24

Composite Supply Chain Applications

Thomas Gulledge, Scott Hiroshige and Danielle Manning
Enterprise Integration, Inc. (EII)
USA

1. Introduction

A number of commercial software vendors sell supply chain software suites that cover essentially all needs of the enterprise. For example, a vendor's product can handle everything from creating an order, to logistics planning for that order, to logistics execution of that order, and finally to financial settlement. Allowing a single software suite to enable all supply chain-related transactions has some significant benefits including reduced integration costs, improved data integrity, and increased process optimization; however, the reality is that many organizations explicitly choose not to perform all of their transactions in a single software suite. Instead, supply chain processes are almost always executed across a heterogeneous system landscape, often involving communications among systems that were not designed to communicate with each other.

Akin to the system landscape decision is the decision of how to implement business processes. There are some business processes in which no competitive advantage is gained from "doing things your own way"; for example, Enterprise Resource Planning (ERP) software has optimized the processing of payroll to the point where a customized payroll process probably will not give you much of an edge over the non-customized competition. However, there are other business processes where innovation can provide a competitive advantage; for example, in the 1980s Wal-Mart enhanced its logistics operations with cross-docking and gained an operational advantage over other retailers. Process innovation is widely acknowledged as a means of increasing business value (Davenport, 1992). In such cases, a single unmodified commercial software product might not support the customized process, and the question becomes how to best develop a solution supporting the customized process while keeping interfacing and interface maintenance costs under control.

In recent years, a number of enterprise software vendors have put forward offerings in the genre we call "Model-to-Execution." These offerings provide a viable means of designing and implementing custom solutions in a manner that is economical in terms of both implementation and maintenance costs.

We begin by presenting our hypothesis and briefly introducing the concept of Model-to-Execution. We then discuss the case study that is used to test the hypothesis and the solution that was designed and implemented via Model-to-Execution. Finally, we describe the benefits of Model-to-Execution for organizations and discuss some of our lessons learned from testing the hypothesis.

2. Theoretical discussion

As described in the introduction, modern supply chain solutions are automated. There are two possibilities: the supply chain business processes could be automated with a single software product or with multiple software products from different vendors. Some companies have elected to use a single vendor, but most companies use products from multiple vendors. Our research focuses on an efficient and effective way to implement supply chain software in a multiple-vendor environment.

Our primary hypothesis is that logistics business processes can be described in business terms and fully automated using Model-to-Execution software solutions. To test this hypothesis, we perform an actual implementation project across multiple vendor components. This is the primary scientific contribution of this paper.

3. Model-to-Execution (M2E)

Model-to-Execution (M2E) is a methodology for designing and implementing software solutions that support business processes. "Model" refers to business process modeling, the practice of analyzing business processes and thoroughly documenting them, typically in step-by-step process flow models. However, a business process model does not execute business processes; such models only illustrate how operations "should" work. This is where "-to-Execution" becomes important. Technologies now exist for converting business process models into running workflow code, and these running workflows combined with humans, Graphical User Interfaces (GUIs), and software services can in fact execute business processes.

Figures 1 and 2 show, in part, the similar approaches taken by two M2E vendors: Oracle and Software AG. We note the common elements "Implement," "Execute," and "Monitor", as well as doing either "Model" or "Strategize" and "Design" prior to implementation. We know from experience with the Oracle and Software AG product offerings that many of the software tools in their suites are also analogous across the two vendors. In fact, the same business process modeling tool is used in both vendors' M2E offerings—the ARIS Platform from IDS-Scheer. (IDS-Scheer was purchased by Software AG in 2009, and the ARIS product is included in the Oracle Fusion Middleware (OFM) suite under the name Oracle Business Process Analysis (BPA) Suite).

A key component of Model-to-Execution is the business process modeling. Business process modeling is performed primarily by business analysts and serves several purposes. First, because a process is executed across organizational stovepipes, it assembles all relevant business process stakeholders into the same room, something that does not occur often enough in many organizations. Second, it drives agreement. Forcing all constituencies to collectively draw a single representation of the solution helps ensure that differences in vision are worked through and agreed upon before the implementation is started. Third, it provides a significant portion of the content for a requirements document for describing the desired solution to the implementation team. If the purpose of a development initiative is to build an executable business process, then a detailed diagram of the business process is necessary to have in the requirements document. Finally, in all Model-to-Execution packages that we have worked with to date, the business process model is transformed into the skeleton around which the rest of the solution is built. Without the business process model the rest of the solution cannot be built using this methodology.

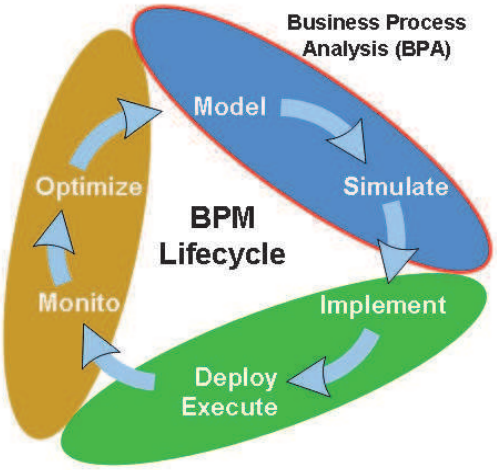

Fig. 1. Oracle BPM [Business Process Management] Lifecycle (Oracle Corporation, 2008)

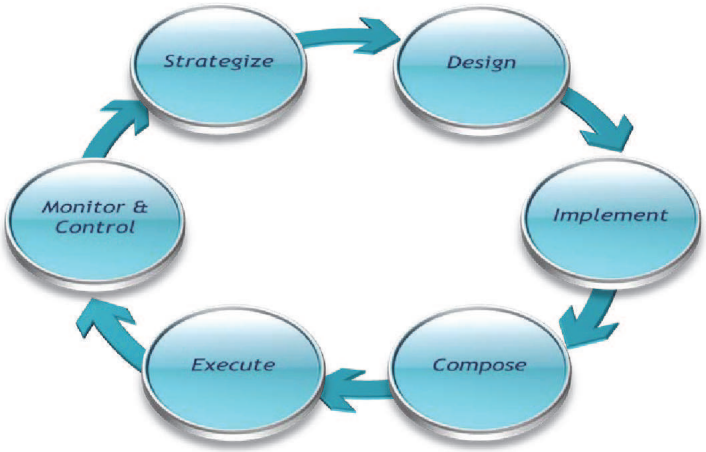

Fig. 2. Software AG BPE [Business Process Excellence] Lifecycle (Jost, 2010)

Conversion of the business process model into executable software occurs in several steps that vary by M2E platform. This conversion is performed primarily by technical staff since the work involved is to some degree (again varying by M2E platform) like "coding" of software. However, as mentioned, the business process model is transformed into the skeleton around which the rest of the solution is built, helping to ensure that the technical resources are building the solution envisioned by the business analysts. An analogy is that the technical resources attach muscles (software engines and services), brains (business rules, software logic, etc.), nerves ("alerts" that monitor Key Performance Indicators (KPIs) and notify people when significant events occur), and skin (graphical user interfaces) to the skeleton.

Figure 3, by Gulledge (2010), illustrates this process of converting a business process model into executable software. The left side of Figure 3 shows the business process model (labeled as "Business BPM"). The business process model is turned into a skeleton, depicted in the center of Figure 3 (labeled "Technical BPM") prior to adding additional functionality (the muscles, brains, nerves, and skin) to it. Finally, the right side of Figure 3 (labeled "Development/ Deployment") shows the skeleton as it is being "fleshed out" with the additional functionality. Whereas the business process model on the left is just a picture and the un-fleshed-out skeleton in the center is also incapable of action, a fully fleshed-out skeleton on the right is capable of executing the business process on which it was based.

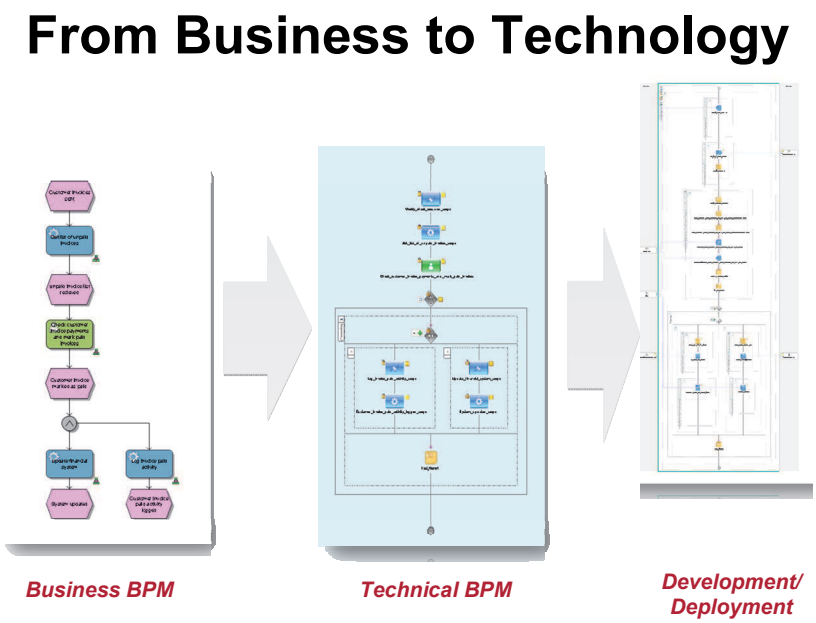

Fig. 3. Going from Model to Execution (Gulledge, 2010)

Although this is not a requirement of M2E, Model-to-Execution lends itself readily to Service-Oriented Architecture (SOA). This is in part because the SOA pattern of breaking down a large software task into smaller tasks performed by reusable Services dovetails nicely with the Business Process Management (BPM) pattern of breaking down a large business task into its smaller component tasks. SOA also embraces the idea of work being performed by multiple systems not under the direct control of the process owner, something that is often encountered in an end-to-end business process. Finally, SOA promotes the idea of loose coupling between systems, which we have found to be a good approach in our M2E implementations thus far as it allows one to quickly swap out one service or sub-process for another.

Model-to-Execution is not the ideal approach to every problem. The following non-exhaustive list contains some of the characteristics that would make a business problem a good candidate for applying the Model-to-Execution approach:

- Possibly a custom or highly specialized business process
  - Routine problems are probably already solved in a commercial software package
- The desired solution is not well-defined (perhaps it is new or innovative) and requires much analysis and agreement before implementation can begin
  - If the solution is already well-defined, then the process modeling effort adds overhead but minimal benefit.
- The business process changes frequently (i.e., the solution requires flexibility and agility)
  - The more frequently an organization desires to change a business process, the more important it is for Information Technology (IT) to adapt quickly to these desires. M2E helps with quickly communicating the desired changes from business staff to technical staff.
  - A frequently changing process leverages the M2E agility advantage more often.
- The business process involves more than one person and/or a heterogeneous system landscape.
  - We leverage the workflow aspect of M2E when we route tasks between different people and systems.
  - The functionality that a system exposes to external callers is typically less powerful and less optimized than what is available to the system internally. M2E solutions will typically call other systems from the outside, so if only one system is involved then perhaps working natively is a better solution.

4. An M2E case study scenario

Cognizant of the fact that supply chain organizations often run more than a single enterprise system, we present a case study examining the potential of Model-to-Execution for executing a supply chain business process within a heterogeneous system landscape.

The proposed business scenario is as follows: Company XYZ is a small printer and fax distributor in the United States (US). It ships printers from several warehouses around the US directly to customer locations. The company's customer delivery performance was deteriorating, and management identified a few issues in the supply chain execution process that were impacting their KPIs:

- Orders were being confirmed and released from warehouses with insufficient inventory. The company wanted to ensure that if orders could not be filled from a particular warehouse location, then inventory would be released from a different warehouse to fulfill that order. This would allow for accurate management of inventory turns.
- Customer delivery addresses were inaccurate, resulting in undeliverable shipments. The company preferred that "the system" confirm a delivery address before orders are released.

Company XYZ automate its logistics and transportation using the Oracle Transportation Management (OTM) product. However, it automate its back-office processes using SAP Business One, making this a heterogeneous system landscape. In addition, the Warehouse

Management System (WMS) is proprietary, presenting unique interfacing challenges compared to commercial products with documented interfaces. Finally, despite the desire to confirm customer delivery addresses in an automated manner, the company only has the ability to manually confirm.

This case study scenario, while not performed for any particular customer, is based on real-world observations. For example, many organizations use OTM for shipment planning and execution, and even though OTM has order-creation capability, they elect to create orders in SAP. The problem of sending items to invalid addresses frequently occurs, as does the problem of interfacing proprietary systems with the rest of the IT landscape. Finally, the integration of multiple disparate systems to enable cross-functional business processes is a problem area that that is well known in the research literature.

5. Applying model-to-execution to the case study scenario

We will now describe in detail the different steps followed in the M2E approach to create the solution. In this case study, the task of taking one from Order to Shipment is accomplished not by a single software application, but rather by a "composite application" made up of the three applications previously mentioned (SAP Business One, proprietary WMS, and OTM) and a fourth application—an address-checking service provided by the United States Postal Service (USPS). Note that the steps followed in this case study are by no means the only way in which M2E could be applied to this problem. Nor are the tools used and systems involved necessarily the tools and systems we would choose today if redoing this case study. The point of the case study is merely to demonstrate that a working solution to a real problem can be developed via the M2E methodology.

5.1 Choosing a technology stack

Model-to-Execution requires roughly the following components:

- A tool for modeling the business process. (The reader will recall that the "skeleton" from our body analogy is created from the business process model.)
- Tools for fleshing out the skeleton, turning what was "just a skeleton" into executable code by:
  - Attaching various software components to the skeleton.
  - Building new software components that are needed but do not exist ahead of time.
  - Writing business rules or other process logic needed to guide the process execution.
- An environment for running the executable code. Just as Mac programs won't run on a personal computer (PC) and PC programs won't run on a Mac, likewise M2E executable code will not run unless it's in an appropriate environment.

The tools we chose for this project are the following:

- Oracle Business Process Analysis (BPA) Suite for modeling the business process, and also for converting the process model into a skeleton.
- Oracle JDeveloper for fleshing out the skeleton.
- Oracle Application Server (OAS) for executing the code produced by Oracle JDeveloper.

These three components are a part of what Oracle brands as Oracle Fusion Middleware (OFM). The inherent integration of the various suite components was one of the primary drivers for choosing to take the development stack from a single vendor (not to be confused with taking the entire application stack from a single vendor). Minimal effort is involved in

converting a business process model into a skeleton, and it is likewise easy to move the fleshed-out skeleton into the execution environment. We could have used the same logic to select the Software AG development stack instead, and indeed have done so on other M2E projects. Rarely would we choose to take development stack components from multiple vendors, as this often means extra effort on the part of the development team to make the components of the stack work together.

There is also another problem associated with using components from multiple vendors. Kemsley (2010) says that "using separate, non-integrated tools created a communication barrier between business and IT."

5.2 Assembling a development team

The project team was small: one business analyst (an experienced supply chain consultant), one technical person (an experienced software developer), and a few Subject-Matter Experts (SMEs) (technology consultants, supply chain consultants, former warehouse operations employee). The whole solution was essentially built by two people—the business analyst and the technical resource—in the span of a few months, not by an army of programmers over the course of several years.

The small team size is an advantage of this approach, and is enabled by several factors. Among these is the fact that we are not building an entire solution from the bottom up. Rather, we are leveraging many things that already exist and just building the parts that do not exist. For example, it would take a long time to build a shipment planning feature from scratch, so we instead leverage the shipment planning feature already built in OTM. However, the Warehouse Management System had no feature for rerouting an order in the case of insufficient inventory, so we had to build this feature from scratch.

5.3 Business Process Modeling

The Business Process Modeling effort was led by the business analyst, with the technical resource having only a supporting role in the effort. Through several process-modeling workshops held with the SMEs, the business analyst obtained an understanding of how the SMEs wanted the business process to work, modeled the process in Oracle BPA Suite, and validated the model with the SMEs to confirm that her understanding was correct. Figure 4 shows the result.

Those not familiar with the Event-driven Process Chain (EPC) notation used in this process model can think of a green or blue rectangle as an action ("Function") performed during the business process, and a pink hexagon as an "Event" that triggers the next step in the process. The blue rectangles with gear icons represent fully-automated steps in the process—"the system" should execute these actions without any human interaction. The green rectangle in the middle-right with the person icon is a manual human task—a human must take some sort of action (in this case, reviewing the order and either making corrections or rejecting it outright), often through a GUI. The other green rectangle in the lower-right with the mail icon is a notification step—"the system" sends an email notifying someone of something (in this case, "The order is being canceled").

The short description of the process is as follows: an end-user enters an order into the SAP Business One system. Two process steps are then executed in parallel: the delivery address specified in the order is automatically verified against a system run by the US Postal Service, and an automated check for and subsequent reservation of inventory is made against the

proprietary Warehouse Management System. Assuming these process steps encounter no problems, execution follows the left leg of the model where the order is moved into Oracle

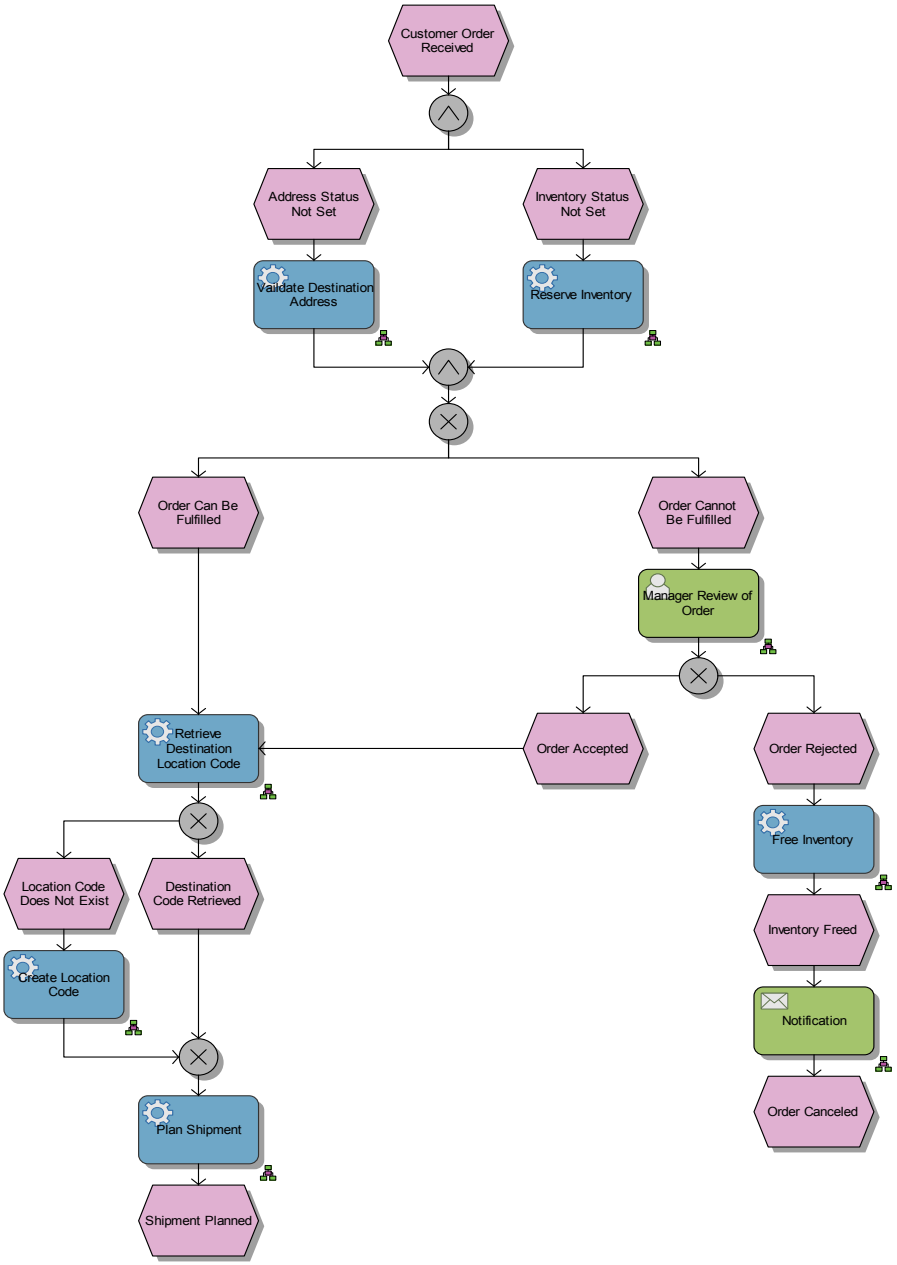

Fig. 4. Business Process Model

Transportation Management (OTM) for shipment planning and execution. (The Functions "Retrieve Destination Location Code" and "Create Location Code" get or create technical information required when putting an order into OTM.) However, if either the address is declared invalid or there is insufficient inventory to cover the order, then execution follows the right leg of the model and enters the Human Step. A screen built specifically for this case study enables human intervention in hopes of resolving the problem. In resolvable cases (e.g., a mistyped ZIP code), the human can make a fix and send the order down the left path to OTM. In irresolvable cases (e.g., a delivery address containing a city, state, and ZIP code but no street address), the order is canceled and order owners are notified accordingly.

Note that various M2E solutions may use different modeling notations. Two that we commonly encounter are the EPC notation used above, and Business Process Modeling Notation (BPMN). The choice of notation is driven primarily by organizational preference and what formats the modeling and conversion tools support. The oracle BPA Suite supports both EPC and BPMN notation, and we have worked on projects where the process modeling was first done in EPC for one audience and converted into BPMN for a second audience. The notation doesn't necessarily matter so long as the SMEs and modelers understand the notation and the model can be converted into a skeleton.

5.4 Moving to the development environment

As mentioned in the previous section, the Oracle BPA Suite was used to model the business process; however, the model is not executable (i.e., one cannot click a "Go" button and watch an order move from SAP Business One into OTM). For a model to be executable, the model must be transformed into a skeleton and fleshed out with technical components. The Oracle BPA Suite is not able to flesh out the model, but it does handle converting the model into a skeleton.

In this project, we created a skeleton in the Business Process Execution Language (BPEL) format, as this is a format that can be easily imported into Oracle JDeveloper. Other tools may use other skeleton formats; for example, if using Software AG's technology stack, we would convert the EPC into BPMN, and the BPMN to XML Process Definition Language (XPDL)—the XPDL would then be imported into the development environment as the skeleton. In the toolsets we have worked with to date, these conversions have been almost entirely automated—simple copies-and-pastes or right-clicks or wizards make for a painless process of creating the skeleton from the business process model. In this case study, the conversion is initiated by choosing the menu option "Share Blueprint with IT" in the Oracle BPA Suite.

Finally, the skeleton must be moved into the development environment. Our development tool, Oracle JDeveloper, provided a menu option for importing the skeletons from the Oracle BPA Suite database. The specifics of moving a skeleton into a development environment will vary by toolset, and we have seen various vendors making this part of the process more streamlined across new toolset releases.

5.5 Working in the development environment

Once the skeleton was in the development environment, the technical resource's work began in earnest. As the reader will remember, the skeleton itself does nothing but provide shape and structure; it is the muscles that do the work, the brain that coordinates the actions of the muscles, and so forth.

5.5.1 The process trigger

One of the first things built was a software component that determined when the business process should execute. In this case study, we execute the process when a new order is created in the SAP Business One system, so we wrote code (or the BPEL equivalent) to occasionally poll SAP Business One and determine when a new record had been written to the Orders table. Once a new order was identified, the relevant information about the order (what was ordered, how much was ordered, to where the order would be delivered, etc.) was queried from the database for use by the rest of the business process.

The above approach to retrieving the order would be considered a "pull" of information. Alternatively, we could have extended SAP Business One software to "push" the new order to us as soon as it was created. The business result is the same: the process is initiated with all the information needed to execute the process. However, there are technical and social considerations that are not the same. For example, the owner of the SAP Business One system might object to your continual polling of her system and prefer to push you the data. Or, alternatively, she may prefer to have you poll the database rather than modify the existing SAP Business One implementation. Thus, while this is a trivial decision from the business perspective, there are technical and social factors that should be considered when determining how the process will "know" to begin execution.

5.5.2 Data considerations

In the executing process, data are passed to each process step; for example, in order for the Address Validation step to execute, an address must be passed in. In some cases, a process step also passes data out; for example, the Address Validation step must signal either "Valid" or "Invalid" so that the process can proceed down an appropriate path. This raises several questions, including:

- What data must I pass in?
- Do I have all the data I need to pass in, or are there missing items that I need to obtain before calling the next process step?
- What data will be passed out?
- Of the data that was passed out, which data do I actually need to execute this process and which are extraneous to this process? (For example, consider listening to the entire weather report when all you really need to know is tomorrow's high temperature.)
- Do I have the data structures needed to hold the data I will work with?
- What data formats are used by the process steps I am calling? What formats should I use for my data structures? If they are not the same, how do I convert between them?

Similar questions arise for branches in the process:

Which field or combination of fields tell me which branch to go down?

- If the decision criterion is complicated or may change in the future, should we employ a business rules engine to, for example, simplify complicated decision criteria into a simple "Go Left" or "Go Right" flag?
- Is such a flag already in my data structure, or do I need to create one?

These questions relate to discussions about enterprise data models, canonical data models, etc. that we will not get into in this chapter. A few of our suggestions relating to data are:

- If your organization has an enterprise data model, canonical data model, etc., then the interfaces (inputs and outputs) to your process should follow the standards defined in them.

- If a field is extraneous to your process, consider excluding it as an input. However, consider including it if it is part of a data structure containing other useful information. The idea is to make it easy to invoke your process, whether that means passing in one Purchase Order data structure or just the minimum number of fields.
- Within your process, use a process-specific data structure to carry the process from start to finish. It may be an unnecessary complexity to carry a hundred-field Purchase Order if you only need five fields off the purchase order. Likewise, you will probably find a need for fields (such as the "Go Left or Go Right" flag) that would not exist on any business process model or enterprise architecture. However, keep in mind that your process will be calling other processes and services, and any such calls made to processes and services created by your enterprise will be expecting data in the canonical data format.
- Expect defining the process-specific data structure to be an iterative process. Despite our best efforts, we rarely anticipate 100% of the structures, fields, and flags needed prior to writing our first line of code.
- When calling other processes or services, accept their input and output formats as given and map to them, even if your data representation is "better." It is rare that the other party will change to accommodate you, especially if they already have other users of that data.
- As you implement, search for opportunities to improve or extend the enterprise data model and canonical data model. It may be that you are the first to use a field or data structure that the rest of the enterprise will soon find valuable, especially if the data model is young or you are entering a new line of business.

5.5.3 Building the executable process

After analyzing the triggering of the executable process and initially defining the data structures, we began implementing the process. In general, it was easiest to start at the beginning and work sequentially. Testing was done after each component was built to ensure that it was behaving as expected, as the outputs of a component or the path chosen by a logic gate affected what happened farther downstream. Several of the different objects we built are described in subsequent paragraphs.

Address validation was performed through a web service hosted by the United States Postal Service (USPS). A "wrapper" web service was created in Oracle JDeveloper to convert the composite application's data into the format required by the USPS web service. Wiring was done in Oracle JDeveloper to connect the composite application and the wrapper web service, and the wrapper web service to the USPS web service

The proprietary Warehouse Management System posed a different challenge: rerouting an order in the case of insufficient inventory was an entirely new feature, so there was no existing functionality to leverage. Rather, since the company owned the WMS, the technical resource modified the WMS and coded this functionality from scratch. He then used Oracle JDeveloper to create a web service interface exposing this and other features of the WMS. Finally, he used Oracle JDeveloper to wire up the composite application to call the new web service.

A GUI was needed to facilitate the human task of reconciling faulty orders (invalid addresses or insufficient inventory). It was decided that the GUI would be displayed in the Oracle BPEL Worklist application, an out-of-the-box application from Oracle facilitating task execution. Figure 5 shows the Oracle BPEL Worklist displaying a list of tasks to be executed, and Figure 6 shows the GUI built for our particular human task. The actual building of the GUI was done in Oracle JDeveloper.

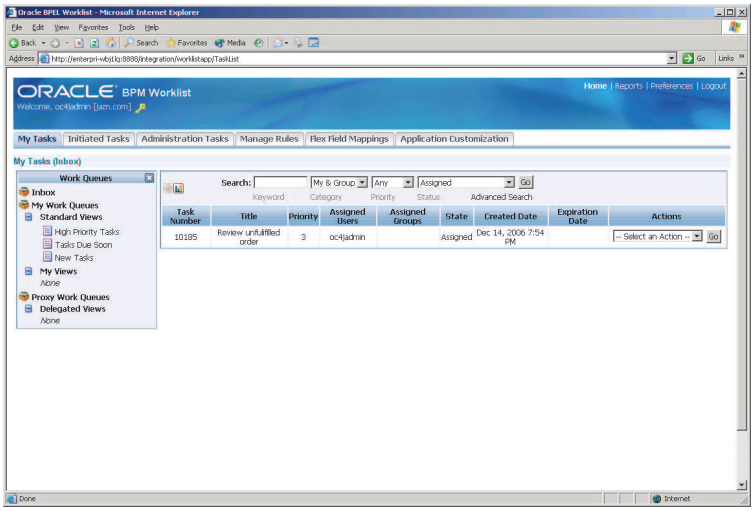

Fig. 5. Oracle BPEL Worklist

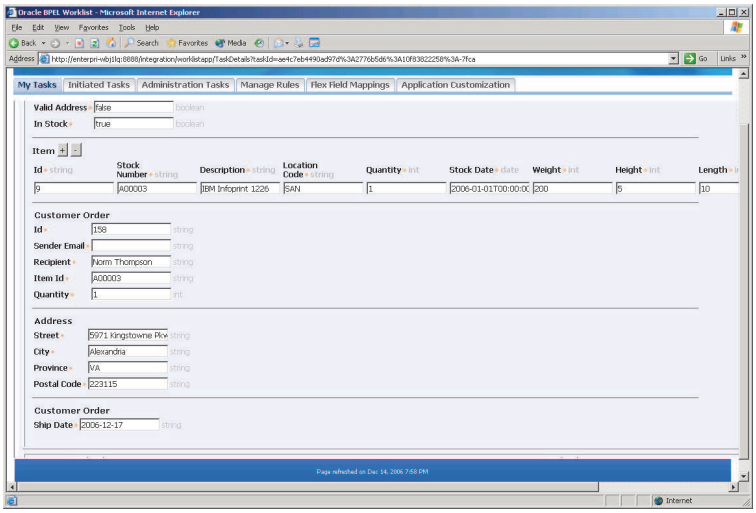

Fig. 6. Human Task as rendered in Oracle BPEL Worklist

The email notification was also created in the Oracle JDeveloper. Oracle Fusion Middleware did not send the email, but instead leveraged an external mail server. Oracle JDeveloper was used to specify the port and address of the email server, as well as any other relevant configuration information. Oracle JDeveloper was also used to define the content of the email.

Interfacing with OTM involved integrating directly with OTM's HyperText Transfer Protocol (HTTP) POST interface. We did this by wrapping the HTTP POST interface in a web service. Calls were made from the composite application to the web service, and the

web service created the HTTP POST message and sent it to OTM. All of this work was done in Oracle JDeveloper.
Wiring between the various steps was done within Oracle JDeveloper. This consisted mostly of passing variables into and out of process steps, converting between formats via eXtensible Style Sheet Transformation (XSLT) as needed. When all the components were wired up, the skeleton was fully fleshed out and the composite application was ready for deployment.

5.5.4 Testing

As mentioned, testing was done along with building in an incremental, iterative manner. We would build a feature or component, test it, and remove any bugs prior to beginning work on the next feature or component. This let us catch and fix as many problems as possible upstream, before downstream functionality was built around flawed upstream inputs. Oracle JDeveloper has a built-in feature for executing the web services locally prior to integrating with the Enterprise Service Bus (ESB) for what we might call developer-level testing, but it is recommended that one follow standard software development practice and maintain a separate environments for development and production.
In testing this type of composite application, two types of testing are very important. The first is the component test. Each sub-process or web service should be built such that it can be reused by other processes; thus each sub-process or web service must be tested in a stand-alone manner. The second type of test is the end-to-end scenario test. This runs the composite application—i.e., the executable business process—from start to finish, and ensures that the right things occur as the process is executed. Doing only scenario testing may not detect some component-level bugs if the scenarios do not exercise particular features of the component. Doing only component testing might appear to be sufficient—if all the parts work, shouldn't the whole work as well?—but often mistakes are made in the wiring between components, mistakes that tests of individual components will not catch.

5.5.5 Moving to the execution environment

The execution environment was Oracle Application Server (OAS), an OFM component that was standard at the time but has been replaced by Oracle WebLogic application server as of the 11g release of products. As both OAS and Oracle JDeveloper are Oracle products, Oracle provided out-of-the-box integration for easy transferring of code from Oracle JDeveloper to OAS. A few simple menus and wizards let us easily move each module into OAS. The modules to be moved included the web services we created (not the USPS web service, which was already available on the internet), the human task, and the application definition. Note that the components are moved separately from the application definition; they exist apart from the application definition, such that any composite application (including the one we just built) can use them if the application is so defined. Figure 7 is a screenshot showing some of the components in the execution environment. (Note that there are also other components in the environment, presumably used by other composite applications, which our application does not use.)

5.5.6 Running the composite application

Once the application definition and all application components are in the execution environment, we can run the application. Keep in mind that the application is a composite application—that is, we leverage pre-existing features of other applications rather than building our own from scratch—and that its purpose is to execute the particular business process that we defined in the scenario description.

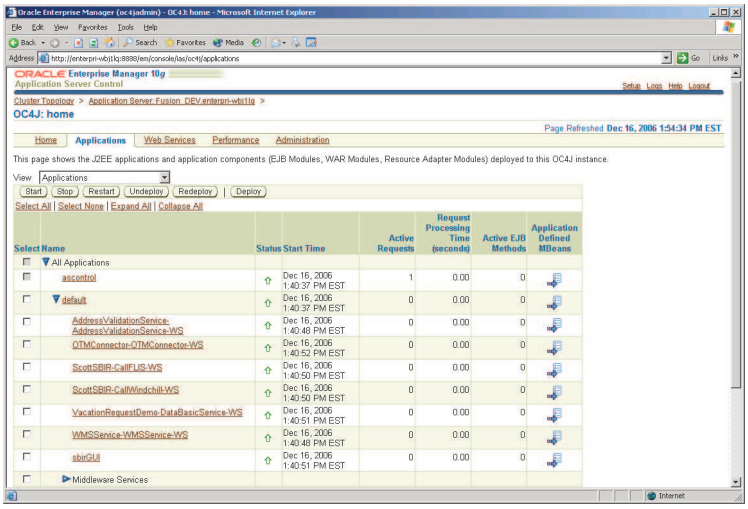

Fig. 7. Components moved to the Execution Environment

Figure 8 shows the first GUI in our application, leveraging from SAP Business One, in which an order is created. When that is completed, the database poll observes the new order and triggers downstream the process. Web service calls are made behind-the-scenes to the Address Validation web service and the WMS web service to validate the delivery address and reserve inventory, respectively. Figure 9 shows the human task that is generated when the Address Validation web service reported the delivery address as being invalid. A company employee uses the human task GUI to correct the delivery address. Finally, the order arrives in OTM, as shown in Figure 10, for shipment planning and execution.

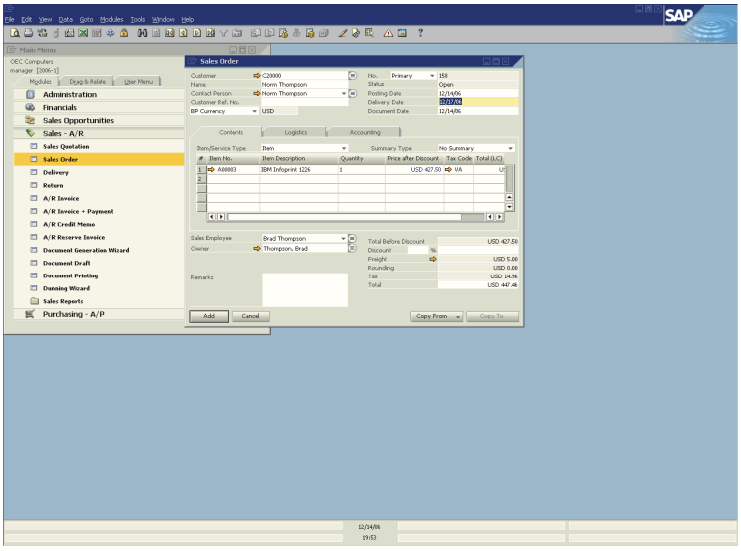

Fig. 8. Executing the process by placing an order in SAP Business One

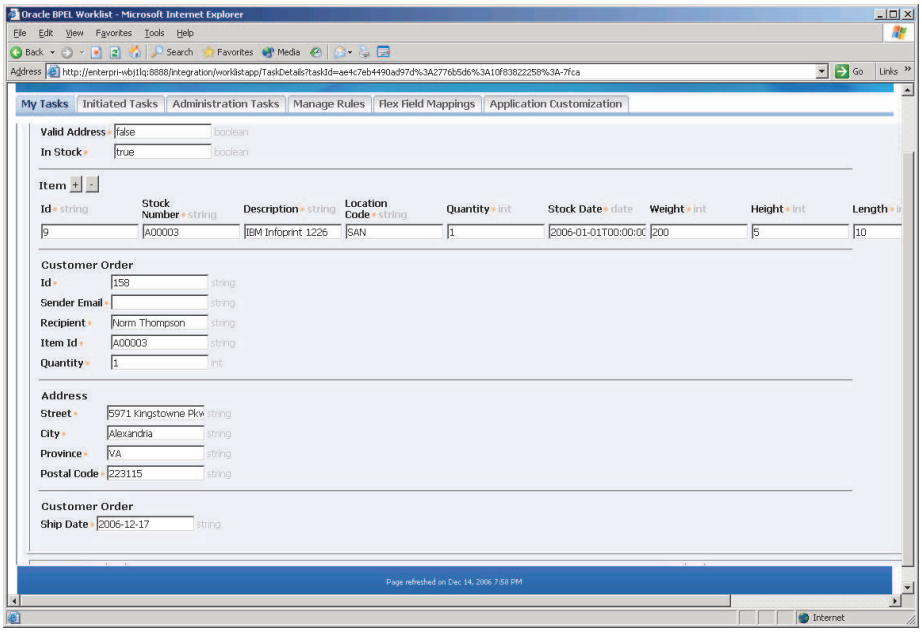

Fig. 9. Using a Human Task GUI to fix an incorrect ZIP code

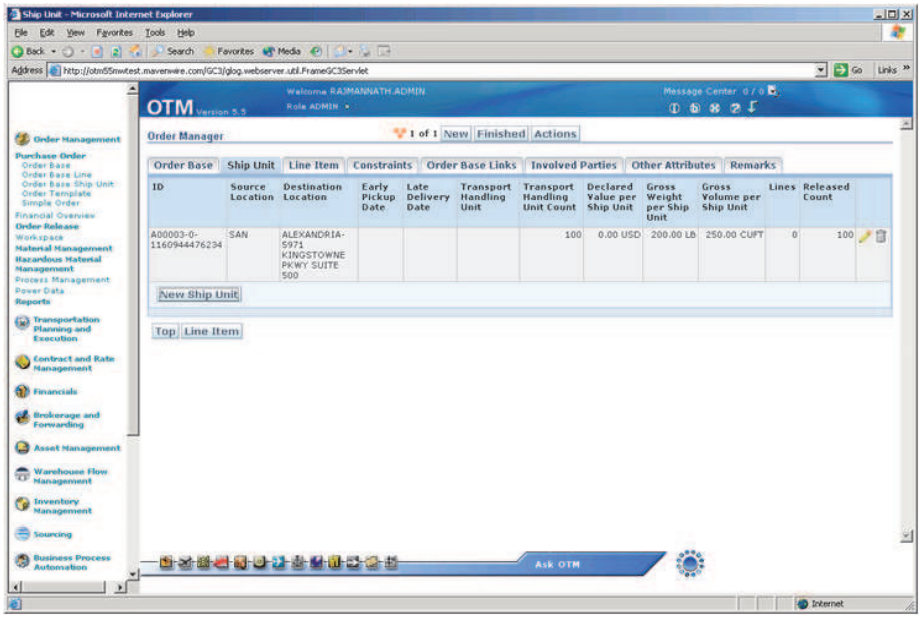

Fig. 10. Order moved into OTM, ready for shipment planning and execution

6. Discussion

In the case study, we did the following:

- Identified problems in the current business process
- Specified changes required to address the problems
- Developed a custom application supporting the changed process

An important point is that the business analyst, not the technical staff and not a one-size-fits-all commercial software package, defined how the business process would work. This let the business focus on its specific problems and do business the way it felt would be best. However, unlike many other custom applications, this application was built not from scratch but by leveraging as much functionality as possible from the existing system landscape. This reduces the time and resources required to complete the project. Finally, the composite application was built with minimal customization of commercial software, avoiding the often great expense incurred in modifying and maintaining a custom solution.

Was this a truly differentiating business process? It was probably not, making shopping for a commercial software solution an acceptable alternative to building the composite application. However, assuming that no commercial solution is found to be a solid or economical fit, developing a composite application using the M2E approach described here is a viable alternative to living with the status quo.

6.1 Benefits for supply chain organizations

We believe that Model-to-Execution offers the following benefits:

- A custom, business-oriented solution; the process executed in the composite application is defined by the business managers, not by the commercial software vendor.
- A feasible means of implementing a "best-of-breed" solution, often argued against because of the high integration costs.
- An agile solution. IT can quickly make changes because any component system or sub-process can be "swapped out" for another comparable system or sub-process with relatively little additional coding. The business can change direction more quickly because IT can change more quickly.
- The Model-to-Execution approach makes the heterogeneous system landscape a workable reality rather than a situation to be avoided.

6.2 Lessons learned

Having now completed several Model-to-Execution projects, the authors offer the following observations as "lessons learned."

6.2.1 Expect to do some wrapping

The ideal M2E (or for that matter, SOA) world has a myriad of plug-and-play web services and sub-processes already existing and available for discovery by composite applications. Many of us do not live in such a world. Rather, the functionality we need is often in legacy systems that are not service-enabled, or are perhaps partially service-enabled, but not in the parts that we want to leverage or at the correct granularity to be useful. Thus, we must first "wrap" legacy systems to expose needed functionality in a service-oriented manner.

One should go into an M2E project prepared to do at least some such wrapping. Consider each such wrapping a one-time investment that will be leveraged by any future M2E or SOA projects needing the same functionality. Initially, most of the functionality you want to use

will be unwrapped; however, over time an organization's library of services will grow and less time will be spent wrapping because another project has already done the wrapping work. The natural retirement of legacy systems and activation of more modern, SOA-enabled solutions will also result in more available services and less time spent wrapping.

6.2.2 Understand where the value comes from

In one recent M2E project, the customer asked us to duplicate the existing business process currently carried out entirely in an ERP system as a proof-of-concept for M2E. The project was focused on the technical feasibility of M2E but specified with no reuse in mind, eliminating a key factor in how SOA and M2E reduce development cost. The project also did not allow for any changes in the business process, eliminating the possibility that revenues would increase or operating costs decrease as a result of our work.

Manes (2008) in fact cautions that there will be "big challenges measuring ROI [Return On Investment]" on a SOA initiative. As a result of the experience with this customer, we now know how important it is that organizations pursuing an M2E or SOA solution understand how they should and should not expect to see value. Value comes from increased revenues or decreased costs.

- If no improvements are made to the business process, there is no reason to expect that revenues will increase, regardless of whether you implement a composite application or continue to use your existing systems.
- Costs decrease sharply if you retire a system. However, a system cannot be retired if a composite application is going to leverage its functionality, so "replacing" a system with a composite application is often a misnomer and not a way to decrease costs. (However, if the composite application is designed to leverage the same functionality from a different system instead, then perhaps it is an avenue to facilitate the retirement of a particular system.)
- M2E and SOA solutions cost less to develop because some amount of functionality is reused rather than rebuilt from scratch. The less your solution reuses, the more you should expect its development to cost.
- A solution or component that can be built in a service-oriented way can also be built in a non-service-oriented way. If there is no reuse involved (as is often the case with an organization's first SOA implementation), then it would be incorrect to assume that the service-oriented implementation will show reduced cost over the non-service-oriented solution. It may even cost a little more, considering that any component services being built for the SOA solution should probably be built with both present and future uses in mind (whereas non-SOA solutions need not take other uses into account).
- It is easy to assert that there is some value in the future flexibility and reusability offered by an M2E or SOA solution. Quantifying that value is a more difficult exercise, but one that you will probably have to undertake if pursuing funding for a SOA or M2E solution.
- There is some value—perhaps even synergy—when one thing is "made for" the other. Romantic interests and custom-made suits are two prominent examples. An M2E or SOA solution gives up this value in most places where components are reused. The "made for" value can be retained in things that are not reused—for example, a customized user interface designed to facilitate a particular business task—but in

general there is a trade-off of quality for cost because of the generic-building-block approach to SOA solution design.

6.2.3 Business people and technical people working together

When it comes to creating a new business application, many organizations have a divide—formal or informal—between the business staff who will use the application and the technical staff who will build the application. Often, the business staff will create requirements documents with no input from the technical staff, then hand off to the technical staff who will build the application without any further interaction with the business staff. This situation is often referred to, disparagingly, as "throwing it over the wall." In other situations, the technical staff has responsibility for gathering requirements from the business staff, with the result being that a lot of business input is missed.

Table 1, recreated from Ellis (2008) shows the results of requirements ownership by either the technical organization (row 1) or the business organization (row 2). Note that both cases result in budget and time overruns—less so for an IT-led requirements process, but in part because the IT-led initiative underdelivered on the desired functionality whereas the business-led initiative delivered far more than was needed (not necessarily a good thing). However, note that a jointly-owned requirements process results in less overrun and more accurate delivery of the desired functionality.

Who Owned Primary Responsibility for Requirements?	Budget % of Target	Time % of Target	Functionality % of Target	Stakeholder time % of Target
IT Organization	162.9	172.0	91.4	172.9
Non-IT Business	196.5	245.3	110.1	201.3
Jointly Owned	143.4	159.3	103.7	163.4

N=109

Table 1. Diagnosing Requirements Failure (Ellis, 2008)

This finding agrees nicely with our experience on M2E projects that we get better results when our business and technical staff work side-by-side to define and implement the solution. This arrangement helps to ensure that requirements are technically feasible and that the nuances of the business are accurately implemented. While the business staff should drive the requirements gathering, involving technical staff allows for better level-of-effort estimates and occasionally ideas about how new technologies can aid the business. However, good requirements do not automatically result in successful solutions. It is ultimately not the requirements document that gets executed in production but rather the code produced during the implementation. Having the M2E skeleton is helpful for keeping

the code close to the business requirements, but perhaps more useful is a business person sitting next to the programmer, able to provide clarification and point out where the implementation can be improved.

M2E is not inherently a situation in which requirements responsibility is jointly owned. In fact, the intent of M2E is specifically to make "throwing it over the wall" more accurate. One can see evidence of this in the fact that the menu option in Oracle BPA Suite for skeleton creation is labeled "Share Blueprint with IT," suggesting that IT was not involved prior to skeleton creation. Nevertheless, despite the improvements made by M2E to the "throwing it over the wall" process, we strongly advocate joint requirements gathering and joint development.

7. Conclusion

In conclusion, Model-to-Execution is a viable means of integrating a heterogeneous system landscape. The solution described in this case study is one example, and we expect that in the future other organizations will follow our lead and use a Model-to-Execution approach to develop their own supply chain composite applications. The approach that we present addresses the problem as it actually occurs in industry. That is, our logistics business process is automated using multiple system components, which is the most realistic scenario.

To test our primary hypothesis, we developed an actual composite solution, proving that such an approach is possible. This type of hypothesis test is definitive.

We have explained at a useful level of detail our solution and how we used Model-to-Execution to develop it. We also discussed some of the benefits of Model-to-Execution and some of our lessons learned over various M2E projects.

8. Acknowledgment

The authors wish to acknowledge our customers, business partners, colleagues, and former colleagues that have shaped the way we think about this Model-to-Execution paradigm. We learn something new on every project, and our approach is more sophisticated for it.

9. References

Davenport, T. (October 1992). *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, ISBN 978-087-5843-66-7, Boston, USA.

Ellis, K. (2008). *Diagnosing Requirements Failure*, IAG, New Castle, Delaware, USA.

Gulledge, T. (2010). Integrated Business Process and Service Management, In: *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, 1st Edition*, J. vom Brocke and M. Rosemann, (Eds.), 481-496, Springer-Verlag, ISBN 978-364-2004-15-5, Berlin Heidelberg.

Jost, W. (June 2010). *Software AG Product Roadmap & Vision*, Keynote presented at ProcessWorld 2010, Washington, D.C., USA.

Kemsley, S. (2010). *Business & IT: Sharing the Vision of Process Excellence*, 2011.04.01, Available from http://www.softwareag.com/corporate/images/sec_SAG_ARIS-wM_BusIT_WP_Nov10-web_tcm16-80009.pdf

Manes, A. (2008). *SOA Report Card*, Presentation at the Burton Group Catalyst Conference 2008, San Diego, California, USA.

Oracle Corporation. (2008). *Oracle Business Process Analysis Suite*, 2011.04.01, Available from http://www.oracle.com/technetwork/middleware/bpa/overview/oracle-bpasuite-11-datasheet-1-130499.pdf

Supply Chain Management - New Perspectives

Edited by Prof. Sanda Renko

ISBN 978-953-307-633-1
Hard cover, 770 pages
Publisher InTech
Published online 29, August, 2011
Published in print edition August, 2011

Over the past few decades the rapid spread of information and knowledge, the increasing expectations of customers and stakeholders, intensified competition, and searching for superior performance and low costs at the same time have made supply chain a critical management area. Since supply chain is the network of organizations that are involved in moving materials, documents and information through on their journey from initial suppliers to final customers, it encompasses a number of key flows: physical flow of materials, flows of information, and tangible and intangible resources which enable supply chain members to operate effectively. This book gives an up-to-date view of supply chain, emphasizing current trends and developments in the area of supply chain management.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821