BPM on Top of SOA: Experiences from the Financial Industry

Steen Brahe

Danske Bank and IT University of Copenhagen, Denmark
stbr@danskebank.dk

Abstract. Service Oriented Architecture (SOA) forms an ideal infrastructure for Business Process Management as applications are invoked using standard interfaces and protocols. Automatic services can be composed together with human tasks into complex business processes that cross departmental borders and integrate customer and partner processes. Despite the current hype around SOA and BPM, reports on industrial experiences are still very limited. This paper presents results from empirical studies on adopting BPM and SOA throughout the last 4 years in the IT organization of Danske Bank, one of the largest financial institutions in northern Europe and a pioneer in adopting SOA. The study shows the benefit from automating a traditional business process using BPM and SOA, but it also reveals several challenges, technical and organizational, of converting traditional development into service- and process-oriented development.

1 Introduction

Service Oriented Architecture (SOA) [1] and Business Process Management (BPM) [2] are claimed to be two important topics for making an enterprise responsive to a changing market. By service-enabling its existing legacy systems and using service-oriented development techniques for new application development, the enterprise should be able to create loosely coupled and reusable services that can be composed and orchestrated into complex business processes which integrate human tasks and systems across departmental silos. The concepts of SOA and BPM are getting much attention from both academia and industry, and there seems to be an agreement on the importance and the benefits for an enterprise to adopt these.

Although BPM has its roots in Workflow Management (WFM) [3], a topic of research since the 70'es [4], BPM based on SOA and SOA itself have been around for only a few years. As with WFM, BPM has not yet got an industrial break through and there are only limited documented experiences about adopting BPM and SOA. Experience reports mostly describe the benefits seen from the business perspective. Not much is said about challenges in adopting the concepts, methods and technologies as seen from the IT development and the organizational perspective.

G. Alonso, P. Dadam, and M. Rosemann (Eds.): BPM 2007, LNCS 4714, pp. 96–111, 2007.
© Springer-Verlag Berlin Heidelberg 2007

This paper presents empirical research carried out in the IT organization of Danske Bank, one of the pioneering companies in adopting BPM and SOA.

The author, who previous was a part of the development team responsible for the BPM infrastructure, has through interviews, workshops and document studies, examined two large and independent projects both implementing cross departmental processes. The *customer package* project implements the business process for creating financial products such as credit cards, bank accounts and internet bank access. This process integrates services from more that 10 different systems. The *Account settlement* project implements the business process to finish a customer's engagement in the group, e.g. closing accounts. This process integrates services from around 15 different systems. Both projects consist of around 30 separate processes and integration to 50 different service operations in total, and include several human tasks. There are large similarities between the two projects; they were developed shortly after the adoption of SOA and were the first business processes to be automated using workflow techniques. All people involved had no previous experience with BPM and SOA.

Throughout the paper, the customer package project is used as the main case to describe the experiences. First, the history of the customer package process is described from the business perspective. This shows the business value of using BPM and SOA to support an existing business process. Second, the same story is described as seen by the IT organization. This reveals another side of the story as several challenges were faced by the development team.

The rest of the paper is organized as follows. Section 2 presents some background information and introduces Danske Bank. In section 3 the history of the customer package process is described from the business perspective including the benefits gained from automating the traditional work practice. Challenges experienced by the development team responsible for implementing the process are described in section 4. Section 5 describes lessons learned based on the experiences. Section 6 describes related work, and section 7 gives a summary.

2 Background Information

This section introduces Service Oriented Architecture and Danske Bank, the enterprise in which this study has been carried out. The following concepts related to the term "process" are used throughout the paper:

- Process / Business process. A coordinated set of tasks for handling a business event. For example the work practice of handling loan applications.
- As-is / to-be process. High level conceptual (business) models of the business process.
- Solution model. A detailed model of the business process. Is a logical specification of how to implement the business process.
- Workflow: A program that is able to coordinate and control the different tasks that make up the business process. It is an implementation of the solution model.

- Process instance: An instance of a workflow, e.g. a loan application for customer A.
- Development process: The software development practice followed by a development team to define a solution model and implement it as a workflow.

2.1 Service Oriented Architecture

Service Oriented Architecture is an enterprise architecture that advocates loosely coupled and reusable systems. It has evolved from component-based development and distributed internet architectures as a new abstraction layer that allows internal and external systems to interact using common standards and protocols. SOA makes it possible for an enterprise to open up its legacy systems to other systems and services. As SOA provides an enterprise *architecture* for building systems, service orientation represents a new paradigm of software *development* that seeks to bridge the gap between business and IT. Business analysts and architects define requirements and solutions in terms of services. When developers implement the solution, these services are either located in the local service repository or developed from scratch. The business and technical people have got a common language for their work.

2.2 Danske Bank

The financial group, Danske Bank, dates back to 1871 where it was founded in Copenhagen as "The Danish Farmer Bank". Since then, it has grown to become the largest financial group in Denmark - and one of the largest in northern Europe. It comprises a variety of financial services such as banking, mortgage credit, insurance, pension, capital management, leasing and real estate agency.

Danske Bank has grown through acquisitions, mainly due to its successful IT strategy - one group, one system. This strategy focuses on using the same systems throughout all products, distribution channels, brands and markets. When acquiring a new company, its current products, processes and data are converted to the Danske Bank platform, while existing systems are dismissed.

To support and fulfill its IT strategy, Danske Bank has adopted a Service Oriented Architecture at which all new application development is targeted and where existing legacy systems are service enabled. Applications and services developed for one part of the group, can through a central service library and repository be located and used by other parts of the group. As Danske Bank started out implementing SOA before the web services standard was defined, it has developed its own proprietary standard for service specifications. Currently, the enterprise has several thousand different service operations.

Support for automating business processes is achieved through a BPM system from IBM, but is extended in areas where business requirements were not fulfilled. Business processes are implemented using BPEL [5]. Fig. 1 illustrates BPM on top of SOA; how a business process implemented as a workflow through SOA is able to bind together people and applications across departmental borders.

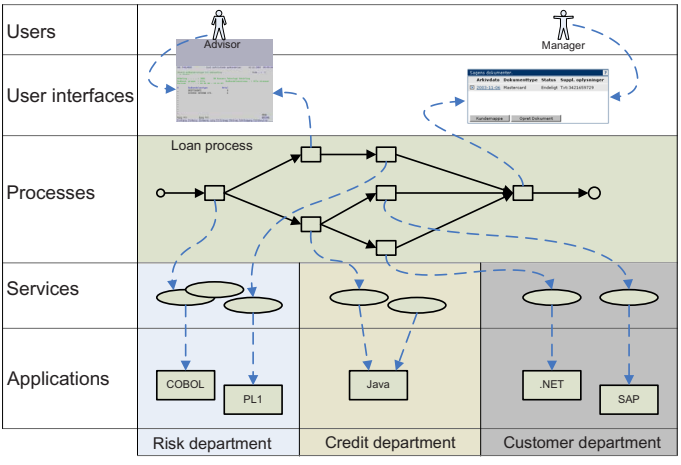

Fig. 1. Business processes as a composition of services and human tasks

For business and IT development, Danske Bank has defined its own service-oriented development process that is based on models; in fact most requirements and design decisions are captured by models. In the analysis phase, a business analyst together with the process participants define current and future work processes in high level terms as two models called the *as-is* and the *to-be* process. Further, a solution architect defines a solution model, which describes all automatic and all human tasks that make up the business process. In the specification phase, the solution model is further detailed with references to existing and new services, user interfaces, etc.

The solution model, related documents and related models, as e.g. models for service specifications, describe the solution in detail and are the specification used by a process developer to implement the business process as a workflow.

Analysts and architects define business process models as flowcharts using an enterprise specific modeling notation. Until late 2006, an outdated modeling tool called CoolBiz was used for modeling. Today, CoolBiz is replaced by Websphere Business Modeler from IBM.

3 Experiences – From the Business Perspective

In June 2003 Danske Bank introduced a new sales concept called *Customer Packages* which bundled a number of financial products, e.g. a credit card and an account. When a customer visited a branch he or she could sign up for a customer package containing e.g. an account, a credit card and an internet bank account. A Word document was printed, filled in and signed. When the customer had left the branch, the customer adviser would send the document by mail to a back office department, where a group of people were responsible for reading through the received documents and creating the different products in different legacy systems.

The customer package creation process handled by back office was highly predictable and production-like and involved systems from several departments throughout the group. A workflow for this cross departmental process would be able to link the different systems together and eliminate the need for entering the same information repeatedly in different systems. It was decided to use the customer package process as a pilot for the new BPM system that had recently become ready for use. The workflow implementing the process was put into production in December 2003.

The first version of the workflow was basically implemented precisely as the back office workers used to create products. All products except one were still created manually in the same systems as before. But now, the workflow automated the distribution of the different tasks and in which sequence they should be carried out. Based on the document filled in by the customer adviser, the workflow created a list of tasks that back office workers should handle. A back office worker would log onto a task system, where the list of tasks - or products to be created - was listed. When accepting a task, the worker was automatically transferred to the system for creating the given product, and available data were present, delivered by the workflow. The workflow hereby became the glue that bound the different systems together, see Fig. 2.

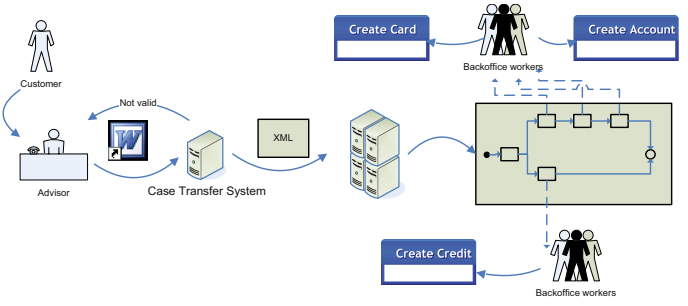

Fig. 2. Workflow enabled back office process with workers creating the products

The workflow-enabled process was an improvement of the previous product creation process because the system made the relevant information available to the back office workers, and guided them directly to the relevant systems from the task list.

Now, having implemented the business process as a workflow, the business department began to look for optimization possibilities. The obvious optimization of a workflow-enabled process is to automate the manual tasks in the process, i.e. to perform the worker's tasks through automatic services. The business department started to contact the departments responsible for the product systems and requested automatic product creation services. Unfortunately, most of the departments did not have the resources for developing the required services. However, one important product that always had to be created for a customer

package was an account. The Account department agreed to develop an automatic account service. It was incorporated into a new version of the customer package workflow. Now, the back office workers did not have to create accounts anymore; it was handled automatically by the workflow and the new account service. This was an eye opener for the business department; "Are we able to automate the creation of accounts, then we will also through systematic work be able to automate much more of the product creations".

For about two years, the process has systematically been improved and optimized by looking for the most expensive and time consuming tasks. In the first version of the process, all tasks were handled manually. Today, the process is running in version 6 and 80% of all the products are created automatically. The back office workers have saved much time, which today is used for other activities.

3.1 Current Status and Future Development

The customer package process will be optimized further. It may not be possible to automate the process completely, but it should be possible to automate more than 80%. It requires system owners to develop automatic services and this has been experienced to be a bottleneck because of lack of resources, as other departments have other tasks to handle with higher priorities.

The business process consists of one main controlling workflow, a product creation workflow and workflows for each product creation. In total, it consists of about 30 workflows and 200 service invocations or human tasks.

Back in 2003 when the customer agreement department was established, about 200 customer packages were handled each day. Today, that number is about 1800, of which about 80% is handled by BPM. The other half is handled manually due to complex settings or errors in input data, cases too complex or expensive to include in the workflow.

The history of customer packages shows how a successful business idea has gradually been optimized by use of workflow technology. The use of BPM first automated the coordination of tasks. Next, it allowed the continuously optimization of the process by automating manual tasks in the process. The efficiency of customer advisers and back office workers has improved significantly, and the use of BPM has proved to be of real business value.

4 Experiences – From the IT Development Perspective

After having described experiences from the business perspective, let's look inside the IT development organization and evaluate their experiences. First, we shall look at the challenges of getting from a business model of current work practices to an implementation, and second, we shall categorize the problems into organizational and technical issues.

Problems related to test, deployment, operation of the process instances, and change management have also been observed but will not be described here. For further information about these challenges the reader is referred to [6].

4.1 From Business Model to Implementation

Two business analysts and the back office workers analyzed the current work situation and defined a model of current work practice. This model illustrated the different process steps and dependencies between them; which products should be created and in which sequence. To keep the transition from manual to automatic process control as simple as possible, it was decided to implement the existing work practice directly instead of a reengineered work process. A solution model was defined in cooperation with a solution architect. For the developer, this model described what the implementation should contain. The initial solution model contained 12 activities, mainly manual creation of different financial products. It was approved by both the users and the business analysts as a valid solution.

Solution not Complete: Only Main Road. The solution model and related information were given to the developer, who started to construct the implementation. Soon, during the initial unit test of the workflow, it turned out that the solution model only considered the "main" road of the process, the process of handling a customer package when everything was as expected. Many exceptions and special cases were not covered. After confrontation with the analyst and the users, they recognized many scenarios that they had not taken into consideration; different card types had to be created, the customer might have required a special leather bag, what should happen if the user forgot to sign the document, etc. Such exceptions are crucial to describe to ensure that the automatic process control executes in the same way as current work practice. Several times during the implementation phase, the developer had to talk to the analyst and to the users to understand the business scenario and to update the solution model. After several iterations, the solution model was complete and the workflow implemented. The first version of the workflow that was deployed contained 36 activities, three times the amount of the initial solution. The solution model had through the entire project been used by the developer as the contract to communicate to the business analysts and users what was to be implemented. The solution had clearly grown much compared to the initial design. It was a large surprise to the analyst and the users how much they had missed in the initial solution. They had not previously tried to describe work processes in such detail and were not used to get a solution that exactly matched their description. Only through good will and hard work by the developer, the business got the solution they needed.

Missing or Imprecise Information. Imprecise or missing information in the solution model was another challenge faced by the developer. Each time the developer discovered imprecise definitions or missing information about data, decisions or presentations, he had to stop developing and contact the analyst and architect to discuss what to do. Roughly, these challenges can be divided into three types:

Activities not Broken Down. As an example, an activity was described as "create all cards". When the developer should implement such an activity, he had to

consider if a new service should be developed to create a bunch of cards, if an existing service for creating one card should be called several times in a loop structure, and what should be done in case of failures when creating the cards? Such decisions are not implementation issues; it is decisions that should have been modeled in details in the solution model. The activity should have been broken down into smaller pieces in an earlier phase of the development process.

Sequence of Dependent Activities. Activities in a process may depend on each other. For instance, an account must be created before creating a card. Such dependencies were not always described explicitly and the developer had to figure out how to organize the control flow. These dependencies should have been described in the solution model.

Missing Information. Some important information was neither defined by the analyst, nor by the architect. The architect had not considered which data to use when defining service invocations or user interface based activities. Both activity types may require data that is not present and that has to be retrieved from somewhere else. The developer discovered missing data definitions when it was not possible to invoke a service, because that data was missing e.g. an account number for creating a card. When defining a human task, the architect had to decide how to present such a task to the back office workers, e.g. what text labels and data to show in the task list that is presented to the worker. Often, these data were not described either. When defining decision points in the process, the architect often described in plain text what the decision was about, but he did not describe which data to use.

Common to the above challenges is that the implementation process cannot continue until the developer gets more information from the analyst or architect. Such extra iterations causes a longer development time.

Using SOA. In addition to challenges in getting a consistent solution model, the developer also faced challenges regarding system integration because the workflow was integrating systems from different departments. All new software components must be developed and exposed as services in order to be accessible from other systems. The following three challenges were faced:

Service Location and Documentation. Thousands of service operations exist in the enterprise, but it has been experienced as difficult to locate a required service. Further, it is rarely documented well. All services can be found in a service library where also documentation, input/output descriptions as well as examples should be present. However, the service library is a new feature in the group and only a few service operations have been documented. The two examined projects use a total of about 50 different service operations from several business units, and of these only four have been documented. Because of the lack of documentation, in all cases except one the process developer had to contact the developer responsible for a given service to understand how to invoke it and how to handle response values. Some services require up to 100 input parameters without any documentation, which makes it very difficult for anyone to invoke the service.

Service Standards. Most of the services integrated into the workflow have been implemented using naming conventions and other rules defined by the service provider. Instead of naming services according to their functionality, they are often named according to their system name. A CreateCustomer service operation may e.g. be named KNI001, which makes it impossible for anyone to find and understand it. Most services are implemented on a mainframe using COBOL, and often they return codes describing different states of the service execution or possible exceptions. As no enterprise standards have been used for return codes, exception handling must be implemented differently for each service invocation.

Service Granularity and Reusability. Services to be invoked from the workflow have in several cases been too general or too specific to be useful. For instance, a credit service operation covered many different situations of creating credits for a customer, but it also required information that were not available from within the workflow. This granularity challenge was solved by requesting the responsible department for a new service operation responsible for creating the credit needed for the customer package workflow. This new (composite) service operation then collected required information and subsequently invoked the general credit service. Such a service has to be developed by the responsible department and in several cases, departments did not have resources for developing required services for the customer package workflow.

Repeated Manual Implementation Work. The idea of Model Driven Development (MDD) [7] is to transform models directly into code, but the commercial development tools in these projects do not provide flexibility to allow customization of the transformations. When implementing the solution model as a workflow, the developer therefore manually reads, interprets, and transforms the model into code. This is a repetitive and time consuming process with great risk of mistakes. First, when the developer starts implementing the solution model, he maps each task in the model to an implementation. Often, one task corresponds to a service invocation, a manual human task or a human task executed using a user interface, but it may also refer to an enterprise specific type such as a bundle. A bundle is a concept used by the architects in Danske Bank to describe a service invocation that must be executed a number of times and the process may only continue when all invocations have finished. This concept is similar to the workflow pattern "Multiple instances with a priori runtime knowledge" [8]. The WFM system did not support implementation of this pattern for the first versions of the customer package workflow. Hence, the enterprise extended the commercial WFM system to allow implementation of such a construct directly in the workflow language. Second, the enterprise has defined standards for implementing workflows with regard to logging and exception handling. Throughout the workflow, a specific logging mechanism used by all systems in the enterprise is used to log status information about the execution. For each service invocation or human task, different mechanisms are implemented to handle possible system and business failures. When implementing a service invocation, a human task or a more complex task such as a bundle, the developer has to do the same job again and again. It is the same patterns,

the same code and the same type of information that must be created. This is trivial, time consuming, and error prone. Further, there is a risk that developers do not follow standards, which results in low-quality implementations.

Fault Handling. As in traditional programming, BPEL processes need to take fault handling into consideration. Workflows are executed by a process engine and when errors occur during execution, they have to be handled by the BPEL program. Here, faults fall into two categories, business faults and technical faults.

First, business faults are errors returned from invoked services. Such errors typically occur when services are invoked with incorrect data, some preconditions that are not met, or some other internal conditions inside the services cannot execute correctly. Business errors are recognized in the process based on special return codes from the invoked service. These return codes must be known by the developer to make the correct error handling. As described earlier, such return codes are seldom documented and the information must be obtained orally from the service developer.

Second, technical faults may occur several places in the process. For instance, when mapping data from one variable to another, which is done before all service invocations, values have to be retrieved from the underlying database. Although this is handled by the process engine, the database connection may be missing due to system breakdown which means that an exception is thrown and has to be handled by the process. If this is not considered by the developer, there is a risk of process instances getting into invalid states.

Error handling turned out to be one of the most time consuming activities during development of a BPEL process. Further, it has shown to be of utter importance to avoid process instances getting into invalid states.

Model Synchronicity. During the first versions of the customer package workflow, the developer and the architect manually synchronized the solution model and the code. Much information about the solution had to be defined in both places. At some point, changes began to be implemented directly in the code without updating the solution model and some technical documentation was made in plain text. The original solution model diverged more and more from the actual implementation and hence it became useless as a design and documentation artifact.

Many of the challenges described above are common for software development in general. For instance, missing and imprecise information in solution models are related to not following the development process, and careful fault handling has always been difficult. Specific to BPM and SOA are challenges related to service granularity and reusability. Services must be developed for reusability across the entire enterprise which requires right level of granularity, documentation and use of standards.

4.2 Organizational Challenges

Several of the challenges described above are due to organizational issues. All involved parties in both projects were new to service-oriented and model-driven

development, and as the project was the first of its kind in using BPM, there were no in-house experiences. As stated in section 2.2 the enterprise has defined its own service-oriented development process. To some degree it is similar to the service-oriented analysis and design steps described by Erl [1].

Had the projects followed the prescribed development process probably many of the experienced challenges would have been avoided. However, the challenges of locating existing services, getting the right service granularity, and missing documentation would not be solved by following an appropriate development process as the challenges are caused by other project teams.

Three things were missing regarding the development process; *Education, best practice examples* and *architectural governance*. Both model-driven and service-oriented development are new ways of developing software and requires changes in the mindset of developers, architects and analysts. Such changes are hard to implement and requires much effort from the organization and people.

People involved in the two projects were not educated in model-driven and service-oriented development. Further, the development process was hard to understand, and it was difficult to find out where to get support. In particular, there were no best practice examples available to learn from. Therefore, people worked as they used to. The architect responsible for the first version of the customer package workflow, though, used the development process to document design decisions in the solution model and used it as a contract for what to be implemented. The users and the analysts were not used to this, they worked as they used to and therefore they missed to describe large part of the solution. Following the development process, they would probably have recognized the missing and contradiction parts during initial tests. Today neither of the examined projects have any valid solution models, mainly due to not following the development process.

Developers responsible for services in other departments were probably not educated sufficiently either. Their services were developed as traditional mainframe systems with a new service interface on top which indicated that they neither had been service-orientated when developing. This is obvious for service enabling of existing legacy systems, but it also appeared for newly developed systems. This directly caused challenges for both of the examined projects as it became harder to integrate the services into the workflows. Had the service developers been service-oriented, they would probably have created more well-defined and loosely coupled services which were easier to integrate.

An architectural governance instance would be able to stop the projects early on and guide them on correct use of the development process. If the first version of the solution model for the customer package business process had been through an initial test before implementation, much of the missing activities would probably have been found. Such a governance instance would also be able to guide service developers to develop reusable and documented services, meaning easier integration for other projects.

4.3 Technological Challenges

Further from organizational challenges, several challenges are related to technology as follows.

Complexity. It requires knowledge of many technologies to develop workflows. The developer must understand technologies such as WSDL, XML, BPEL, Java and XPath. Furthermore, complex concepts as transaction control and compensation handling must be understood and how to be used in the workflows. Fault handling, event logging and common enterprise specific patterns and standard must be known, understood and followed.

Technology Evolution. Technologies to support SOA and BPM are still under strong evolution. For instance, the area of BPM is characterized by rapid change in technology. Two radical changes in the basic language in about three years indicate an immature technology. First, the proprietary FDML language was used, which was based on the WSFL standard, a predecessor to BPEL. It was then replaced by BPEL in version 1, and now it is BPEL version 2. Each change has been without backward compatibility meaning much work of converting existing workflows.

Tool Support. There is a significant gap between a solution model of a business process and the actual implementation as a workflow. The commercial tools used are not able to bridge from the solution model to an implementation as they are not extensible to support enterprise specific standards. The developer must interpret a solution model and make the transformation manually based on achieved domain knowledge. Therefore, the same implementation patterns must repeatedly be implemented. When changing the solution, changes have to be applied manually in two places; in the solution model and in the implementation. Effective model-driven tool support should address at least four issues; 1. Allow enterprise specific modeling standards and transformations. 2. Ensure that required information is present in models and that these are valid. 3. Consistency between model and code. 4. Allow smaller changes to be made in generated code. Brahe [9] describes an approach that uses Domain Specific Languages (DSLs) and customized pattern-based transformations for business process modeling and implementation. The approach addresses many of the observed challenges and solves the first three issues above; Using DSLs, enterprise standards are directly available in the modeling tools and it can be ensured that required information is present in a model. Customized transformations make it possible to retrieve the implementation directly from the model as enterprise specific patterns and standards are captured by the transformations. Hence, changes are made only to the model and they can be synchronized to the implementation. Furthermore, repetitive manual implementation work is eliminated. Unfortunately, the tools used, i.e. CoolBiz and Websphere Business Modeler does not support such an approach. A trend among software vendors seems to be that a business process modeled in e.g. BPMN is able to be directly mapped to BPEL using standardized transformations. This approach does not fulfill the four issues and has not been sufficient for Danske Bank.

5 Lessons Learned

Based on past experiences, the enterprise has gained much knowledge to be used for future projects. This is described in the following sections.

5.1 Development Process

As stated above many of the experienced challenges would probably have been met if the project had followed the prescribed development process. As the organizational challenges illustrate, it is important that projects follow a service-oriented and model-driven development process to ensure that services are developed for reusability and that defined solutions are complete. It is not easy to shift from traditional software development to service orientation, therefore it is necessary with sufficient education in using the service-oriented paradigm. Best practice examples are important as examples are one of the easiest way to learn from. As people tend to work in such ways that they achieve short term goals fast, a strong architectural governance function is important to ensure that all projects work in the same direction to also achieve long time goals by developing services that are reusable across departments. The group has learned from the early experiences that goes back about 3 years, and today focuses much on organizational implementation. Project teams are offered education in the development process, improved tool support and guidance by enterprise architects. Further, architectural governance has been improved by having checkpoints throughout the complete development process, where projects are expected to deliver certain development artifacts and participate in events as e.g. static tests of solution models.

Further from not following the prescribed development process, many of the experienced challenges in the two examined projects can be attributed to inexperience. Therefore, having people experienced in SOA and BPM on a project is crucial. At least one person, an architect or a developer needs to master the technology as well as having an understanding of the business scenario. Such a person is able to communicate directly with users and translate requirements to technology and hereby bridge the gap between business and IT. The customer package project has shown that direct cooperation between users and developers is beneficial as much misunderstanding is eliminated when the user can explain directly to the developer about current work practice and the developer understands how to implement it. Many exception conditions were surveyed in this communication. The developer directly understood how the work practice was carried out, and new ideas came up for the solution.

Business Process Reengineering (BPR)[10] is an approach that seeks to reengineer and optimize a business process at the same time as new IT support is developed for the process. This approach has not been used. Actually, the stepwise optimization of the customer package business process seems to be very successful. By implementing the manual business process as it was, where only the coordination of work was automated, made the transition easier for the participating workers. Gradually, the process has been optimized and it has been

easy for the back office workers to adopt to the changes. While the BPR approach may be able to provide higher return on investment, the stepwise optimization gives the possibility to gradually learn from execution statistics, to locate bottlenecks and to introduce changes to the back office workers in a controlled matter.

5.2 Tool Support

While a mature development process is crucial for successful adoption of BPM and SOA, support for the development process by efficient tools is crucial for adoption of the development process. Otherwise, developers will circumvent the process. Tools and technologies related to SOA, BPM and model-driven development are still evolving rapidly. As described previously, manual synchronization of changes between models and code is not an efficient development practice. Tools should provide a high degree of flexibility to allow an enterprise to define and utilize its own modeling concepts and write its own transformations from model to implementation. Such flexibility would allow architects and developers to create precise models based on enterprise-specific standards which by the tools can be transformed to an implementation. As commercial tools have been insufficient to support the development process efficiently, the enterprise has developed several coding standards and tooling extensions to make developers more efficient and the resulting implementation less error prone. This includes a unit-test and simulation framework for testing services and workflows, a validation engine to check a workflow against enterprise-specific coding styles and automatic error fixes, a pattern generator to generate parts of a workflow from a specification and a graphical presentation of process instances used for monitoring during test and operation of workflows. These tool extensions have shown to be very valuable although commercial tool support were preferable.

6 Related Work

Not much experience has been described about challenges in adopting BPM and SOA. A few papers have been describing experiences and challenges on adopting SOA, but none of these have been including BPM. For instance, Mahajan et. al. [11] present lessons learned from 3 years of SOA implementation in a large US city government but do not describe any experienced challenges. Archarya et al. [12] make a more detailed presentation by describing experiences in building an enterprise business application based on SOA. They mention the right level of granularity of services as a key issue. Further, they also point out weaknesses in current tools for building SOA based applications and request for tools that simplifies the complete development process by utilizing higher level tools that are fundamentally aware of SOA. Both issues are in line with what have been observed in this paper. Lewis et. al. [13] discusses common misconceptions about SOA. The intent is to provide a more differentiated picture of SOA and to caution about important issues while creating a SOA strategy. A key point of the paper is, that even if SOA may be the best approach available to achieve

interoperability, agility and reuse goals, building and managing large scale IT systems is still difficult. To the author's knowledge, the only paper describing challenges regarding both BPM and SOA is Woodley et. al. [14] who discus challenges regarding service granularity, transactions, and error and exception handling, though this paper is not based on real experiences.

7 Summary

This paper has described results from an empirical study on early experiences in adopting SOA and BPM in a large organization. The examinations cover both business experiences as well as experiences from within the IT organization.

From a birds eye view, the study has shown business value of using BPM and SOA to integrate systems across different departments and platforms and to automate manual work procedures. By automating the traditional work practice of handling product creations for a customer package it has been possible continuously and stepwise to optimize and automate expensive manual tasks of the process with the result that today only 20% of all products are created manually compared to previous practice. Further, data is automatically carried around between different systems making the work for the back office workers easier and more efficient.

Going from the birds eye view to look inside the development organization and follow the team responsible for implementing the business processes reveals another picture. It shows the complexity and difficulties of adopting BPM and SOA. Many challenges known from traditional programming languages are still present when developing workflows in BPEL. This includes fault and exception handling, lack of documentation of integrated services and synchronicity between solution model and implementation. Business processes implemented as workflows rely heavily on SOA. Therefore it is crucial for easy integration of different services that these have been developed for reusability and are documented properly.

The empirical study shows that although BPM and SOA provide value to the business, they are concepts, methods and techniques that are not easy to adopt. It requires organizational implementation which includes educational efforts, best practice examples and architectural governance to ensure that projects follow the development process and service-oriented guidelines. Further, commercial standards and tools have not yet been found mature to support a model-driven and service-oriented development process efficiently.

References

1. Erl, T.: Service Oriented Architecture: Concepts, Technology and Design. Prentice-Hall, Englewood Cliffs (2005)
2. van der Aalst, W.M.P., Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)

3. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice-Hall, Englewood Cliffs (2000)
4. Zisman, M.D.: Representation, Specification and Automation of Office Procedures. PhD thesis, University of Pennsylvania, Wharton School of Business (1977)
5. Andrews, T., et al.: Business Process Execution Language for Web Services (BPEL4WS). Version 1.1 (2003), http://www-128.ibm.com/developerworks/library/
6. Brahe, S.: Early Experiences on Adopting BPM and SOA: An Empirical Study. Technical Report TR-2007-96, IT University of Copenhagen (2007)
7. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, Chichester (2006)
8. van der Aalst, W.M.P., Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14, 5–51 (2003)
9. Brahe, S., Bordbar, B.: A Pattern-based Approach to Business Process Modeling and Implementation in Web Services. In: Georgakopoulos, D., et al. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 161–172. Springer, Heidelberg (2006)
10. Davenport, T.H.: Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press, Boston, Mass (1993)
11. Mahajan, R.: SOA and the Enterprise – Lessons from the City. In: IEEE International Conference on Web Services (ICWS'06), pp. 939–944. IEEE Computer Society, Los Alamitos (2006)
12. Acharya, M., Kulkarni, A., Kuppili, R., Mani, R., More, N., Narayanan, S., Patel, P., Schuelke, K.W., Subramanian, S.N.: SOA in the Real World - Experiences. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 437–449. Springer, Heidelberg (2005)
13. Lewis, G.A., Morris, E., Simanta, S., Wrage, L.: Common Misconceptions about Service-Oriented Architecture. In: Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), pp. 123–130. IEEE Computer Society, Los Alamitos (2007)
14. Woodley, T., Gagnon, S.: BPM and SOA: Synergies and Challenges. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 679–688. Springer, Heidelberg (2005)