analysis Business Process Modeling we'd like to emphasize prime goal, which should be realized at this stage – designing company organization by defining decomposition and choreography for top-level business processes. For the reason that analysis models impose less constraints than execution models, they're perfect for presenting business requirements, the way business processes work and how the new system should work. High level of abstraction and independency from implementation considerations allow very close cooperation with stakeholders and end-users, which is necessary to streamline the communication between business and IT. Analysis business process model belongs to the easiest artifacts to build, however most of the time it appears to be very helpful in optimizing operational costs, integrating data and other information with existing applications, as well as managing internal and cross-enterprises processes.

4 XML Model as support for Business Modeling

Previous section for the first time in this paper directed our attention to XML format, which appeared to be useful to specify message formats exchanged between processes. Of course it's not the only element in business modeling, which utilizes XML format. Among the available tools offered by PowerDesigner we can find dedicated XML model, which allows graphically editing and representing the following XML files: XML Schema Definition, Document Type Definition and XML-Data Reduced.

XML model provides two perspectives: tree-hierarchy and XML diagram. We need to remember that there are two things to consider when modeling XML: the data we want to model and how we want to write the XML document, which can be represented in many different ways by different schemas but contain the same structure. Both perspectives provide global and schematic view of all the elements making up our model, which is extremely useful, when we need to quickly check, edit or interpret any XML element.

To show how to model XML elements and then utilize them in business model let's build XML Schema file, which could represent 'Order message format' for our 'Order Processing System' example. We need to start with creating new XML model, in which we have to define XML element 'Supplier Order'. All we have left to do is creating XML diagram for our new element (using this diagram we have to build hierarchical structure representing content of our 'Supplier order' element). Among the available objects, which can be used for creating XML diagrams, we can find for example: 'xml elements', different kinds of groups (sequences, sets, choices…), attributes, references and entities. Figure 4 illustrates possible structure, which defines 'Supplier Order' element.

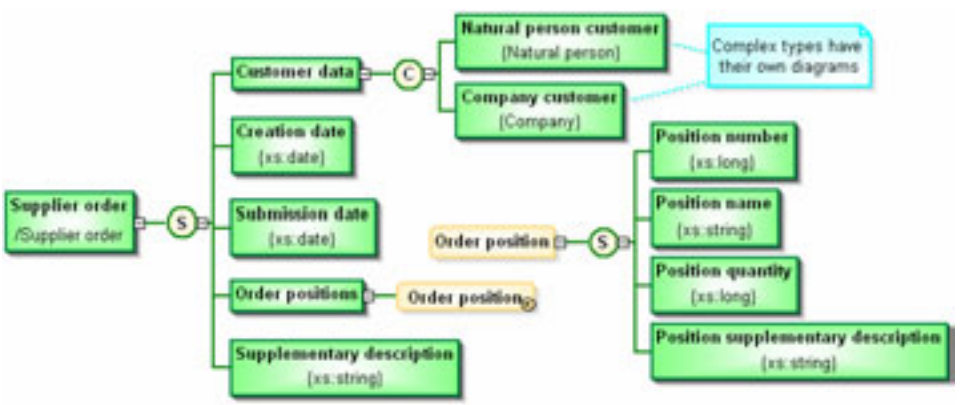

Figure 4: XML diagram representing hierarchical structure for 'Supplier Order' element

As the Figure 4 shows, XML diagram becomes an extremely lucid structure, which makes interpretation and editing much easier because it doesn't require digging through hundreds of XML tags to find interesting element. Of course every valid XML diagram can be automatically transformed into XSD, DTD or XDR file, which can be imported into any BPM and freely utilized. Conversely, we can also reverse engineer a DTD, XSD or XDR file into an XML model.

5 Executable Business Process Model (EBPM)

Building analysis business process model, which describes hierarchy and choreography for discovered business processes still makes it impossible to model, how we can use services provided by our business partners, other departments of the company or provided by potentially existing inherited system, which is to be updated. Of course Sybase architects didn't pass over this subject and from version 11.0 PowerDesigner platform includes Executable Business Process Model (EBPM), which becomes the element, where SOA shows the most of its potential.

The most popular approach to building EBPM starts with generating existing analysis model into EBPM for SOA execution language, which remains abstract from any execution platform and at the same time allows implementing elementary processes. This implementation is done by designing choreography for activities, which make up a single elementary process. The activity, which is responsible for getting SOA into action, is of course 'external service execution', that can be:

- the call of a stored procedure in a database
- the call of a component in an application server
- the exchange of a message with an external partner

To make use of existing services we need to import appropriate WSDL file. PowerDesigner processes WSDL files in a way, which converts Web services into corresponding service description objects and EBPM objects. However only the abstract description of Web services can be imported: messages, operations and port types. Among the objects, which are created in EBPM are: service providers (objects that gathers a set of service interfaces, for which they represent a namespace), service interfaces (objects that gathers a set of operations, for which they represents a namespace) and operations (objects that describe the implementation of atomic processes).

WSDL elements	EBPM objects
Message	Message format
PortType	Service interface
Operation	Operation
WSDL file	Service provider

Figure 5: WSDL elements and their corresponding objects in EBPM [PDUG]

When we don't have existing Web services, which could be utilized to implement an activity, instead of importing WSDL files we can define operations manually. We need to start with creating service provider, then define a service interface for it, and within the interface define operations by specifying type of the operation and required message formats. Available operation types are: one-way operation, request-response, solicit response and notification. The differences between them reside in the presence and the sequence of input and output messages. Naturally operations defined manually can't be executed until appropriate components, which implement these operations, are deployed. That's why typically defining operation manually becomes a trigger to create component in Object Oriented Model (Figure 6 presents typical schema for implementing processes in EBPM).

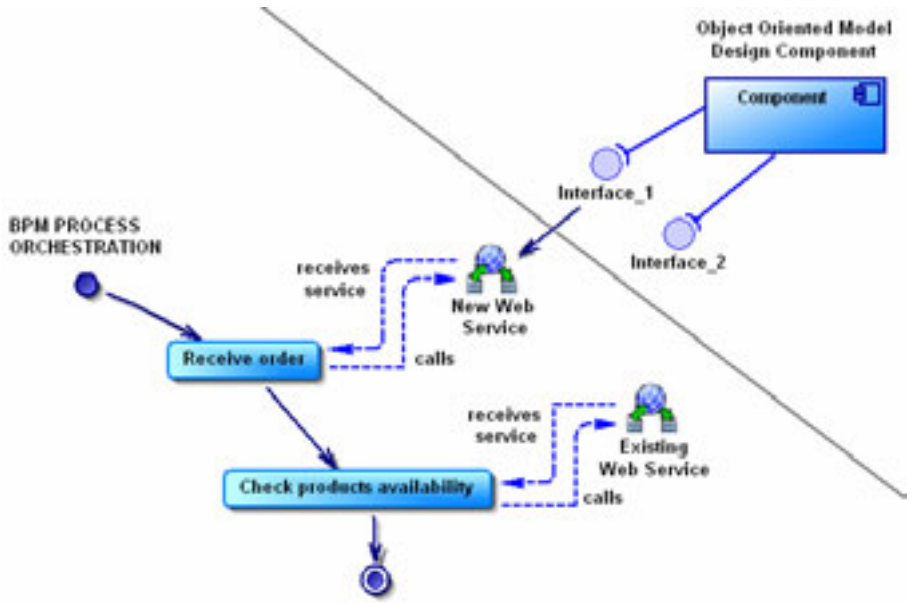

Figure 6: Typical schema for implementing business process in EBPM [PDUG]

Once again we'd like to use our 'Take Order' process to present a possible implementation. Let's suppose that we don't have any existing Web services, which could be used, that's why we have to start by defining a service provider, which will represent our 'Order Processing System', and add an interface (just to make it more real let's have two interfaces: one for public customers, and one for enterprise members, which provides some extra logic to receiving order – Figure 7 presents defined structure). Now we can define required operations within the interfaces. For every single operation we must provide input message format and output message format – once again XML model might come in handy. Of course if we had an existing system instead of creating new service provider we could import appropriate WSDL files (alternatively use UDDI server) and simply pick required services from the list of available ones. Once we have all the necessary services, all we have left to do is assigning services to corresponding elementary processes.

Practically at this level of abstraction this is as far as we can go. To be able to perform more detailed implementation we need to transform our EBPM to another one choosing less abstract language. So what exactly are the advantages of building EBPM for SOA? Every EBPM makes it possible to utilize effort, which was put into analysis level and go further with our design by creating basic implementation architecture. We should think of it as a 'bridge' between analysis BPM and one of the execution target platforms.

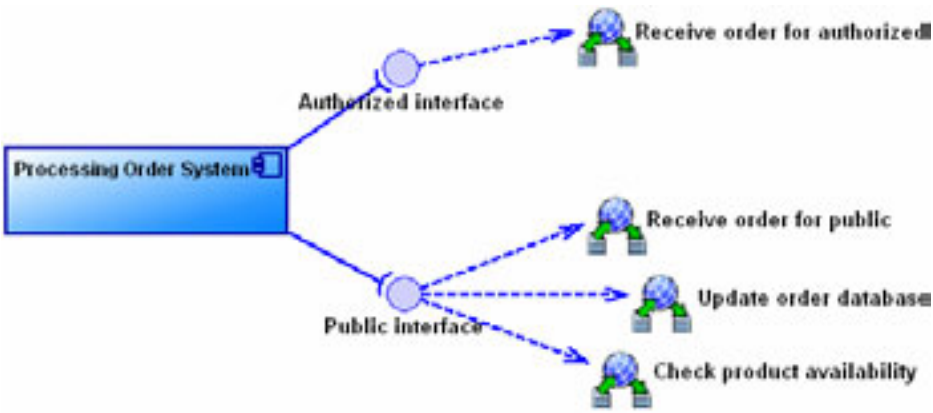

Figure 7: 'Processing order system' as a service provider and its related objects

6 Integration of PowerDesigner and Unwired Orchestrator

Now it's the time, when we need to decide, which executable language we want to use for generating EBPM for target execution engine. We can choose to generate for different business languages like Sybase Integration Orchestrator, Sybase WorkSpace Business Process, or Business Process Execution Language for Web services. This generation enables several levels of abstraction so the analyst can regenerate the executable model several times; it also performs some, but not all, transformations to adapt the source model to the constraints of the selected executable process language. In this article we'll concentrate on PowerDesigner's integration with Unwired Orchestrator (Sybase's composite application engine).

It's really hard to clearly define border between PowerDesigner and Unwired Orchestrator but typically the former is used for designing and the latter for refinement and deploying (Figure 8 presents business modeling phases with its corresponding tools). Choosing target process engine finally gives us full control over process implementation. All the stereotypes used in EBPM for SOA have to be regenerated to assure compatibility with Orchestrator, because at this level we can use only object types supported by chosen execution platform. The three most important concepts (except those presented earlier) at this level are:

- Map - Defines a data transformation using an XSLT file. Used within a business process, it defines a map activity.

- Service interaction - Invocation of one operation defined by a service. It defines which XSD files are used to match the inputs and/or outputs of the operation. It is defined in an interface file (.iface). Used within a business process, it defines a service interaction activity.
- Business process context - Set of variables that defines a state of the process.

Looking at the above list we can see that at this level we can finally put the whole process execution together by defining necessary variables and transformations to share data within complex processes. At this stage we also have a lot bigger tool palette. Among the available tools we can find such new elements as:

- for control flow designing: loops, breaks, delays, splits, rules, joins
- for error handling: faults, compensates
- for context defining: maps, assigns

Once again we'll use our 'Take Order' example for illustrating a possible implementation (Figure 9 presents the result). At this level it is pure implementation, we need to choose necessary elements and specify their attributes providing as much details as possible. As soon as our EBPM is valid we are ready to export it to our integration engine using automatic generation mechanism for Unwired Orchestrator code. Next we have to open up the generated project file in Unwired Orchestrator and deploy the business processes. Once deployed they can be automatically tested using a set of previously prepared XML documents containing test data.

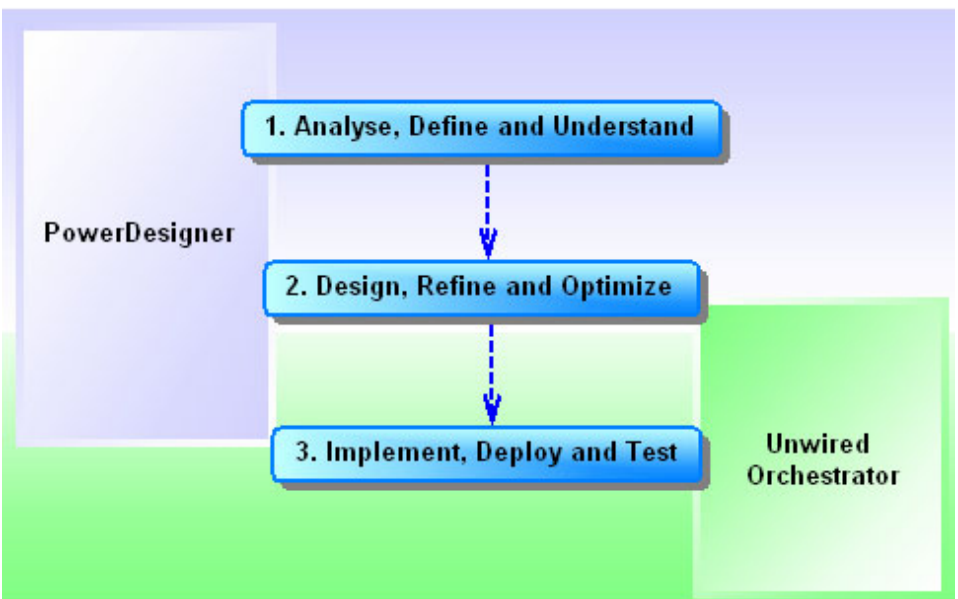

Figure 8: Business modeling phases and its corresponding tools [SWUG]

To sum up executable business process modeling we'd like to emphasize that although this part of modeling appeared for the first time a short time ago; at the time being it becomes the most important part of the whole system development lifecycle. Thanks to the incorporation of SOA, Web Services, WSDL, etc.; EBPM is expected to revolutionize the use and benefits of software modeling. Looking at companies like Sybase we can observe a methodological paradigm shift. So far we were lost somewhere between data and application modeling, however these days are over - nowadays technology is finally capable to push software development into a higher level and put business in the first place.

Figure 9: Possible Orchestrator implementation for 'Take Order' process

7 Going even further

Every enterprise organization needs to quickly respond to business events as they occur within the context of a business process, that's why enterprise business modeling must support collecting, analyzing, and displaying business data. So far we didn't provide any information on, how to monitor business processes during execution. We should remember that this might be crucial for discovering disadvantageous issues in our business architecture such as: bottlenecks, leaks and so on. The easiest way to monitor our business is using Business Activity Monitoring (BAM) included in BizTracker, which is an integral part of Unwired Orchestrator and provides comprehensive monitoring and auditing functionality. Presenting all the features of BAM and BizTracker would be a great subject for an individual article, that's why in this paper we decided to list some of the key functionalities providing only a very short description. Among BAM's features we can find for example:

- Message and data monitoring – we can monitor process execution having 'anytime' access to events, messages, and data used within the process regardless of source.
- Defining metrics and performance indicators – we can easily define any kind of a metric or key performance indicator (KPI), moreover we can define events based on our metrics.
- Visual presentation (customizable dashboard) – for every metric or KPI we can define visual control (graph, chart or table) and put it on a web-based dashboard (accessible through standard Web browser).

- Repairing and reprocessing failed messages – when we discover a failed message we can manually correct it and once again put it into message queue to reprocess.
- Probe technology – for sending process data to BizTracker we can use predefined probes or manually define our own ones.
- Alerts and notifications – BizTracker can be also used to generate alerts and notifications for mobile phones, wireless PDA's, pagers, email and so on.

Figure 10: BizTracker's architecture

We should remember that to be able to fully utilize Orchestrator's monitoring features we need to fulfill a few technical requirements, which arise from BizTracker's architecture. For example, as shown in Figure 10, Business Process Monitoring requires using database management system, which supports messaging and database events technology. We won't have this problem if we use Sybase's native DBMS – Adaptive Server Enterprise, which can get the most out of BizTracker. However decision to use some other DBMS platform may reduce available features.

Let's incorporate Business Activity Monitoring to our example, just to show a possible application and result. We could start by defining a performance indicator to monitor how many messages (in our example 'orders') per second can be processed by 'Take Order' process. For our newly created indicator we must also define a chart and put it on a dashboard. More interesting example is defining a new variable within our process, which could indicate message status:

- a message, which is being processed at the moment should have 'processing' status
- a message, which resulted in accepting order should have 'accepted' status, otherwise (resulted in rejecting order) should have 'rejected' status

Now we must define a chart presenting the number of messages in each state and same as previously add it to our dashboard. To perform execution we can define set of XML documents representing 'orders', put them into message queue and start the process execution. Figure 11 presents a possible view of our dashboard after couple seconds of execution.

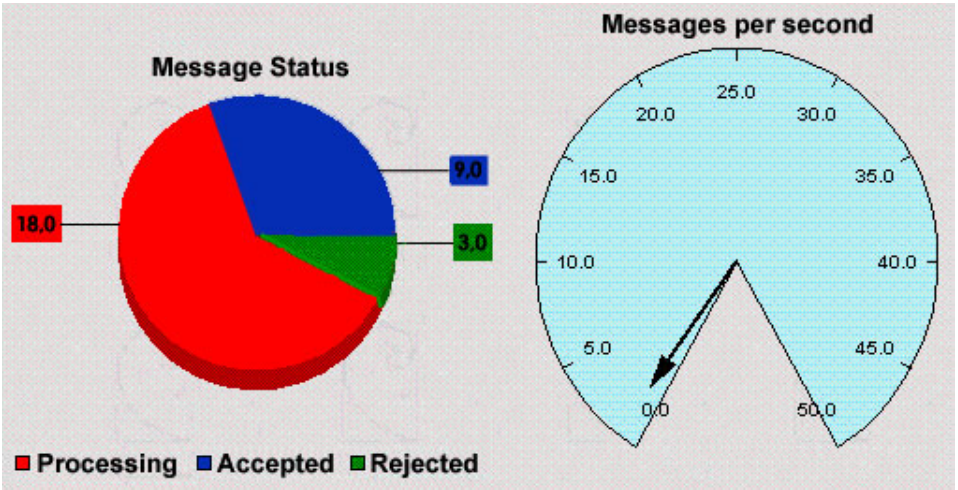

Figure 11: Possible dashboard view generated by BizTracker

8 Summary

This short article describes practical framework for modeling and deploying business processes. The main goal in creating this framework was to separate business modeling and management from application logic. We need to emphasize that although we didn't mention a single word about writing code, we were able to build enterprise business architecture and deploy it into executable platform. That's why we must clearly admit that using Sybase technology allows us to think of an IT system as three separable elements: data management logic, business logic and application logic. Having in memory revolution triggered by separating data management aspect, we should expect major explosion of new approaches, which put business requirements in the very foreground. We should also expect increase of interest with business modeling tools – in 2005 Sybase carried out a survey, which showed that 37% of developers declared using business modeling tools and another 18% declared investing in such tools within a year [HE2006]. At the time being it is estimated that at the moment business modeling tools are vastly used in 56% of IT developing companies, and by 2009 this number is expected to be just over 70%.

Such rapid development in BPM wouldn't be possible without the appearance of new technologies such as Web services, process description languages, business execution engines, etc. Once all the necessary technologies became grown enough, many different companies tried to utilize them for moving BPM to a higher level. Solution proposed by Sybase seems to push BPM further than ever, moreover it could clear the way for other incoming solutions, which may arise from Sybase's Business Driven Approach.

Right now business modeling is developing faster than ever before, which makes it almost impossible to predict the direction it will follow. The greatest challenge seems to concern an increase in the degree of automation in business process modeling through incorporation of Semantic Web services and AI technologies - we should expect appearance of the first commercial solutions based on Semantic Web services in the immediate future.

References

[SYB] Sybase official portal, http://www.sybase.com

[PDUG] Sybase PowerDesigner User's Guide, v.12.0

[SWUG] Sybase WorkSpace User's Guide, v.1.0, Business Process Integration chapter

[SD05] SOA: Delivering on the promise of Business Driven IT, www.sybase.com/detail?id=1040323

[GO05] Unwired Orchestration, D. McGoveran, Alternative Technologies, 2005, www.sybase.com/content/1036885/UnwiredOrchestration_WP.pdf

[IM05] A. Ivanyuckovich and M. Marchese "Service Oriented Architectures for Business Process Management Systems", in IADIS International Conference on Applied Computing., Algarve, Portugal, February 22-25, 2005

[GO04] Gold, N., et al, 2004. Understanding Service-Oriented Software. In IEEE Software, No.3

[HE06] Data Modeling: A Gateway to Enterprise Architecture, Stephen D. Hendrick, March 2006 IDC