Empowering Full Scale Straight Through Processing with BPM

Eric D. Schabell[1] and Stijn Hoppenbrouwers[2]

[1] SNS Bank, Postbus 70053, 5201 DZ 's-Hertogenbosch The Netherlands
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. The SNS Bank (the Netherlands) has made a strategic decision to empower her customers on-line by fully automating her business processes. The ability to automate these service channels is achieved by applying Business Process Management (BPM) techniques to existing selling channels. Both the publicly available and internal processes are being revamped into full scale Straight Through Processing (STP) services. This extreme use of online STP is the trigger in a shift that is of crucial importance to cost effective banking in an ever turbulent and changing financial world. The key elements used in implementing these goals continue to be (Free) Open Source Software (FOSS), Service-oriented architecture (SOA), and BPM. In this paper we will present an industrial application describing the efforts of the SNS Bank to make the change from traditional banking services to a full scale STP and BPM driven bank.

1 Introduction

The SNS Bank in the Netherlands is making a strategic move to automate her support and selling channels to provide her customers with modern on-line services. Realizing that it will take more than just an on-line web shop to excel in the financial world, she has also moved to automate many internal processes. The key elements used in implementing these goals are full scale Straight Through Processing (STP) [1] and Business Process Management (BPM) [2].

In this paper we will present the efforts made to change from traditional banking services to a full scale STP and BPM driven financial institution. We begin in section 2 by clarifying what *full scale STP with BPM* means and why this is of importance to the future of SNS Bank. In section 3 we take a closer look at our case study, the *STP Purchasing* project. We will provide some insights into the application of STP with BPM within an open source development environment, discuss the component architecture, take a look at our process modelling steps, examine how we utilized customer testing, and conclude with an overview of some general empirical data. We will present our experiences, both good and bad, in dealing with a large BPM implementation. As can be expected, there will always be challenges to be met when such an expansive shift in strategy is being implemented and we start our tour in section 4 of the issues encountered

in the project. Section 5 will discuss the brighter side, outlining the positive impact that this project has had on both business and technical realms. This will leave the reader with a good idea of the challenges involved, hopefully helping in implementing other industry BPM applications. We will take a into the futrue in section 6 and provide an overview of ongoing development. Section 7 is a look at moving ahead while applying the lessons we have learned.

2   Full Scale STP

The application of STP with BPM is not a new phenomenon in the financial industry, with other banks having reported some success with relatively straight forward on-line financial solutions [3,4]. Some are even dreaming of taking on the more challenging processes within the banking industry, such as mortgage processes [5]. The difference between these types of solutions and the one presented here concerns complexity. We offer the follwoing definitions:

Definition 1 (Business Process Mangement). *Business Process Mangement is focused on aligning business processes to the customers want and needs by applying methods, tools and solutions.*

This is a simple and straight forward look at how we intend to apply BPM within our organization.

Definition 2 (Straight Through Processing). *Processing a business transaction automatically, without requiring people to be involved in the process. The purpose of STP is to create efficiencies, eliminate mistakes, and reduce costs by having machines instead of people process business transactions.*

This definition is in line with most of the definitions we have encountered in the financial world [6,7,8]. It will work fine as a beginning definition of how we construct our processes, but we need to refine it a bit for real world financial business processing:

Definition 3 (Full Scale STP). *A straight through process (STP) implementation that requires the solution to encompass a wide range of system integration and will include human tasks which embody the complex decision making that automation either cannot legally implement, or is precluded by technical limitations.*

We exclude cost as a factor to determining if an implementation is full scale STP or not. We feel that cost, in terms of time, money, or other value risk, is a business concern that is not related to complexity, but rather to some current operational or environmental situation (i.e. budgets, deadline pressures, politics, environment, etc.).

   The drive to push for full scale STP with BPM is multifaceted. The leading goals are cost reduction, manpower reduction in business processes, removing potential (human) mistakes, and channel independent processing. Users should

experience such processes as transparent, quick, simple, directly usable, and should be able to complete their task in one attempt.

SNS Bank is targeting effective and efficient processing where as much human intervention as possible has been removed. The customer will be kept informed at crucial process steps, communication always being an important factor in customer experience. For the cases that are exceptions or fall out of STP processing, there will be clear and predefined processes to ensure expeditious handling. Last but not least, the entire communication process is as paperless as can be. This encapsulates the SNS Bank's idea of full scale STP processing.

As Heckl and Moormaan [9] conclude "...long term success cannot be achieved without the development of new business ideas, innovative products and services, and customer retention." We believe that such success can only be achieved if BPM techniques are fully integrated. Full scale STP with BPM will continue to be expanded on and implemented throughout the range of products, sales channels, and business processes that affect both customer and customer support. We believe that the time for full scale STP with BPM is now.

3   A Case Study

In the beginning of 2007 the first full scale STP project at SNS was launched, with the goal of putting four new savings products on-line at the start of 2008. This project is known as *STP Purchasing* and will provide us with a case for closer examination of full scale STP with BPM. This section will present the component architecture, take a look at how the process was modelled, show how customer testing was used to verify the solution, and provide some empirical data on the results.

3.1   Overview

The goals for this project were: for a customer to be asked as few questions as possible during the purchasing process; that the entire process would be completed within a maximum of five clicks in the on-line website; and that the customer would be kept informed during all crucial steps in the process with clear, directed communication relevant to a specific purchasing process. A further desire was to maximize paperless communication with the customer. It was essential to maintain as short a processing time as possible, with processes that land in human action stages causing no more than one day delay. It should be volume independent, deliver reusable processes, reusable services, be multi-label, and multi-channel. Above all, the project should provide a full scale STP solution with a maximum degree of automation.

With our definition of full scale STP [definition 3] in mind, we already have an idea that the process is not free from human tasks. There are several instances in which we could not avoid having human interaction as part of this process. The resulting challenges will be discussed in more detail later on in this section.

The project resulted in a general end-to-end purchasing process, initially for savings products, and a new process for document scanning and storage.
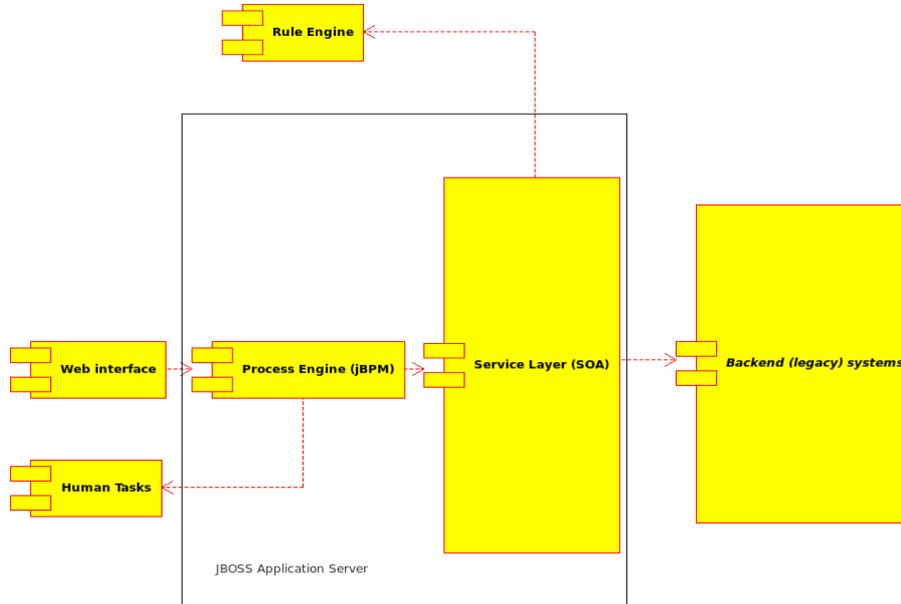
Fig. 1. STP Purchasing architecture

A purchasing request database implementing the data model for each processing request was delivered along with a BPM process flow; a web front end was created for the initial savings products, and the relevant SOA services. A new department was created, called *Process Management Evaluation and Processing.* Total project IT investment was 14,000 hours.

3.2 Architecture

The SNS implementation environment for full scale STP with BPM is one of pure Java [10]. The emphasis is on building solutions within the bank's own IT department, making use of Free Open Source Software (FOSS) where possible, achieving reusability of existing applied solution components, and using best of breed components when forced to shop outside of our existing code base.

There was a shift in component strategy in 2004 from three main commercial suppliers to one where FOSS components are preferred when possible. Open source is now quite pervasive throughout the solution architecture of all SNS projects. Furthermore, the development environment and tooling used to implement the solution consists of almost only FOSS. This is outside the scope of this paper and will therefore be excluded from further discussion. The component architecture as shown in Figure 1 is a very generic and high level view. We will discuss the components as shown, from left to right.

Web interface. The entry point for any full scale STP application is the web interface as seen by the customer in the on-line banking website. This is a Java

based website that makes use of a content management system. In the STP Purchasing project it provides the user with the option to apply for one of four saving products. If placed, a request is gathered together with user information, verified through various web services, and then using a web service it is deposited into the *Request Database.*

One might expect that a request is submitted directly to the *jBPM process engine,* but each request is put into a database to ensure that no single customer request is every lost due to the process engine begin unavailable. This is required by a banking regulation that ensures that no risks are taken with customer submitted information. We must and will always be able to trace and audit every single step in the chain of events from customer request to product delivery. This small design step has been left out of the component diagram as it happens underwater and is of little importance to industries where intensive risk protection is not needed; we mention this in the interest of completeness.

Human tasks. A human action interface was implemented to provide functional administrators with the ability to deal with tasks as they drop out of the automated process for various reasons. Furthermore, Service Center employees provide input to the system through another interface with the document monitoring section of the process flow. Communication with the customer can require for a human task to be performed, such as customer's reply to questions which needs to be judged on completeness, correctness, and validity. This input to the jBPM process flow causes pending processes to be triggered into their next stages, to be stopped, or to be restarted. The interfaces have been created in-house by the project development team.

Within the project process definition it is always possible to encounter problems, planned or not, that need human intervention to solve. This intervention is called a human task, where the process is dumped into a task bucket for further action by an authorized person. We refer to the need to invoke human tasks as having the process *fall out* of the process flow. This fall out can then classified as either technical or functional. The first is often related to some error in processing a request within a process step, the latter is related to a problem in the application flow logic. When we look at full scale STP we are concerned with processes that by definition contain planned functional fall out points in their process descriptions.

STP Purchasing supplies a web based Java interface that provides an interface for humans to manipulate the tasks that they have been authorized to view. This component makes use of web services in the SOA layer to retrieve and manipulate process data located in various locations. It is mostly concerned with the *Request Database* where we find the complete request data structure that is maintained during the process life-cycle. One example of a functional fall out is a planned review of the applying customers credit rating results. This process might legally require that more than one person must review the customer's rating results before approving them as new bank customers.

Rule engine. This is a non-FOSS component supplied by a third party which we access from STP with BPM projects for business rules. This allows the

business entity to maintain their own rule set regarding their businesses unit within the financial organization. For example, within a savings product you will have various rules and regulations as to the various conditions that must be met before a customer can be allowed to purchase that specific product. These rules and regulations can change over time or due to a special offer on that product during a specific time frame. It is often a wish from the contracting business unit to be able to manipulate these rules and regulations without having to contact the software vendor (i.e. project team).

JBOSS: jBPM and Service Layer (SOA). The application server is an open source component called JBOSS [11], from the JBOSS component family we have adopted the jBPM engine [12] and its process definition language (PDL) implemented in jPDL [13]. These are the main FOSS components in our project solution and are considered core components in the enterprise architecture.

The jBPM process engine is used for all BPM projects, so component selection was not an issue. The BPM process flows are defined by the information analyst together with the business customer for the application. It is a process involving workshops and use cases. It provides the lead developer of the project with a starting point, in the form of a process flow. This is mapped almost one to one into the process definition language which delivers a jPDL file. The resulting process definition is used for matching nodes to business services. In most cases this is a one to one mapping and the design of the services is the most time consuming part of the implementation. Should there be any technical details that call for adjustment to the flow, consultation ensues with the information analyst, and eventually with the business customer. Individual developers are then given technical designs based on use case realizations that allow them to integrate their implementations into the proper process steps.

The project was completed using only simple nodes that contain all business logic in plain Java. Basic service calls were combined in the Java code to achieve what later could be implemented as a complexer business service. There were no nodes implemented as actual wait states, where the process can wait for action from an external system. Our backend systems are not yet setup to trigger jBPM process instances to allow for real wait states. To facilitate wait states, a polling mechanism was used at points in the process were external systems needed to be checked for completion of a task. For example, while waiting for a customer to correctly identify themselves by returning a signed contract with a copy of a valid identification, the process will use a scheduler to periodically poll the backend system via a web service to determine if the identification has been completed. Once completion is detected, the scheduler triggers the process via a web service. Furthermore, there are the standard decision nodes, transitions, and human task nodes within the project's process implementation.

We have implemented a standard Service Oriented Architecture (SOA) [14], referred to in-house as our Service Oriented Architecture Layer (SOAL). Granularities of the services in this layer have been defined as basic services, business services, and some very simple composite business services (CBS) [15]. A basic service brings the existing transaction out of the backend system and makes it

available through a web service. For example, to validate a postcode, the basic service *postcodeCheck* has been created to expose the backend mainframe transaction that checks if a given postcode is valid. The business services concern complex processing that may consist of one or more basic services. One of the more complicated issues is that of allowing the existence of CBSs in our SOA layer. These are business services that can contain not only calls to basic services, but to other business services as well, if the business service being called is in the same classification category as the caller.

The SOA layer deploys web services with versions. If a new release of the SOA layer contains services with interface changes, then the version of the release will be increased. To support backwards compatibility, a total of three versions is maintained for production applications to use. This allows for applications to upgrade to the newer versions over time.

Back-end systems. These systems can be anything in the wide variety that exist within our banking infrastructure: banking applications that provide and interface, external third party services, legacy systems, or some form of data storage like a data warehousing solution. It should be noted that these systems are always approached from our projects via the SOA layer in the form of a web service. We will provide the three most important backend systems that are used in STP Purchasing.

A *request database* was implemented for tracking each purchasing request as it migrated through the BPM process flow. This was the direct implementation of our purchasing request data model. As stated in context of the *web interface* and *human task* components, this database is filled with the initial request data, manipulated by the process as it migrates through the various steps, and directly affected when technical or functional fall out occurs. Access is arranged by a very specific service, dedicated to accessing, reporting, and updating data in the database. I works for the web interface, the human task interface, and from inside the process itself.

Another important component in the backend is the *customer information system*, used to maintain all customer and prospect contact information. This is a marketing data pool and there is a specific service dedicated to accessing and updating the information kept here.

A central system in our backend network is a legacy COBOL mainframe. This is is where the bank customers are managed and it is accessed via web services that make use of a Java communication layer. This layer bridges the gap between Java and COBOL mainframe functions which are provided when functionality is exposed from the mainframe.

3.3   Customer Testing

From the very beginning of the project, customer input was sought. An initial prototype was created, four customers and four internal customer support personnel were invited to conduct usability testing in a controlled environment. These 8 sessions where 90 minutes long, each dealing with a single respondent

and a task assignment walk through. The walk through was done by the respondent with verbal communication accompanying all actions which were recorded by an observer sitting in a different room with a hidden view.

Even though it was a small usability test, it did provide relevant details which lead to advice for the development team in the areas of information structure, interaction, navigation, content, graphical information, style, layout, and features. Our view is that any steps taken to improve the customer satisfaction should be exploited to the fullest.

Another point in customer testing occurred before the project was released into production. It is was a last test that the business users took to test out the entire project. The testing users were guided by a test leader during the earlier project iterations to develop functional stories. These were then set up in the databases to allow them to test actions on submitting new requests, handling functional fall out, schedulers, and other such actions as they deemed necessary for project acceptance. This is a standard practice in our project release cycles and it remains a valuable feedback loop for finding functional problems before the project hits production status.

3.4   The Running Process

Empirical data providing results concerning running STP Purchasing in production since February 2008 will be presented below in an overview. The numbers represent the total number of processes per month, with a rather large spike in the months starting in September 2008. This was the beginning of the world wide Financial Crisis, which lead many Dutch citizens to spread their savings to different financial institutions.

Taking a look at Table 1 we can clarify some of the dips and peaks in the numbers. In February 2008 the project was released half way through the month, resulting in a low start number. It picked up steam and was pretty steady until August 2008, which we believe is due to the vacation period when most Dutch people tend to be on their holidays and away from computers. In September we

Table 1. Production process overview - 2008 monthly

Month	Requests
Feb	750
Mar	2750
Apr	2000
May	1200
Jun	1100
Jul	1500
Aug	850
Sep	4250
Oct	2250
Nov	1000
Dec	1800

Table 2. Status overview of customer processes

Status	Percentage
Completed on time	52%
Rejected for various reasons	8%
Human action (functional)	0.7%
Human action (technical)	0.3%
Currently in a fall out status	4%
In Document Monitoring	12%
Taken out of STP flow, completed by hand	23%

see the explosion of interest due to the Financial Crisis, followed by a leveling of interest. At the end of November 2008 the second set of 5 *deposito* products hit production. Unfortunately, we cannot say much about their impact as the December 2008 results are again of a partial month.

Another view of results is given in Table 2, which shows us percentages of the various statuses a process can be in. We must take into consideration that our metrics are limited and that we are only able to report on the process totals. Even so, it is encouraging that the amount of functional and technical fall out that needs attention are both less than 1% of the total. Also encouraging is that over 50% of all processes are completing on time. The ones that do not complete on time and are listed in *Document Monitoring* tend to be waiting for customer reactions to documentation problems as previously discussed. We have a timer running that ensures a customer receives reminders several times. Should the customer not reply at all, we eventually abort the request. The category listing 23% of processes taken out of the engine and completed by hand needs more explanation. This feature was added to allow special cases to be handled in the original manner, by hand.

With only 8% being rejected due to various reasons, it appears we are hitting the target audience and providing a process that is effective.

4   Observations

Not everything is as pretty as it seems, with issues remaining for both the business and technical sides of the playing field. We take a closer look at these, starting with the business challenges.

4.1   Business Challenges

Even though the use of business process models is proving itself successful at SNS Bank, there is room for improvement concerning the activities of conceptualization, communication, and engineering that are part of the ongoing development process.

Quality of business process models is a notion that has many aspects and thus is quite complex [16]. Engineering-oriented, mathematics-based aspects are

involved (correctness, formal expressiveness, and various more specialized aspects such as mentioned by Vanderfeesten et al. [17]), but also social aspects (validation, agreement through collaboration, and common understanding [18]). For high quality process models to be realized, sufficient investment in detailed knowledge exchange and discussion is required.

The main challenge is a common one: the business (analysts, managers) have the best knowledge of detailed processes to be supported/automated, and have the authority to decide about them, yet are neither willing nor able to be too actively and intensively involved in high-quality, detailed, engineering-like specification of business processes, which they consider a "technical" job. Technicians, on the other hand, resent being forced to guess at details required for successful implementation and point out that "technical" is not the same as "involving detailed, precise, and well-conceived descriptions". This is mostly a cultural issue [18], but therefore also a deeply rooted one.

In an ideal situation, we would still need the people with the proper knowledge and authority concerning the business to describe and/or design products and processes. How this can be realistically achieved in the short run is an open question yet. Possible options include:

- Teach business people to read (at least) and create (less likely) formal modeling techniques.
- Find and use alternative ways to represent formal process models; verbalization, perhaps, or alternative (simpler) schemes.
- Encourage and allow business-oriented stakeholders to get involved in more detailed process modeling.
- Arrange for discussion and negotiation about process models to be optimally collaborative from the start, i.e. involve all relevant stakeholders at an early point and create explicit agreement (e.g. in workshops). The more divergence occurs in this phase, the more the diverging parties will fight for the survival of their initial ideas later on, and the harder it will be to reconcile alternative models. Discussion should take place, certainly, but not because effort and authority has been invested in particular diverging positions.

4.2   Technical Challenges

There are some interesting technical challenges that need to be watched for future projects. They cover issues concerning BPM, business logic, and (business) service releases. A currently completing BPM reference implementation [19] project has taken a closer look at these challenges and has come up with a few solutions and suggested ways of dealing with them.

Starting with the BPM issues, we have spent much effort to move the business logic out of the BPM process engine and down into the architecture to the SOA layer. This keeps the BPM engine lean and mean, requiring a lot less testing during the deployment phases of a project. Once the BPM flow is working, tests are passing, handlers call the correct services, and the infrastructure to support all of this is available, then there is not really much looking back. The main focus

in searching out application problems are contained in the SOA layer. Developers spend their time testing and maintaining the business logic in the services, where it belongs. The delivered BPM flow should be almost maintenance free.

Many of the problems that the developers encountered with BPM process definition designs as described by Brahe [3] were avoided in our process by keeping the process flow definition, creation, and modification out of the hands of the developers. Modeling took place at a higher level, with a smaller group containing information analysts, business representatives, and the lead developer. This process led to a completed BPM process definition, in the process definition language, but expression in that language happend only at the end of the modeling process. We would like to look more into *directly* generating actual BPM process designs close to the chosen process definition language, *together with the business*. This is something to be further examined in the future.

Individual developers were able to concentrate more on working out the individual process steps (nodes and handlers), the given initial business service designs, test coverage, and documentation. This has worked well for us and we will continue to use this approach in the future.

Although there has been some literature on the use of SOA [20,21], we have found that most of the issues discussed where of little help when dealing with our own service construction. It seems that issues are often related to local conditions and infrastructure limitations. One complex issue arose in our environment, the issue of unreliable services due to all web service calls being implemented over the HTTP protocol [22]. The problem is more complicated when the basic services, themselves mapping to single backend transactions, are unreliable. It is conceivable that a service call is made to some complex business service that makes use of several basic services, and that it fails somewhere in the processes of executing basic services. We have no ability to implement anything other than a functional rollback and often are not sure what state the backend systems are left in.

There are potential problems with any service releases in the SOA layer that migrate to a major version number. For example, all minor version number releases from v1.0 to v1.1 of a given service contain no interface changes. These are therefore backwards compatible and should continue to work with all previously written consumers of the service. For major version changes, such as v1.1 to v2.0, we are confronted with a service containing an interface change that might break existing consumers of that service.

Service granularity has started to become a problem with more and more projects attempting to make use of basic, business, and composit business services that they find in the SOA layer. We hope to spend more time on looking into composite business service issues and lay some ground work with regards to guidelines for future projects.

A very sticky problem that has raised its ugly head is what to do with BPM process instances that are running when the new service release is planned. We are looking at our options at this time but have come up with the following strategy to provide a choice depending on the given situation:

1. Phase out older service versions when all old process instances have completed.
2. Build service converters that translate calls between different versions.
3. Activate a new BPM process instance for each existing old process instance.
4. Build a process converter that translates old processes into the new process definition (one time).
5. Human interaction to guide the process or complete the process flow.

This is an integral part of our current SOA service release strategy and can be found in the internal SOA documentation.

A solution is currently being tested that provides a custom class loader for each individual jBPM process engine. This allows each deployed process definition to provide the exact service version for each service it uses. This allows different deployed processes to access any of the SOA layer deployed service versions, independent of each other. This will have a positive effect on testing phases when multiple processes can be deployed on a single jBPM process engine, thereby saving extra hardware resources. This solution will also allow older instances of a process to be run next newer ones so that they can be phased out as mentioned above.

All contact from the process with internal systems is realized via web services. These calls are synchronous, but many of the backend systems are not. Many run batches which means that the web services provide transactions to functionality that can only report that the request has been received correctly. For example, a fictitious account is opened via a web service call, but this actually happens in a night batch run on the backend mainframe. The web service call will get the mainframe reply, *Account Opened*, but this process will not be actually completed until later. This indirectly means that web services can not be transactional or atomic in nature and a great effort is made in business service implementations to create as much of a functional roll back as can be achieved. More often than not, it means having to fall out of the process with a technical problem to be fixed by human hands.

At the time of this writing, a *state-proxy* is being implemented to allow for real wait states in the process definitions. When using a wait state, the business service call is done through our state-proxy. The process is then put into a wait state and the proxy handles the web service call, returning either an exception or the results. This state-proxy can then be expanded with extra plug-in like functions, such as dealing with service windows for known down time on backend systems running a batch, allowing for technical retires to services that can be offline for short periods of time, and dealing with standard exceptions. These plug-ins are on the drawing board for future implementation.

The scheduler discussed in section 3.2 is a point of concern. This does not scale well and in the future we will need to look into getting our backend systems to trigger on certain events. This should be possible and the discussion is underway.

Another nice-to-have would be to remove the non-FOSS rule engine discussed in section 3.2. We want to spend some time looking into the JBOSS rule engine in the coming year which seems to provide a solution that is integrated in our existing development tooling.

5   The Benefits

As we have seen, the benefits of BPM are promising, based on the empirical data collected in the deployed production process. A closer look at the customer and development benefits will make it clear that there is much gained already.

5.1   Improving the Customer Experience

A key concept in the vision of this solution is that the customer must be central to the process. A customer centric business model is not new [9], but we feel that aligning the entire strategy to empower one's customers is breaking the mold.

As strategic products are made available through full scale STP with BPM we are able to adjust easily to customer needs. Products and product lines can be introduced into existing business processes in a cost effective manner. The flexibility to combine extends beyond products, product lines, and selling channels to become a very effective tool to reach customer bases in a timely and personalized fashion.

Customer communication can be personalized and tailored to specific processes, products, and customers' personal needs as the data generated by their behavior within the processes is documented. There have been very positive reactions from customers with regards to the speed, quality, and the level of detail in communications.

5.2   Development Process Improvements

The initial STP Purchasing project has provided a starting point for the IT department to build on for future full scale STP with BPM projects. Lessons learned and best practices are being applied, resulting in some interesting improvements to the process.

To our initial surprise, BPM process definitions can be easily changed with a minimal impact on the development time. The work is not in the process definition, but in the business services and basic services in the underlying structure. A standard way of implementing process nodes and testing has made this part of the development process much less critical. It is important to focus on what we call the *Happy Flow* during initial development. This is the backbone of the process flow which represents a positive test case that processes as expected. For example, we would focus in the STP Purchasing project on a single saving product being requested by a verified and known customer of the bank. This means that you do not have to deal with any exceptions during the initial run through your process implementation. The focus of the first iteration of development is to get this *Happy Flow* working. By providing a quick working Happy Flow, the business can be shown tangible progress in the project at an early stage.

With an ever growing base of BPM process definitions it is clear that the time to market for similar products is much quicker. We have projects with estimates ranging from one third to one half of the initial development hours put into STP Purchasing. This is quite a big improvement. One thing of note here would be

that the development of business services should always be carefully considered, as they tend to be the focus point of complexity.

The initial process definitions as provided by the information analysts and business analysts are not in our process definition language. Much depends on the quality of this process flow model, but with some care and attention to this step it is not too much trouble to map this process flow model to our process definition language. The generated image of the flow is a very good communication tool with the business. No better way to let them see the business services and understand where the development time is spent. Bringing the business closer to the development team with regards to communication over the process flow has been a positive experience that we would like to see continued.

6   Future Plans

In the coming year(s) we will be tackling the projects described below. They are a clear sign that the success that has been achieved with previous full scale STP with BPM projects is relevant. SNS bank is moving ahead full steam with projects for certificates of deposit, debit accounts, and the migration of existing internal and external service processes.

Since STP Purchasing completed in the beginning of 2008, work has started on expanding the product category with five new *deposito* products. This project went live in November of 2008, just in time to provide the Dutch market with an easy way to spread their saving money around between banks: note the peaking numbers in the empirical data that reflect the public reaction the the start of the Financial Crisis in September 2008.

6.1   STP Payments

This project will take place in 2009, with plans to focus on debit accounts for standard customers, children, and students. The solution will need to implement the following business processes for these accounts:

- requesting and receiving a bank pas for the account
- requesting and receiving internet access for the account
- requesting and updating the credit limit for the account
- requesting and receiving a new credit card

6.2   Migration Service Processes

A service process migration project is looking at migrating 169 internal processes to make full use of BPM and STP. These are processes that internal SNS Bank employees make use of to process various customer needs. All of these processes will make use of our full scale STP with BPM solutions as much as possible. In 2008 one of these service processes was put into production: a process to allow customers to submit a name or address change through full scale STP with BPM.

A list of items that have caught our attention and imagination are provided here without further discussion. These are possible points for exploration that might provide added value to the process of deploying full scale STP with BPM in the future.

- enable business customer to be more involved with "business engineering"
- better motivate business customer to be involved with "business engineering"
- enable business customer / information analyst to deliver better PDL models as input to the development process
- enable migration of existing process definitions to new process definition releases (results of reference project)

7   Moving Ahead

In this paper we presented the efforts of a Dutch bank at migrating from traditional banking services to a full scale STP with BPM driven financial institution. The components being used to realize the STP Purchasing project were described and the resulting empirical data were presented for evaluation. The issues and benefits were covered along with the challenges yet faced by both business and IT development. The large shift in strategy has started to deliver the desired results and these will continue to roll in as future full scale STP with BPM projects are implemented.

The positive effects on customer interaction, improvements on accelerating product deployment, and more flexible product/customer support channels have energized some internal ideas about becoming a facilitator to external third party enterprises. Imagine a future where individual entrepreneurs would be able to open a banking store with complete full scale STP with BPM selling channels for products and services.

We hope that our experiences, lessons, and observations will be of value to the industry as a whole. This is a financial industry story, but it could be applied to many different situations and time taken to learn from this story would be well spent.

References

1. Khanna, A.: Straight Through Processing For Financial Services: The Complete Guide. Academic Press, Burlington (2007)
2. van der Aalst, W.M.P., ter 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. Brahe, S.: BPM on Top of SOA: Experiences from the Financial Industry. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 96–111. Springer, Heidelberg (2007)
4. Guerra, A.: Bloomberg Aims To Simplify Straight-Through Processing. On: InformationWeek (December 18, 2000),
   http://www.informationweek.com/817/bloomberg.htm

5. Strickland, R., Aach, D.: Getting to straight-through processing: in theory, there is a way to deliver faster and better service in the mortgage lending business. On: BNet Business Network (February 2006),
   `http://findarticles.com/p/articles/mi_hb5246/is_/ai_n29277448`
6. The Free Dictionary (February 10, 2009),
   `http://encyclopedia2.thefreedictionary.com/Straight+Through+Processing`
7. Answers.com (February 10, 2009),
   `http://www.answers.com/topic/straight-through-processing`
8. Investopedia (February 10, 2009),
   `http://www.investopedia.com/terms/s/straightthroughprocessing.asp`
9. Heckl, D., Moormann, J.: Matching Customer Process with Business Processes of Banks: The Example of Small and Medium-Sized Enterprises as Bank Customers. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 112–124. Springer, Heidelberg (2007)
10. Java Technology (March 18, 2008),
    `http://java.sun.com` (March 19, 2008)
11. Jboss.org: Community Driven (March 19, 2008),
    `http://labs.jboss.com`
12. jBPM Overview (March 19, 2008),
    `http://labs.jboss.com/jbossjbpm/jbpm_overview`
13. Welcome to jBPM jPDL (March 19, 2008),
    `http://labs.jboss.com/jbossjbpm/jpdl`
14. Erl, T.: Service Oriented Architecture: Concepts, Technology and Design. Prentice-Hall, Englewood Cliffs (2005)
15. Neuman, S.: Composite Business Services. IBM Global Business Services (October 25, 2008),
    `http://www-935.ibm.com/services/us/index.wss/offering/gbs/a1027243`
16. Recker, J.: Towards an Understanding of Process Model Quality. Methodological Considerations. In: Ljungberg, J., Andersson, M. (eds.) Proceedings 14th European Conference on Information Systems, Goeteborg, Sweden (2006)
17. Vanderfeesten, I.T.P., Cardoso, J., Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Quality Metrics for Business Process Models. In: Fischer, L. (ed.) BPM and Workflow handbook 2007, pp. 179–190. Future Strategies Inc., Mississauga (2007)
18. Hoppenbrouwers, S.J.B.A.: Community-based ICT development as a multi-player game. In: Conference proceedings of What is an Organization? Materiality, Agency and Discourse, May 2008, University of Montreal, Canada (2008)
19. Schabell, E., Benckhuizen, J.: Software Architecture Document – jBPM Reference Project. SNS Bank IT, s-Hertogenbosch (2008)
20. Mahajan, R.: SOA and the Enterprise – Lessons from the City. In: IEEE International Conference on Web Services (ICWS 2006), pp. 939–944. IEEE Computer Society, Los Alamitos (2006)
21. 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)
22. HTTP − Hypertext Transfer Protocol (February 27, 2008),
    `http://www.w3.org/Protocols` (March 19, 2008)