Combining SOA and BPM Technologies for Cross-System Process Automation

S. Herr[1], K. Läufer[2], J. Shafaee[2], G. K. Thiruvathukal[2], G. Wirtz[1]

[1]Distributed and Mobile Systems Group, University of Bamberg

Feldkirchenstraße 21, 96052 Bamberg, Germany, sebastian.herr@gmail.com, guido.wirtz@uni-bamberg.de

[2]Emerging Technologies Laboratory, Dept. of Computer Science, Loyola University Chicago

820 N. Michigan Avenue, Chicago, Illinois 60611, U.S., {laufer|shafaee|gkt}@cs.luc.edu

Abstract

This paper summarizes the results of an industry case study that introduced a cross-system business process automation solution based on a combination of SOA and BPM standard technologies (i.e., BPMN, BPEL, WSDL). Besides discussing major weaknesses of the existing, custom-built, solution and comparing them against experiences with the developed prototype, the paper presents a course of action for transforming the current solution into the proposed solution. This includes a general approach, consisting of four distinct steps, as well as specific action items that are to be performed for every step. The discussion also covers language and tool support and challenges arising from the transformation.

Keywords: SOA, BPM, BPMN, BPEL, WSDL, standards, application integration, BPIOAI, web services

1. Introduction

As part of their efforts to automate enterprise-wide business processes, organizations are often faced with the challenge of integrating the data and business logic of several independent application silos [6]. This issue is traditionally addressed through custom-made solutions that are expensive to build and maintain, inflexible to changing requirements, error-prone and often poorly aligned with the enterprise's business goals. One promising and increasingly recognized approach to this problem is to combine Business Process Management (BPM) and Service Oriented Architecture (SOA) concepts and technologies with the goal of forging a flexible and cost-efficient process automation and system integration solution. In combination, both paradigms appear to be of significant benefit to each other. BPM's lack of focus on architectural principles (e.g., loose coupling, service reusability) and its poor flexibility in regard to technology choice (i.e., vendor lock) are addressed by SOA. SOA on the other hand can benefit from BPM's top-down, requirements-oriented, and visualization-focused approach that considers the entire life cycle of business processes.

Despite the general recognition of a SOA and BPM convergence and the resulting synergy [3, 7, 9], there is still an extensive lack of best-practice examples and real-life industry adoption, in particular in medium-sized and small enterprises. One reason for this is a shortage of resources paired with the common believe that SOA and BPM initiatives only bring return on investment (ROI) if applied on a large scale. While this assumption may hold some cases, we argue that using the suggested paradigms results, even in small-scale scenarios, in considerable benefits that have the potential of significantly outweighing the anticipated costs.

To back this proposition, we conducted an industry case study with the goal of applying a combination of SOA and BPM concepts and technologies as a replacement for a custom-made, proprietary process automation solution. Our focus was the universal use of standards, specifically the Business Process Modeling Notation (BPMN) and the Business Process Execution Language (BPEL). In particular, we identified weaknesses of the currently deployed solution and specified a transformation of this solution to the envisioned replacement, including a roadmap with specific required action items along with the selection of pertinent tools. Finally, to address financial considerations as well as several other concerns, we thoroughly evaluated and compared both solutions, with special emphasis on addressing the previously identified weaknesses. The study was conducted in the setting of a top-five global hosting company with the intent of delivering a proof of concept for the feasibility and impact of the proposed approach. The transformation itself was realized in the form of a prototype (PT) that included process-models (BPMN) and -code (BPEL), service interfaces (WSDL) and "dummy" implementations of services, but no end-to-end implementation with the actual enterprise applications.

Our work was influenced by recent publications addressing the SOA/BPM convergence (e.g., [3, 7, 9]) as well as general literature on approaches to application integration [5] and SOA [1, 4, 6]. During the PT development, we have relied extensively on the BPMN-to-BPEL mapping rules [8].

The remainder of this paper is organized as follows. In section 2, after briefly describing the existing solution and its weaknesses, we introduce the chosen pilot process. In section 3, we present the approach and results of the transformation, while section 4 compares both solutions and discusses how the issues from section 2 are addressed. We conclude with a summary and a discussion of future work.

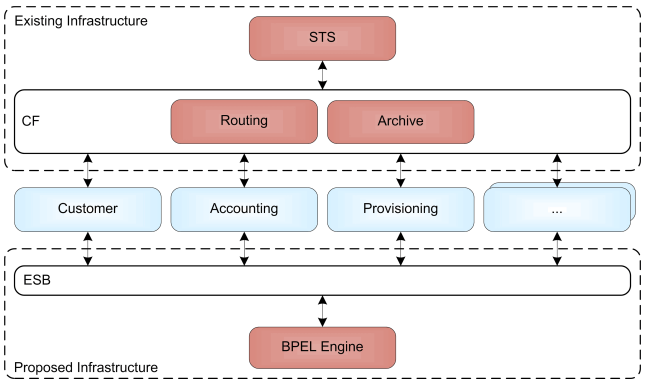

Figure 1. Enterprise Architecture

2. Initial Situation

The environment's system landscape comprises ten physically separated back-office applications that each take over a distinct business or auxiliary function (e.g., Accounting System, cf. Fig. 1). Together, these systems fully automate most of the company's business processes including sales, billing, and provisioning, to name a few.

2.1. Existing Process Automation Solution

The company's process automation approach integrates a custom-made state transition system (STS) for process state management and business rule enforcement with a custom-made communication framework (CF) for system integration (cf. Fig. 1 top). Information exchange between back-office systems is realized via asynchronous XML-based messaging. Typically, a new process is brought to life inside the STS, which is triggered by an external event such as the submission of a new sales order. The STS manages the process execution by sequentially starting various integration flows via the CF. Integration flows commonly visit several back-office systems (sequentially and without the involvement of centralized coordinator), cause business logic updates and/or collect data, which is attached to the message and used later inside a different system as part of the overall process. Each integration flow eventually returns to the STS causing a process status update, which results in further actions.

2.2. Solution Benchmark

We have identified four high-level issues: initial development and setup costs, maintenance costs (i.e., integration of new systems, implementation of new business requirements, etc.), technology and solution learning curve and the likelihood of introducing errors during the mapping of business requirements to process implementations. Upon examining these issues, we have detected four problems as their respective root causes. In turn, these problems give rise to the criteria for the evaluation and comparison with the suggested solution.

1. *Lack of Sufficient and Practical Documentation*

Acquiring sufficient knowledge to understand and work with the existing solution requires collecting and aggregating information from various sources (e.g., people, documents, code) and was experienced as a highly time-consuming part of the case study. Also, the poor documentation increases the likelihood of introducing errors. Besides a lack of language and technology tutorials, which can be ascribed to the proprietary nature of the solution, the environment also misses a common communication basis in the form of a process model that can be shared among business analysts, architects and programmers. Without such a model, new requirements are likely to be misinterpreted when handed to a programmer. Finally, even if sufficient documentation in the form of state-transition (ST) tables, ST diagrams and flow diagrams were available, it would not provide a complete and practical picture.

2. *Complexity of Solution and Process Setup*

First, the initial setup required building most components from scratch. This includes the STS with database setup, database triggers, stored procedures, the event module as well as the CF with its routing and archive modules, message parser but also the development of the message structure (i.e., routing information) and other rules for new process setup. Merely the message queues (MQ) (one per system) could be acquired from a third-party vendor. Second, new processes or even slight adjustments in existing processes require careful and complicated planning and design. States, state transitions and events need to be designed and set up. Routing information has to be added or changed, integration flows and additional message parsers have to be implemented. This high effort of process maintenance is even further intensified by the solution's poor separation of concern (SoC). Although process implementations are distributed across two components (i.e., STS and CF), the solution places some of the process-specific behavior inside the back-office systems (i.e., in some instances, the route of an integration flow is changed during system invocation). This again requires most systems to be aware that they are part of a higher-level process and hence prevents them from being reused by other processes.

3. *Poor Deployment and Testing Conditions*

Process deployment requires manual addition or replacement of routing information, states, events, etc. "One-click" deployment is not supported. The effort for process testing has been indicated by responsible personnel as being extremely high. First, the various steps of a process (i.e., the systems invocations) cannot be tested individually but only as part of an integration flow. Second, process debugging is only possible in retrospect by evaluating the log files of completed flows. Third, the solution does not allow for design-time code validation through respective tools.

4. *Degree of Business-IT Alignment*

The solution is clearly deficient with respect to the alignment of the enterprise's processes with its IT infrastructure. The nature of the solution generally makes it difficult to translate business requirements into implementations. Every process, coming as a whole from the business side, needs to be split up into several flows that are tied together via the STS. It

is often difficult to decide which parts of the process should be realized through flows and which parts require state and corresponding state transitions. Also, the question arises whether process-specific rules should be enforced inside the STS or inside the pertinent back-office system. In addition, the implementer must sometimes deviate from the order of steps provided by the business analyst.

2.3. Selected Business Scenario

As the basis for our PT, we chose the company's sales process as the pilot to be implemented with the suggested approach. We picked the sales process because of its universality (i.e., industry neutrality), its size and complexity (involving all of the organization's back-office systems), and its volatile nature that would emphasize the benefits of the proposed solution. At a glance, after an order has been placed via an online shopping cart environment, the back-end process is started performing various steps including the creation of a customer account, a fraud check, the processing of financial transactions and the initiation of product provisioning. Each step is implemented by a different back-office system.

3. Solution Transformation

After analyzing the existing process automation solution and taking a closer look at the selected pilot process, it became clear that a simple transformation to the suggested technologies and paradigms would be insufficient to best demonstrate their benefits. First, a mere transformation without addressing the poor SoC of the current setup including its weak service reusability potential did not appear to be meaningful or at least would not allow us to demonstrate the potential for reusability in the long run. Second, during the reverse-engineering effort of the sales process from source code and residual process documentation, we were able to discover several crucial errors in process logic that could easily be repaired with the new approach. Finally, the current solution does not implement process-wide transactional behavior; this issue could also be addressed effortlessly and hence was added to the list of objectives.

To reduce the complexity of the transformation process, we decided to address the various goals and objectives in four sequential steps. This so-called "transformation lifecycle" (Fig. 2) can be reused for porting additional business processes in the future. Phase one sought to integrate the existing systems using BPEL. This implies exposing the respective functionality via web services (WS). Additionally, the issues of SoC and reusability have been addressed by shifting some functionality between systems and by applying well-deliberate service design. Phase two aimed to bridge the gap to the business side by providing a visual, more business-oriented representation of the sales process using BPMN. Based on the newly created models from phase two, phase three directly repaired process errors and added business transactions from a requirements-oriented standpoint. Finally, phase four synchronized the adapted process models with the previously created implementation. Each of the four

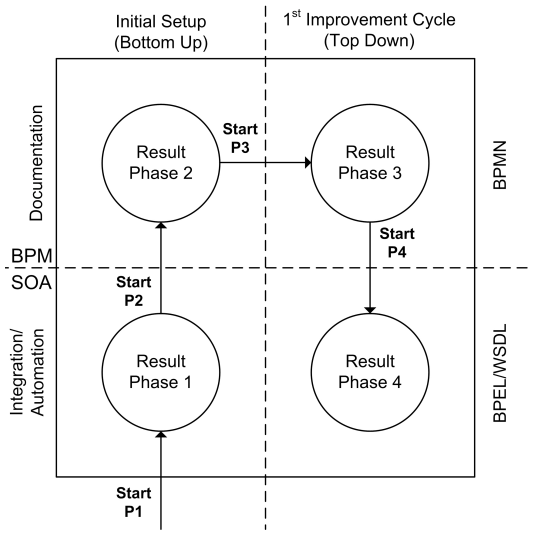

Figure 2. Transformation Life Cycle

steps will be discussed in detail below.

Phase 1: Bottom-Up Process Automation

In phase one BPEL v1.1 and WSDL v1.1 where used with Oracle's JDeveloper tool that supports various SOA-related development tasks including service-stub generation from WSDL-files and visual BPEL modeling.

First, we exposed the existing systems as WS. As a result, 100% of the systems' functionality (i.e., business logic) was reused and system access has merely been transformed from the CF to WSDL-based access. One important task was the definition of service inputs and outputs: business documents, such as an invoice, that are specified through XML schemas. The challenge was to transform the structure of the existing XML message into various modular schemas that can each be passed into the respective service, individually or in aggregated form. The transformation was required for adopting BPEL's orchestration approach. A BPEL process locally stores process data, such as a sales order (consisting of a quote, customer data, product selection, etc.), and extracts data (e.g., one specific product item) in the course of a service invocation as needed. A second challenge was to improve SoC and potential for reusability. To this end, we moved some functionality between systems as well as from the systems into the process. The newly created services have been designed accordingly (i.e., assuming the functionality in the new place).

Second, we implemented the sales process in BPEL by orchestrating the previously designed services. This required a transformation of the state- and flow-based solution into an activity-based solution. Process state management and routing of messages is now realized transparently by the respective infrastructure, and the various integration flows were combined into one process. Our experience was that all aspects of the sales process could be expressed in BPEL with no restrictions.

As our deployment and testing environment, we installed the Oracle SOA Suite. It ships with an Enterprise Service

Bus (ESB) and a BPEL engine and integrates nicely with the chosen IDE (JDeveloper), allowing BPEL processes and services to be deployed from within the IDE. The installation of the BPEL engine and ESB, or optionally a lightweight WS framework, makes the STS and CF obsolete (cf. Fig. 1). With some careful planning, it should also be possible to run both solutions concurrently until all processes have been ported.

Phase 2: Bottom-Up Process Visualization

The potential of the proposed solution can be fully utilized only with the addition of a process model that may be used as a communication basis among participating stakeholders. In phase two, we performed a mapping from the created BPEL code to a visual BPMN-based process model using BPMN v1.0. The model was created with the Microsoft Visio stencil extension from ITP Commerce. The resulting business process diagram (BPD) is illustrated in Fig. 3 (exclusive the highlighted parts). The mapping was done based on the rules by [8] and turned out to be very straightforward. All aspects of the BPEL code could be transformed to the BPD.

Phase 3: Top-Down Process Adaptation

The objective of phase three was to repair flaws, to add additional functionality and to make the process transaction-safe. Tool and language support is identical to phase two. As a first task, we swapped some steps in the process, which were out of order in the existing implementation. Secondly, we added one extra step at the end (i.e., "Send Order Completed Notification"), which was missing in the original solution. Thirdly, we extended the BPD with a manual order verification option and finally added rollback processes that would reverse prior system updates in case of process failure. The resulting artifact is an extended version of the BPD from phase two (i.e., Fig. 3, inclusive the high-lighted parts) as well as additional BPDs for rollback- and related processes (not shown here). It is important to recognize that phase three is entirely requirements-driven and independent of the underlying process and system implementations. Nevertheless, the new functionality was modeled based on design decisions that determined which system had to implement the new requirements in the future.

Phase 4: Top-Down Process Implementation

The final step in our life cycle was the synchronization of the BPDs from phase three with the existing implementation from phase one. Tool and language support are the same as in phase one. Most parts of the BPDs were implemented with no effort by using the visual process modeler of the IDE.

The implementation of the manual verification step and the transactional behavior is of interest. The latter was necessary to map the rollback processes from the BPD to the BPEL implementation. BPEL realizes loosely-coupled business transactions with its built-in compensation and fault handlers. A very convenient BPEL feature is the automatic execution of compensation handlers of those scopes that are already completed when the error occurs. An example for our case is the deletion of the sales order in the Sales System and the reversal of the "Create Customer Account"-sub process (e.g., after an order expired while waiting for manual payment, cf. Fig. 3). Both steps required an extension of the previously created WSDL interfaces with additional operations that simply reverse the existing operations.

The manual verification of orders naturally requires human involvement. Unfortunately, BPEL does not support human interaction in a standardized way. On the other side, we purposely refrained from using Oracle's proprietary BPEL extension for human workflows to avoid vendor lock. We rather addressed this issue by simply hiding the human involvement behind a custom-made service. In this case, the process asynchronously submits the sales order to an "Order Escalation Service," which prompts the need for order-verification to a user interface. The process waits in the current status until it is called back by the service after the issues has been resolved.

4. Evaluation and Solution Benchmark

This section benchmarks the existing solution against the proposed solution. We will discuss the issues identified in section 2.2 and outline how they are addressed in the new solution. We also provide some specific figures (i.e., cost savings) based on our experiences with the PT development. Nevertheless, the results presented here are conjectures that have not been measured owing to time restrictions. Determining the actual ROI would require operation and observation of the new solution in production environment over a period of several months.

In summary, the proposed approach combines state-transitions and integration flows into one artifact (i.e., BPEL process) whereas the BPEL engine orchestrates the various services into one enterprise-wide process. Changes in routes are now handled inside the process. Messages are transmitted per service invocation and not per flow. Service reusability is improved and process-wide transactional behavior is provided. The solution comprises an end-to-end process model clearly displaying all business requirements. Finally, the STS and CF are replaced by SOA and BPM infrastructure components (cf. Fig. 1).

1. *Documentation*

With the BPD (cf. Fig. 3) most information about the sales process and participating services is available at a glance. Aggregation of information from several sources (including source code review) is not necessary. The process clearly visualizes all steps, service invocations, possible faults, events, branches and business rules (e.g., expected values for decision points). The translation from the model to the implementation is unambiguous and the error likelihood is reduced. Since BPMN is specifically designed to be stakeholder neutral [8], it allows all participants to understand the model quickly. By contrast, ST- and flow diagrams may be harder to understand by purely business-focused staff. Furthermore, new stakeholders can quickly acquire

information about languages (i.e., BPEL, BPMN, WSDL) and infrastructure (i.e., BPEL engine, ESB) used in the project by studying some of the myriads of available tutorials and code examples. It will also be easier to recruit new stakeholders (e.g., engineers) that already carry the respective knowledge; this is not possible with a proprietary solution.

2. *Complexity of Solution and Process Setup*

The proposed solution has the potential of significantly reducing time-to market of the initial setup and of new requirements. Infrastructure is entirely replaced by third-party offers, which greatly reduces the setup time of the initial solution. Design and implementation of the generic parts of the STS and CF was indicated by the responsible manager with approximately US$70,000 (including purchase of several MQ licenses), which is already above the purchase price of a commercial SOA Suite license (US$50-65,000 assuming that one license will be sufficient). Nevertheless, despite the possibility of relying on open source infrastructure, we argue that even higher investments will pay off as the maintenance costs of enterprise applications commonly are much more significant [2] and this is where the suggested solution scores big points. Setup of new processes and adjustments in existing processes is far less time consuming. This is besides the improved testing conditions especially also ascribed to BPEL's predefined language constructs for process behavior as well as the tool support with its visual BPEL modeler, code generation etc. Furthermore, services can be designed for reuse, which again may reduce development time in the future. In particular, we anticipated savings of approximately US$10,000 in personnel cost only for the initial setup of the sales process. Simple adjustments in process logic can save up to 80% of implementation and testing time. With more complicated adjustments (e.g., the addition of a new payment option), this is even more significant.

3. *Deployment and Testing Conditions*

The solution's testing and deployment conditions have improved significantly. The infrastructure ships with built-in features that facilitate testing and debugging. Processes can be deployed effortlessly via the respective user interface or directly from within the IDE. Versioning is supported transparently (i.e., changed processes are deployed under a new version, running processes are completed in their old version). Furthermore, pre-deployment code verification is provided by the IDE and greatly reduces runtime errors. Finally, all services can be tested individually before attaching them to a process. The improved testing and deployment situation has a significant impact on the cost savings already discussed.

4. *Degree of Business-IT Alignment*

BPEL's orchestration approach in conjunction with the BPDs improves the solution's general degree of business-IT alignment. The process is visualized and implemented in a more natural way. Process behavior (e.g., branching, business rules, etc.) is integrated into one artifact and not distributed across several components. The number of steps in the process that cannot directly be mapped to a business requirement is reduced, thus, the gap between requirements and implementation is smaller.

5. Conclusion and Future Work

In this paper, we argued that a cross-system process automation solution leveraging SOA and BPM technologies can be significantly superior to a custom-built solution. We also presented an approach for porting the existing processes to the suggested solution. The study showed how the chosen concepts and technologies can be applied to a real-life scenario. All aspects of the pilot project (i.e. process model and implementation, services, etc.) were realized with the selected languages, and the PT was successfully tested. In conclusion, the study was a considerable success and demonstrated the merits of using a combination of SOA and BPM for process automation and system integration purposes. The resulting value gain was greatly recognized by the responsible personnel in the chosen environment.

Finally, to guarantee feasibility and long-term success, we recommend performing a more thorough analysis and evaluation of infrastructure and tool support for the specifics of the environment, including, for instance, issues such as tool selection. Second, an in-depth evaluation of quality of service aspects (e.g., scalability, security, reliability) should be performed. Furthermore, it may be advisable to extend the PT with an end-to-end implementation that involves the pertinent systems. As a last step, we believe that drafting an adoption strategy (i.e., guidelines, education of staff, adoption schedule) will help to discover other possible issues.

In general, the transformation life-cycle introduced in section 3 requires more case studies in order to adjust and generalize the porting process outlined in Fig. 2 in a way that it becomes useful also under different settings.

References

[1] N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah. *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap*. IBM Press, 2005.

[2] L. Erlikh. Leveraging legacy system dollars for e-business. *IT Pro*, 2(3):17–23, May/June 2000.

[3] F. Kamoun. A roadmap towards the convergence of business process management and service oriented architecture. *Ubiquity Volume 8 , Issue 14 April 2007.*

[4] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA Service-Oriented Architecture Best Practice*. Prentice Hall Ptr, 2004.

[5] D. S. Linthicum. *Next Generation Application Integration: From Simple Information to Web Services*. Addison-Wesley Professional, 2003.

[6] B. Lublinsky. Defining SOA as an architectural style. IBM Website, January 2007.

[7] J. Noel. BPM and SOA: Better together. IBM Website, White Paper, 2005.

[8] OMG. Business Process Modeling Notation specification. OMG Website, July 2007.

[9] M. Rosen. BPM and SOA: Where does one end and the other begin? BPTrends, January 2006.

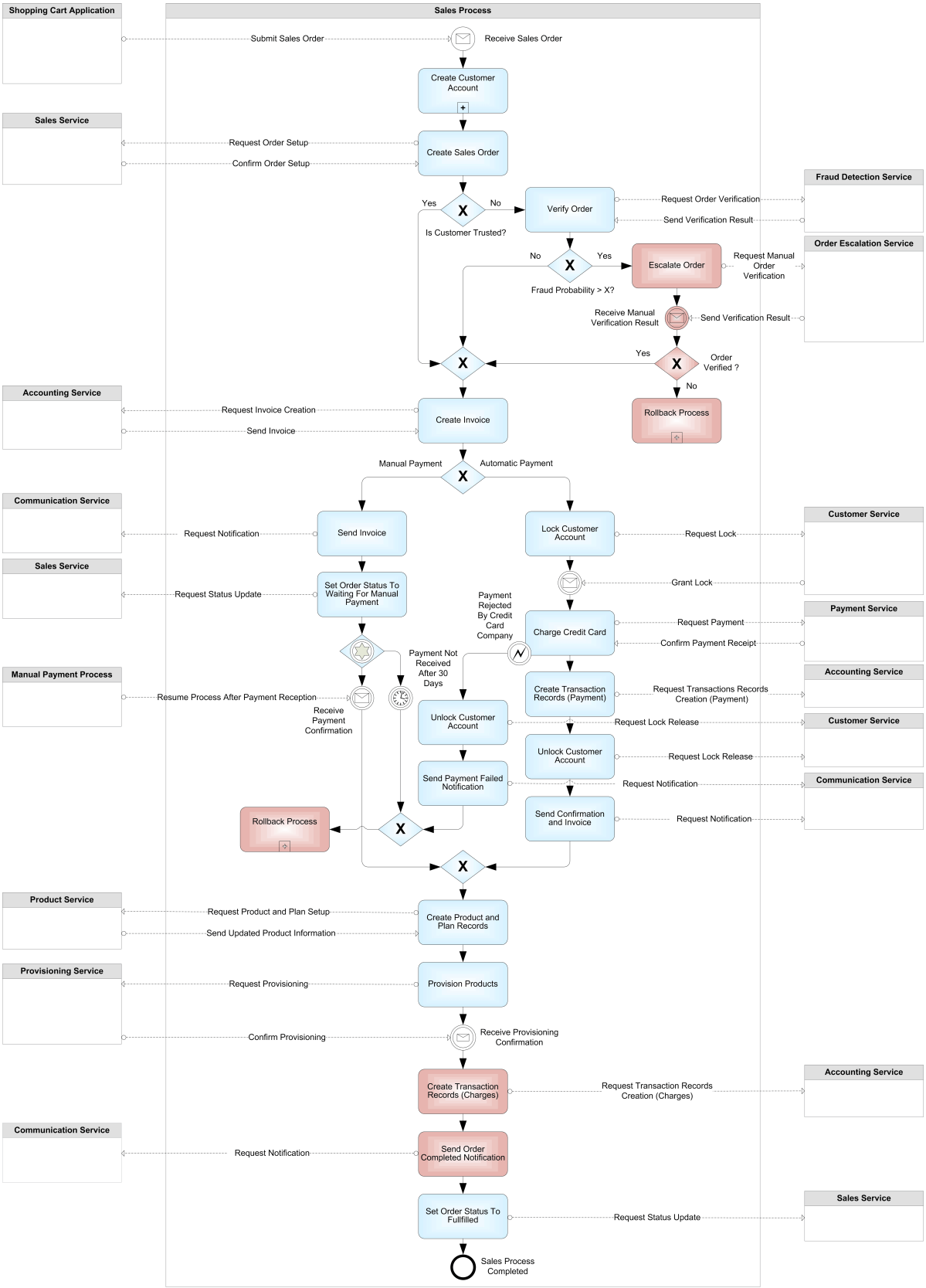

Figure 3. Sales Process BPD