Early Experiences on Adopting BPM and SOA

An Empirical Study

Steen Brahe

IT University Technical Report Series **TR-2007-96**

ISSN 1600–6100 **April 2007**

ISSN 1600–6100

ISBN 978-87-7949-153-3

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web `www.itu.dk`

Table of Contents

Early Experiences on Adopting BPM and SOA: An Empirical Study

Steen Brahe

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

Abstract. Service Oriented Architecture (SOA), Business Process Management (BPM) and Model Driven Software Development (MDSD) are promising concepts that have recently received much attention by industry and academia. Despite the current hype around these concepts, industrial experiences are still limited. Not many studies have been made to show difficulties and pitfalls in adopting them and corresponding technologies within an organization. This paper presents results from an empirical study inside the IT organization of Danske Bank, one of the largest financial groups in northern Europa, which has been a pioneer in adopting a SOA, BPM and recently has updated its complete development process to focus on MDSD. The study shows the business value of automating a tranditional business process using SOA and BPM and following a model driven development process, but it also reveals several challenges during the complete lifecycle of modeling and implementing the business process, i.e. analysing existing work practice and defining the process design, translating the design to code, testing and deploying it, execution, and at last difficulties with change management when requirements changes. Further it describes back offices workers experiences of being a part of an automated process.

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. However, SOA and BPM are concepts that have only been around a few years and there are only limited documented experiences of adopting SOA and BPM in the large scale Experience reports mostly describe the benefits seen from the business perspective while experiences from the development teams involved in implementing the business processes have

not been documented. These reports are primarily made by software vendors and consulting companies, companies that earn their living by selling products and services. Not much is said about challenges in adopting the concepts, techniques and technologies seen from the IT development and the organizational perspective. Further, the experiences of the process participants, or the back office workers involved in executing the business processes have not either previous been reported.

In this paper we present empirical research carried out in the IT organization of Danske Bank [6], one of the pioneering companies in adopting SOA and BPM. Danske Bank has since 2002 been using SOA for all new application development and since early 2004 it has been implementing its business processes in a BPM system. All business and software development is based on Model Driven Software Development(MDSD) [7] techniques where requirements and design decisions are documented as models.

We have through interviews, workshops and document studies examined two large and independent projects both implementing cross departmental processes. The *Customer packages* 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 as 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 packages project is used as the main case to describe the experiences.

The experiences are described from three different perspectives. First, we describe the history of the customer packages process seen from a business perspective. This description is seen from outside the organization and explains the business idea and shows the benefits of using BPM and SOA. Second, we look inside the organization and describe the software development process of getting from previously manual work practice to an implemented workflow that automates the coordination of tasks. We reveal experiences during the complete lifecycle of modeling and implementing the customer packages process as an executable workflow. This involves analyzing existing work practice and modeling the soltion process, translating the solution into code and testing and deploying it. Further, in involves the operation of executing processes and at last difficulties in handling change management when requirements to the solution occurs. Third, we describe the back office workers' experiences of being a part of an automated process and how it has been accepted to work in new ways where the responsibility of handling a complete process has been taken away from them and automated using workflow techniques.

The rest of the paper is structured as follows. In section 2 we provide background information about the concepts of SOA, BPM and MDSD and we also introduce Danske Bank and its IT strategy. In section 3, 4 and 5, we describe experiences from the customer packages project from a business, a development and a worker perspective, respectively. In section 6 we categorize the observed challenges into organizational and technical issues and in section 7 we discuss lessons learned and how both the enterprise and the technology has evolved and how it will affect future BPM implementations in the enterprise. Section 9 contains a summary.

2 Background information

In this section we introduce the concepts of Service Oriented Architecture (SOA), Business Process Management (BPM) and Model Driven Software Development (MDSD). We further introduce Danske Bank, its history and its IT strategy and how it has adopted SOA, BPM and MDSD.

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.
- Solution model. A model of the business process specifying a solution that can be supported by IT.
- 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. With SOA, systems developed on different platforms and technology, e.g. legacy systems, Java and .Net applications, are able to communicate directly without having knowledge about the other's technologies. SOA makes it possible for an enterprise to open up its legacy systems to other systems and services. Although SOA is often realized using web service technology, the principles behind SOA have nothing to do with technology and can be realized in many variants.

As illustrated in figure 1, SOA basically consists of three components, A service provider, a service consumer and a service registry. The service provider

builds a service and publishes its interface specification to a service registry. The service provider, e.g. a department in a bank responsible for risk calculations, creates a service called AssessRisk which they publish to the service registry, a central registry in the enterprise. The service consumer, e.g. a loan application,

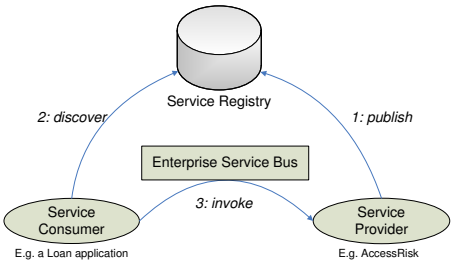

Fig. 1. Basic components of a Service Oriented Architecture

contacts the service registry to discover the risk analysis service and can then invoke it. Invocation of a service is often done through an Enterprise Service Bus [8], which is an infrastructure component acting as a mediator between the consumer and the provider. It decouples a service implementation from the service specification; hence the consumer does not need to know how and where the service provider is implemented.

The wider term *Service Orientation* promotes a development process with focus on expressing business requirements in context of services. Hence, a service oriented enterprise gets a common vocabulary to bridge the gab between business and IT, where services are developed to directly fullfil business requirements and hence provide reusable business building blocks. Such business services can be composed into automated business processes as described in next section.

2.2 Business Process Management

Business Process Management is about managing the complete lifecycle of a business process from analysis of current work practice to the design and implemenation of a redesigned process, and to the execution and monitoring of the process instances. It has been defined by [2] as *Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information.* Although BPM can be realised without IT support, it is often associated with Workflow Management where knowledge about a business process is centralised and documented in a workflow.

Existing business processes, i.e. manual work procedures in an enterprise, often involve a variety of people and applications. Such processes are often not well understood, neither by management nor by the participants. It is difficult for managers to manage and optimize the business processes as the knowledge is

in minds of people and applications. By extracting this knowledge, the enterprise is able to define a model of the current work practice. This model can be used to define a workflow that can be executed and control the distribution of activities.

Business processes to be supported by WFM and BPM systems are restricted to operational and production-like processes that can be explicitly described. Processes that cannot be explicitly described prior to execution, i.e. collaborative and ad-hoc processes, are not suited for traditional BPM systems [9].

With the arise of SOA, BPM can be thought of as an architectural layer on top of SOA and allows composition of services and people into complex business processes with automated coordination of tasks [10]. This is illustrated in Fig. 2. A workflow describes how the business process is to be executed. When a real

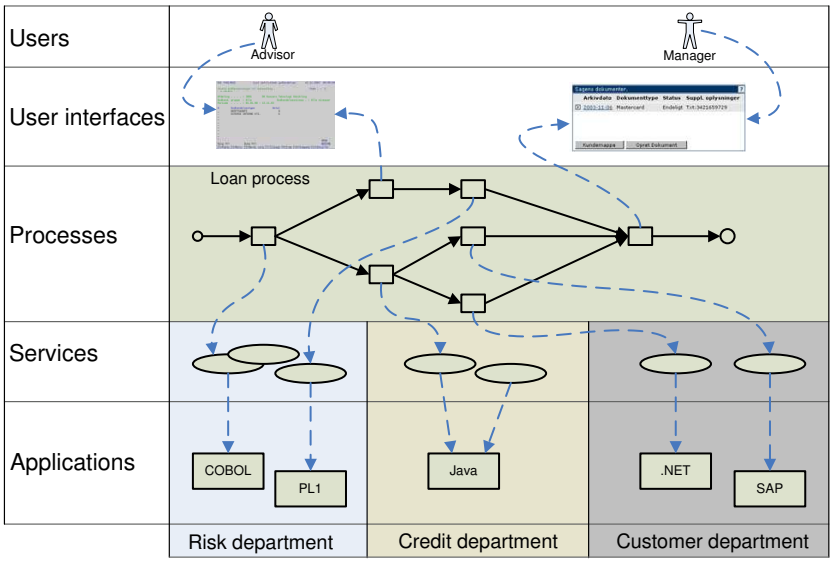

Fig. 2. Business processes as a composition of services and human tasks

business process is initiated by e.g. a customer requesting for a loan, a process instance is created based on the template. This instance knows when to invoke different services and when to ask people to execute tasks. Hence, the previous manual coordination of tasks has been automated.

Automation of traditional business processes is the key aspect in BPM. Here, automation can be divided into two; Coordination automation and task automation. By moving knowledge and responsibility of coordination of tasks from within mind of people into explicitly defined workflows, the coordination of tasks have been automated. People still have to execute the same tasks as previous, but now the process instances are resposible for distributing the right tasks to the rigth people at the right time. Often, after automating the coordination mechanism of a business process, it is further optimised by automating the most

expensive manual tasks inside the process to be handled by automatic services. If all manual tasks inside a process is automated, the process is also called a Straight Through Process (STP). Some of the potential benefits of adopting BPM are

- Systems and people are bound together across departmental silos.
- Centralized process knowledge.
- Homogenous process execution.
- Increased performance by process participants.
- Automatic distribution of manual tasks to right resources at right time.
- Runtime statistics and continued process optimization.

2.3 Model Driven Software Development

When using Model Driven Software Development (MDSD) [10], models are treated as primary software artifacts, from which the implementation is created with the help of software tools. Adopting MDSD in a software development process is claimed to speed up development time and improves the quality of the delivered system.

The Model Driven Architecture initiative (MDA) [11] implements the MDSD approach around a set of technologies and standards like MOF, UML and XMI. Central to the MDA is the idea of model transformations. Defining a transformation from one kind of model, the source model, to another kind of model, the target model, one is able to reuse that transformation for all source models of the same type and automatically generate target models. The target model can also be code.

By defining reusable transformations that captures domain knowledge about the IT platform, languages and coding standards for a specific enterprise, the enterprise is able use these transformations to automatically transform all models of solutions directly to the implementation.

2.4 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.

Today, the group employs nearly 20.000 people and serves around 3,5 million private customers in Denmark, Norway, Sweden, North Ireland, Ireland, and Finland and large parts of the industry and public sectors in these countries. The IT department is at about 1500 people - one of the largest IT development departments in Denmark.

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. The purpose of the strategy is to warrant that the group has the most efficient technological tools for focusing on customers, processes and flexibility and efficiency. The strategy adopts what IBM would call an "On Demand" business meaning that IT solutions must enable flexible adjustments to changing market conditions at the lowest possible cost and a satisfactory time-to-market.

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 and an own Enterprise Service Bus called the Service Integrator.

Support for automating business processes is achieved through a business process management system. The BPM system is based on a commercial product from IBM, but is extended in areas where business requirements were not fulfilled. Business processes are implemented using BPEL.

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 pre-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. In the specification phase, a solution architect defines a solution model, which describes all automatic and all human tasks that make up the business process. 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 in BPEL.

Now after having introduces SOA, BPM, MDSD and Danske Bank, let's look at the experiences of adopting these terms in the customer packages project.

3 Experiences – from the business perspective

In June 2003 Danske Bank introduced a new sales concept called Customer Packages which bundled together a number of financial products, e.g. a credit card and an account, to be ordered by customers. 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 out and signed. The background for introducing this new concept was to enhance customer experience in the branches and to sell more products.

Before this introduction, customers that visited a branch to order e.g. a new bank account, a credit card, or an internet bank had to sign several documents. For each ordered product one document had to printed, read through and signed.

Ordering on average 7 products, many pages had to be printed and signed. The customer could leave the branch with up to 100 pages. This process was not satisfying for the customer as much time was used in the bank, often more than an hour. It was also confusing with the large amount of paper. For the customer advisor, it was not satisfying either. Printing the many pages took time, and after the customer had left the branch, all products had to be created manually using different systems (figure 3). Creating products for the average customer took up to several hours.

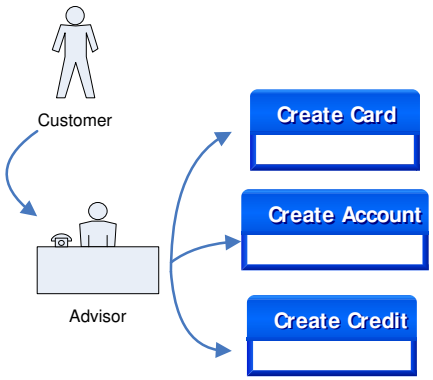

Fig. 3. Customer advisors create all products manually

At some point the business department responsible for cards wanted to sell more American express credit cards. Traditionally the Bank would make a campaign to inform customers about the product, and when it was ordered by a customer, several documents had to be printed such as agreement about the card, and different agreements of information exchange between the different systems in the bank. There was a huge desire in the organization to improve the customer experience and to speed up the process of ordering cards and other products. One day some people at the business department came up with the idea to make a common document where different products were bundled together in packages. Packages were defined to target different customer segments and contained default and optional products. The name "Customer Packages" was born. The same concept had earlier been used in one of the branches of the group with success. The business department analyzed the sale and product creation process in the branches and decided to make a solution without any IT support. The reason was to be able to introduce Customer Packages as quickly as possible for the customers instead of delaying the market introduction until IT support was developed and ready to use. They came up with a Microsoft Word document, which could be customized by the individual branches for greater flexibility. Using the template, the adviser could define all products for a customer in one document. Compared to previous practice, much less paper was printed

and the customer had only one document to sign. Further, the time spent on the sale process was reduced. The Word template was a success. The number of credit cards, as well as other products sold were increased significantly due to the improved presentation for the customer; It was easier to introduce different products and only one document had to be printed and signed.

3.1 Centralized backoffice organization

The customer experience had increased significantly, but the customer advisors still used a large part of their time on creating products after the customer had left the branch. It was difficult to get time to meet with customers because of the time consuming task of creating the products. Someone asked "Why can't we use our time on advising customers and selling products. Someone else can create the product". The business department agreed, and in August 2003 a new back office department named "Customer agreements" was established to handle creation of products ordered through customer packages. Now, after getting the customer package document signed, the customer advisor send the document as an attachment by email to the Customer Agreement department. Here, about 30 people worked on receiving these documents, reading through them and manually creating the products in the different systems (figure 4). The customer advisors were happy with the new back office function. Now, they could use their time more effectively on advising customers and selling products.

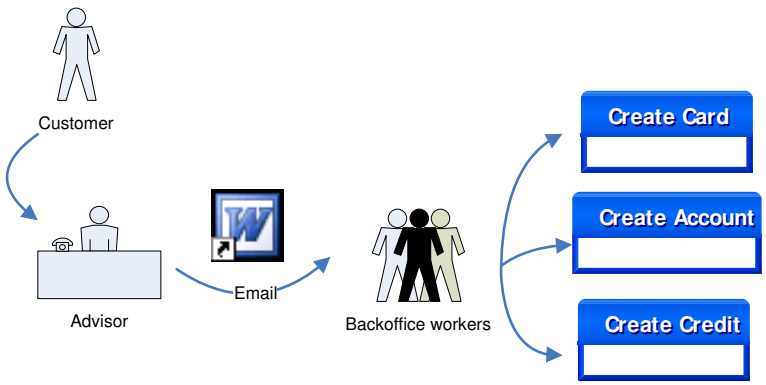

Fig. 4. Backoffice workers create all products manually after getting word document from customer advisors

3.2 A new Case Transfer System (CTS)

The new back office function was an improvement over previous practice but back office workers experienced many challenges with documents received from

the customer advisors. Often, the advisor had filled in misspelled or contradictory information or had forgotten to define required information. The back office workers had to read through the complete customer package document to validate it. Often, they had to phone the customer advisor to get the correct information to be able to create the products. Further from being time consuming, the procedure of validating documents and communicating with the advisors was not streamlined. Each worker had separate ways of validating the document.

There was a great need to improve the process of handing over customer packages from customer advisors to the customer agreements department. Therefore a new system, called CTS (Case Transfer System) was created. This system allowed the customer advisor to choose customer type, getting a new document for this specific customer type, and to send it to the customer agreement department directly after filling in information about the customer and the ordered products. The system was able to extract information from the word document and to make an automatic validation check on the entered data such as required information, marked checkboxes, etc. In case of data conflicts, the document was sent back to the customer advisor without involving the customer agreement department (Figure 5). This was a great improvement to previous practice. For

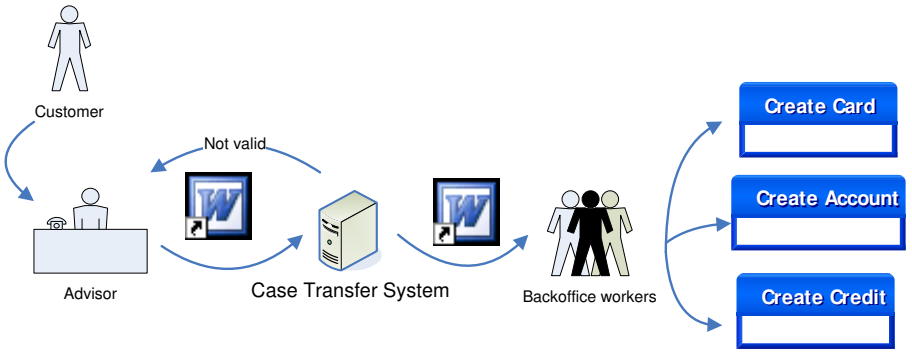

Fig. 5. Backoffice workers create all products manually after getting validated word document from the Case Transfer System

the advisors it was an improvement as they were informed directly about errors in the document and they were able to correct the errors at once, and it was an improvement for back office as they received documents in a streamlined manner with fewer errors. Further, the document was trimmed for unnecessary information so it was easier to find relevant information. With the new solution, it was not necessary to read through the complete documents to ensure validity which made the back office workers more effective.

3.3 Process Automation by Workflow Management

The process was now streamlined for the customers and the customer advisors, but at back office, the handling of the received doucumentes was made differently from customer to customer. The back office department basically handled the product creations in the exact same way as the customer advisors previous did it. Products were created in an arbitrary sequence from package to package, although there might have been an optimal sequence in which to create the products. Further, the same information about the customer, accounts, etc. had to be entered manually in the different systems responsible for creating the products. This often resulted in misspelled information and trivial work as the same information had to be entered several times.

At the end of 2001, two years earlier, the enterprise had decided on and implemented a Service Oriented Architecture at which all new IT development is targeted. Further, at the end of 2003, the enterprise was ready to implement business processes using a BPM system that was based on service orchestration. The people from the workflow department were placed next to the people that developed the CTS system. A discussion between these people showed that the process of creating products for customer packages was an ideal process to be automated using workflow technology. The process 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 all the different systems together and eliminate the need for entering the same information repeatedly in different systems. It was decided to use the manual customer packages creation process as a pilot for the new BPM system. The workflow 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 cariied 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 figure 6.

The workflow management system was an improvement of the 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 sys-

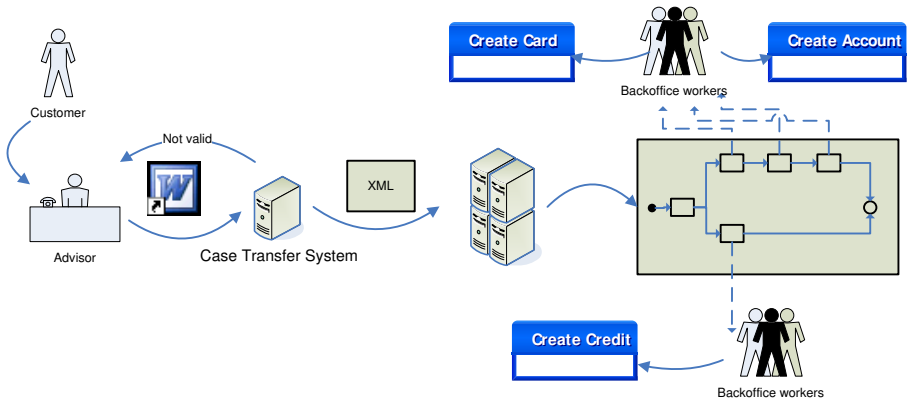

Fig. 6. Workflow enabled backoffice process with backoffice workers creating seperate products

tems 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 (Figure 7). The back office workers have saved much time, which today is used for other activities.

3.4 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.

Currently, the workflow process is used for handling all customer packages in Denmark, while customer packages from brands in other countries are still handled manually though the CTS system. The concepts and the marketing and branding campaign for customer packages are the same through all countries,

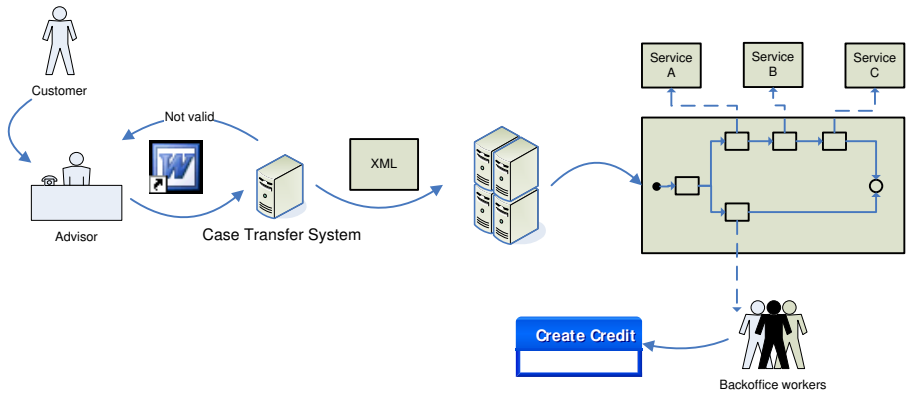

Fig. 7. Workflow enabled and optimized backoffice process with 80% automated product creation

but the legal aspects of the process differ and the products may be different. Hence, the customer package workflow must be created specific for each country. For instance, it requires an elektronic key and a password to have internet bank account in Denmark. The password is sent by a recommended letter. In Sweden, logging into a netbank requires an activecard, a card with one-time-use keys. This card has to be printed and send by letter. In Denmark, cards are created by back office and the card pin code is sent by a seperatte letter by a third party company. In Norway, the customer has to be present physically at the branch to stamp the pin code into the card. These differences must be learned and different processes must be defined for different countries. It is planned to implement customer packages for other countries, i.e. Norway, Ireland, Sweden and Finland and it is prioritized with other requirements for the customer packages.

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 2000, of which about the half 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. Both the customer satisfaction and the efficiency of customer advisers and back office workers have improved significantly. The use of BPM has proved to be of real business value.

4 Experiences – from the development perspective

After having described experiences from the business perspective, let's look inside the IT development organization and evaluate their experiences.

As several people have been interviewed about their role and participation in the project, we have organized the experiences into different phases of the customer package lifecycle:

- From business model to implementation
- Test
- Deployment
- Operation

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 services 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.

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 common standards have been used for return codes, exception handling must be implemented differently for each service invocation.

Service granularity and reusability The services to be invoked from the workflow are mostly provided by legacy systems. Such services are often at a low level of granularity and are difficult to reuse. Some services require up to 100 input parameters without any documentation, which makes it impossible for anyone except the service developer to invoke the service. The developer may explain that only a few of the 100 input parameters have to be set, but this is not documented anywhere else. In contrast, other services take one input parameter, which must be an XML data structure. Such services implicitly expect the XML structure to contain several input parameters.

The above challenges makes it time consuming to locate, understand and invoke a service correctly.

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 repeated 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. The developer knows the pattern of implementing a bundle as he also knows the pattern of invoking a service or a human task. 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 complex tasks 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. Much of the repeated implementation work could be spared if the development tools allowed customization of the transformations.

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 cases, where process instances get into invalid states.

Model synchronicity During the first versions of the customer packages process, 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.

4.2 Test and deployment

As with all system development, a workflow should be unit tested by the developer before deploying it to integration test. A workflow is integrating different systems and services across an enterprise; therefore it is necessary to create test stubs for simulating return values of these services. Ideally it should be possible to make a unit test script defining input values to the workflow and return values for all involved services. Currently, the tools used by the developers do not allow simulating service invocations meaning that unit test of the workflow is not sufficiently done. After finishing unit testing, the process has to be deployed to a process engine, which is able to create instances of the process for concrete customers. When deploying such a workflow, deployment code must be generated, data sources and database table created and then the application can be deploying, installed and started. The deployment takes hours, requires deep technology understanding and the risk for making mistakes is large. If something goes wrong under deployment, the application has to be uninstalled, and a new application must be generated. In addition to the complex deployment process based on commercial tools, the developers also manually has to define data and deploy changes related to enterprise specific extensions to the BPM system. This includes data for creating allocation rules, tables that must be defined, etc. These changes are made in different systems.

4.3 Operation

Throughout the three years of running the customer packages process in production there have been several operational issues, mainly due to immature vendor products and a complex system infrastructure.

System breakdowns After the workflow has been deployed to a production system, the actual customer packages ordered by customers are ready to be handled by the workflow. The business process engine which is responsible for executing the process instances heavily rely on other systems such as database and message queuing systems. If these systems have problems, the process engine is not able to operate. Several times there have been problems with the database and the message queuing system, which has required vendor support to solve. The department responsible for the process engine has not for long periods been

able to continue development of the infrastructure due to solving operational challenges, challenges that origin from bugs in vendor products.

Transactions In the enterprise, cross platform services are invoked using a protocol based on message oriented middleware. Such invocations are always asynchronous and transactions cannot be ensured during invocations. If a service is invoked from a process instance, and a timeout is received, or a system breakdown occurs during invocation, the instance cannot be sure weather the service was invoked or not. If the service is updating, e.g. inserting an amount on a customer account, it cannot be invoked again as it would twice insert the same amount on the same account. This is a consequence of using SOA in a heterogeneous environment. The system owner of the updating service has to be contacted to find out if the service was called or not. It is possible to change the communication protocol to be synchronous to enable commit scope. But the database administration department does not allow this due to too expensive locks on the database.

Service and systems dependencies Running process instances are dependend on the other services and systems to be available when they invoke either a service or set up a human task that depends on a user interface. These services and systems may not be aware that they are being used by running process instances, which can run for months and even years. This is a consequence of SOA and the idea of loosely coupled services. Changes to the interface of such services and systems will make the running process instances fail, which may cause serious operational breakdowns where processes cannot be executed.

Another challenge of service dependencies is debugging and determination of fault locations. A failing service may have been invoking other services, which again have been invoking other services. To find out in which service an error has occurred is a difficult task as services are distributes across several platforms and technologies.

4.4 Change management

The customer packages process has experienced continuously optimization and has recently been deployed in the 6th major version. Each time an optimized version has been developed and deployed, it causes the following challenges regarding change management.

Model and implementation synchronicity . Development tools do not allow synchronicity between a solution model and the implementation. This means that changes have to be defined several places which causes the risk of inconsistency between model and code as the translation is done manually be the developer. Because of the missing tool based link between model and code, updates are often made directly to the code with the result that the solution model becomes invalid.

Long running process instances . As process instances may involve human tasks, they can run for weeks, month or longer. When deploying a new version of a process template, new process instances will be based on the new version, but instances created prior to the deployment will still be based on the old process template. If both the new and the old template depend on a specific service, e.g. a CreateCard service, which has changed its interface to work with the new template, then instances of the old template will fail when they try to invoke the CreateCard service using the old interface, which does not exist anymore. Old instances either have to be migrated manually or to finish to be able to avoid this conflict. Managing long running instances is very difficult and requires much experience to understand the complications.

Time consuming . Even minor changes to a process template may take much time to deploy. In principle, no matter the size of the change, a process template must be tested thoroughly, first in a test environment, then in an integration environment before it is deployed to production. It has been experienced that a minor change like replacing a human task with a service invocation takes several weeks to get into production due to complicated test and deployment procedures.

4.5 Technology limitations and challenges

It was in 2002 decided to use a commercial BPM platform from IBM as the foundation for the workflow management system in Danske Bank. This system had to be integrated into the enterprise's infrastructure and had to support the business in managing business processes. Not all architectural and business requirements were fulfilled by the product. The workflow infrastructure department was responsible for integrating the WFM system with the existing proprietary SOA infrastructure and to extend the tool where it did not fulfill the business requirements. The following extensions had to be made to the product:

Extended flexibility for creating processes . When initiating a process instance of e.g. the customer package workflow, this instance represents the business case for an actual customer and cannot be changed or extended with futher information or process instances. This was not sufficient for the business that would be able to initiate an arbitrary number of process instances based on different workflows and link them logically together to the same case. Therefore, a Case service and a Process Instance service were created to maintain relationship between the case, documents and different process instances. Further, as it is difficult to model a complete business process with all special cases and exceptions, the business also required possibility to create Ad Hoc manual activities on the fly.

Extended flexibility on resource allocation . The initial version of the process engine only supported static and limited rules for allocating people to

human activities in a workflow. This was not acceptable as allocation rules were required to be changed dynamically during process execution. Therefore, an own human task manager, called Allocation Service, was created.

Language limitations . Different process patterns were not possible to implement using the first versions of the language. Therefore, extensions have been created that makes it possible to implement workflow patterns that fulfill business requirements. Today, with the arrival of the BPEL 2 standard some of these extensions are now possible to construct directly.

Integration with own portal . The portal to handle human tasks should be integrated with the existing enterprise wide portal system developed on the mainframe. Because of this requirement and because the own developed allocation service, the human tasks portal, called Activity Overview, was developed which required many ressources.

Process monitoring and administration . Possibilities of monitoring and administrating running process instances were very limited in the first versions of the product. The IT department therefore developed both a monitoring and an administration console with all the required functionality.

SOA Integration . The enterprise has developed its own SOA and enterprise service bus based on proprietary standards. Therefore, there have developed an adapter that is able to translate from a web service invocation to a proprietary service invocation. This adapter is used by the BPM system to allow invocation of proprietary services as web services. The proprietary SOA does not support transaction scopes for invoking services. This has caused the workflow infrastructure department to define standards for service invocations such that the risk of invoking a service twice is minimized.

Above extensions have been developed over a four year period and has required a large effort from the infrastructure team. The infrastructure now fulfills business requirements and provides higher flexibility compared to the basic BPM system provided by the vendor. But the flexibility also means that extensions must be maintained and updated each time new major versions are available from the vendor. Both the examined projects were developed before common industrial standards as the Business Process Execution Language (BPEL) were available. Processes were implemented in the IBM proprietary FDML standard, a continuation of their FDL standard from the MQSeries Workflow product and a predecessor for the BPEL standard. The FDML standard was only used a short time in IBMs products before it was replaced by the BPEL standard which in the meantime had become stable. The change in technology meant that all processes had to be converted to the new standard as there was no backward capability with FDML. This results in complete recoding and testing of all processes. The customer packages project has been through this conversion which

has been expensive and time consuming. Now, another technology change has appeared. Version 2 of BPEL has been adopted, and again, the vendor does not provide backward compatibility. This means that all processes again have to be converted from one technology to another. While the customer packages project has chosen to convert to BPEL1, the account settlement project has decided to wait for the BPEL2 version to be supported by the enterprise infrastructure as tt requires much effort to change technology.

5 Experiences – from the workers perspective

Being a human part of the automated customer packages process was a new way of working for the back office workers. From being in control of the complete business process of creating all products for a customer, now they got separate activities to be executed from the human task portal without any knowledge of the context in which the activities appeared. Using the task portal they could take an activity and were directly linked to the responsible system, where they could handle and execute the activity. Further, from providing the link to the system, the portal also transferred data, data that were available in the process. This was positive, as the worker did not have to open the business system manually and did not have to enter data manually into the system. Further, during the optimization of the business process, many of the activities were automated.

During the first phases of introducing the workflow, the users experienced long response times and unstable systems. Today, the long response time challenge has been solved, but periodically, systems become unstable which is a stress factor. During business process optimization, manual activities have been optimized to be handled by automatic services. In the first version of such an automatic service, for instance the creation of cards, it has been experienced that in some cases it fails. Such a failure has to be taken care of manually which is annoying when the mind has shifted to work in new ways and suddenly it has to shift back to work as before. Now, it has been recognized that the card creation service cannot handle all cases; 4% of the cases are in such a shape, that the card must be created manually. The service does not fail anymore; instead, depending on the data the process handles the distribution of the task, either to the card creation service, or to the back office workers.

Workflow systems advocate activity centric business process handling where the workers do not have knowledge of the business process. It has the benefit that individual tasks can be distributed to different people and departments and that a manager is able to control the resource allocation. The activity centric way of working has not been accepted by workers from either of the examined projects. Independently of each other, workers from the two projects have found ways to handle activities in a case centric manner. They focus on one customer case; through the activity portal they locate all activities that belong to the specific case and handle them all before starting on the next case. Although it is against one of the principles in workflow management, there are both psychological and practical reasons for the workers to work in the case centric manner. First, it does

not feel right not to know the larger context in which one is working. Process knowledge give a better understanding of what one is doing, and handling all tasks on one case give a better feeling of doing a complete piece of work. Second, for practical purposes, one person handling a complete case with all its activities has the advances that this person knows exactly what has happened during the complete process lifetime in case that errors or exceptions occur. This person acts as a process responsible and is able to handle exception cases and failures faster because he or she has knowledge of the complete the process history.

5.1 Involvement in development process

From early on, back office workers were directly involved in the development of the customer packages process. This has been a good experienced for both the developers and the workers as they found out that they could learn much from each other. For the developer, the close cooperation made it smoother to integrate systems, as he got an understanding of the practical business process, how people worked, and how to use different business systems. The worker could directly show which systems were used for different manual activities, which data should be provided, and what preconditions it had. Much of this information was lost when the traditional communication protocol was used; from the worker to business analyst to architect and to developer. The workers felt that they were involved and had influence. Further, they got an understanding of the complexity of the involved systems which made it easier to accept periodically unstable systems and that it took time to develop. The user involvement made them very positive about the project and made it easier to get acceptance and participation from the complete back office department.

In the account settlement project, the users were not involved in the development process. The business processes all contained much detail knowledge about rules and exception cases. Prior to the project, the users did not believe that it was possible to automate the business processes involved as they were too complex. Late in the project, the users were involved, which resulted in changes to the defined processes. Such changes could possible have been caught earlier, if the users had been involved in the development process..

Despite some user skepticism about process automation it proved to be possible. Today, all workers have adopted the new way of working and despite periods with unstable systems, people agree that they have become much more effective and support the business better than with previous work practice.

6 Nature of challenges

We shall now seek to organize the problem into organizational and technical.

6.1 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 it's kind in using BPM, there were no in-house experiences.

Three things were missing, that probably caused many of the challenges; *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 implant and requires much effort from the organization and people.

People involved in the two projects were not educated in model driven and service orientated 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 customer packages, though, used the new 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 legacy systems with a new service interface on top which indicated that they neither had been service orientated when developing. This directly caused challenges for the both 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 customer packages 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.

6.2 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 tool support would have made it possible to define a customized tool based transformation and apply it to the solution model to retrieve the implementation directly. This is the basic idea behind model driven development, which unfortunately is not sufficiently supported.

System architecture The BPM system architecture is complex due to own developed extensions and parts of the standard products that are not used. This causes the developer to make the implementation to take these systems into account. It heightens the requirements for flexible tools to support a custom domain and not primarily solutions created for the standard product only.

Runtime environment The runtime environment is complex and distributed at different platforms. This makes it difficult to locate the origin of errors when they occur. Further, the errors are mostly caused by problems in the vendor systems, either the database system, the message oriented middleware, the application server or the process engine. To determine which product is failing is difficult and to determine the cause of the failure is even more difficult as the products are not open. The heterogeneous environment also makes it impossible to guarantee transactions. This causes challenges when invocation of an updating service fails. It must be examined if the service was invoked before it can be invoked again.

Challenges related to organizational issues can be addressed directly by the enterprise while it is harder with to address technical issues due to limitations

in commercial products. In the next section we will describe what the enterprise have learned from the initial experiences and initiatives taken to solve the challenges.

7 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.

7.1 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 develop 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 example 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 reuseable across departments. The group has learned from the early experiences, that goes back about 3 years, and today focus 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.

The customer package project has shown that direct collaboration 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. The stepwise optimization of the customer packages 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. This is in contrast to Business Process Reengineering [12] which prescribes that the process should be reengineered from the initial version.

7.2 Right kind of people

Many of the experienced challenges in the two examined projects can be attributed to inexperience. Therefore, having experienced people in SOA and BPM on a project is crucial. At least one person, an architect or a developer, in the project 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 gab between business and IT.

7.3 Insufficient commercial tool support

Tools and technologies related to SOA and BPM are still evolving rapidly. Good tool support is closely related to Model Driven Development, also a research area under evolution. To gain an efficient software development process, better tools and mature underlying technologies are needed. It is believed that more real life experience is needed to guide the development of tools and technology in the right direction.

7.4 Tool extensions and standards

Tool support for modeling and translating the solution to an IT implementation has not been sufficient. Therefore, the enterprise has developed several coding standards and tool extensions to make developers more efficient and the resulting implementation less error prone. The following tools have been created:

Service tester . An application to script and automate unit tests of services. It enables the developer to create reusable tests independent of the service implementation. The scripts are developed in a specialized XML based language for service scripting and are stored as plain files in the service library together with the service documentation.

Service Simulator . A server component that is able to configure a test server to use test stubs when services are invoked from a process instance. The developer is able to configure the simulator from a web browser by providing service response messages as XML.

Workflow validator . An eclipse based tool that evaluates a workflow against corporate coding standards. The validator is e.g. able to check that the process has been defined as long running, that log and event information has been configured correctly and that error handling is appropriately done. These validations are made before deployment to a test server and have shown valuable as many errors have been caught before deployment, errors that previously were caught as late as in production when unexpected scenarios appeared.

Pattern generator . This tool captures different reoccurring enterprise specific implementation patterns. Using the pattern generator, the developer can based on information from the solution model enter data into wizards, which then generates implementation code, which only need some configuration before being complete. The generator has decreased development time of certain patterns from hours to minutes and heightened the quality.

Graphical administration client . Understanding the history and current state of a running process instance can be hard using a plain list of activities that does not indicate the sequence of execution, decision points and concurrency. The enterprise therefore developed a graphical monitor that is able to show a graph of a running process including the sequence and states of the different activities. Using the client one gets a quick understanding of the history and current state of the process.

Above tools have shown valuable as development time has decreased significantly and the quality of code has increased. Most of the tools were developed before and under development of the first version of the customer packages process. Today tools provided by the vendor have evolved but only the service simulator has been made obsolete. The rest of the extensions are still used and provide value for the organization.

8 Related Work

Not much experience has been described about challenges in adopting BPM and SOA. Most available information is from software vendors describing only promised benefits of 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. [4] 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 [3] make a more detailed presentation by describing experiences in building an enterprise business application based on SOA. They mension 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 is in line with what have been observed in this paper. Lewis et. al. [5] 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 regaring both BPM and SOA is Woodley et. al. [13] who discus challenges regarding service granularity, transactions, and error and exception handling, though this paper is not based on real experiences.

Model Driven Development is an important and related area of research that seeks to provide better tools for modeling solutions and generating the implementation. The Model Driven Architecture [11] initiative by the Object Management Group is an implementation of MDD around a set of open standards and technologies as UML, XMI and MOF.

9 Summary

We have in this paper described results from an empirical study about early experiences in adopting SOA and BPM in a large organization. Our examinations cover both the business experiences, experiences from within the IT organization as well as experiences from backoffice workers participating in the business processes.

Seen 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 workflow enabling the traditional work practice of handling product creations for customer packages it has been possible continuously 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 processes reveals another picture. It shows the complexity and difficulties of adopting SOA and BPM and following a model driven software development process. Business processes implemented as workflows rely heavily on SOA. Therefore it is crucial for easy integration of different services that these are developed for reusability and are documented properly. This has not been the done in the examined cases and has caused lots of challenges for the process developers.

Experiences from the back office workers has shown that they become more efficient when working as part of the workflow compared to earlier practice were they had to control the sequence of tasks to execute and to carry around information between different systems. They have not accepted to work as part of the automated process without having any knowledge of the overall process. They still work case based and have found ways to ensure that one person is responsible for handling all activities for one customer.

The empirical study shows that although BPM and SOA provide value to the business, it is 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 are not yet 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 (2005)
2. van der Aalst, W., Hofstede, A., Weske, M.: Business Process Management: A Survey. In van der Aalst et al., W., ed.: BPM2003. Volume 2678 of Lecture Notes in Computer Science. (2003) 1–12

3. 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: 3rd International Conference on Service Oriented Computing (ICSOC2005). Volume 3826 of Lecture Notes in Computer Science. (2005) 437–449
4. Mahajan, R.: SOA and the Enterprise – Lessons from the City. In: IEEE International Conference on Web Services (ICWS'06), Los Alamitos, CA, USA, IEEE Computer Society (2006) 939–944
5. 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), Los Alamitos, CA, USA, IEEE Computer Society (2007) 123–130
6. : Danske Bank, http://www.danskebank.com (2007)
7. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S.: Model-Driven Software Development: Technology, Engineering, Management. Wiley (2006)
8. Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., Verschueren, P.: Patterns: Implementing an SOA Using an Enterprise Service Bus. IBM Redbooks (2004)
9. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall (2000)
10. Havey, M.: Essential Business Process Modeling. OReilly Media, Inc. (2005)
11. MDA: Model Driven Architecture, Object Management Group, www.omg.org/mda/ (2007)
12. Davenport, T.H.: Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press, Boston, Mass. (1993)
13. Woodley, T., Gagnon, S.: BPM and SOA: Synergies and Challenges. In: WISE2005. Volume 3806 of Lecture Notes in Computer Science. (2005) 679–688