Chapter 3
Effectively and Efficiently Implementing Complex Business Processes: A Case Study

Volker Stiehl, Marcus Danei, Juliet Elliott, Matthias Heiler, and Torsten Kerwien

Abstract The implementation of business processes has been neglected for many years in research. It seemed to be that only hard coding was the appropriate solution for business process implementations. As a consequence in classical literature about business process management (BPM), the focus was mainly on the management aspects of BPM, less on aspects regarding an effective and efficient implementation methodology. This has changed significantly since the advent of BPMN 2.0 (Business Process Model and Notation) in early 2011. BPMN is a graphical notation for modeling business processes in an easy to understand manner. Because the BPMN standard had the process execution in mind when it was designed, it allows for a new way of implementing business processes, on which the process-driven approach (PDA) is based. This approach has been applied in a huge project at SAP SE since 2015 comprising more than 200 business-critical processes. In order to get an impression about the power of the process-driven approach for really complex business process implementation scenarios, this chapter explains the basics about the process-driven approach and shares experiences made during the execution of the project.

V. Stiehl (✉)
Faculty of Electrical Engineering and Computer Science, Technische Hochschule Ingolstadt (THI), Ingolstadt, Germany
e-mail: volker.stiehl@thi.de

M. Danei · J. Elliott · M. Heiler
SAP SE, Walldorf, Germany
e-mail: m.danei@sap.com; juliet.elliott@sap.com; matthias.heiler@sap.com

T. Kerwien
itelligence AG, Bielefeld, Germany
e-mail: torsten.kerwien@itelligence.de

D. Lübke, C. Pautasso (eds.), *Empirical Studies on the Development of Executable Business Processes*, https://doi.org/10.1007/978-3-030-17666-2_3

3.1 Introduction

Business process management (BPM) in general has been explored over many years covering a variety of topics such as strategic BPM, process organization, process planning, process controlling, process evaluation, risk management for processes, process performance analysis, process optimization, process mining, and change management when introducing BPM in organizations. These areas are well researched, and many improvements have been achieved for all of these topics over time. However, in the authors' view, one area in this whole process universe seems somewhat neglected by comparison: the model-driven implementation of complex business processes. There have been several standards in the past like BPEL which tried to give answers to the topic. However, due to a range of issues, from missing standardized notations to an almost exclusively technical focus, companies were not able on a large scale to use them to implement software to meet the needs of complex, real-life scenarios. Precisely this implementation of complex business processes is what companies need to do when they want to address differentiating business processes which cannot be covered by standard processes delivered by standard software (e.g., SAP S/4HANA) due to their uniqueness for a company. But what options do we have at our disposal for implementing differentiating business processes? For many years, the only option seemed to be hard coding the processes using an appropriate development environment of choice, e.g., environments based on widespread programming languages such as Java/C#/JavaScript or proprietary environments like SAP's development environment based on the ABAP programming language. Experience has shown though that this approach has some weaknesses. Issues that companies have experienced include the following:

- Development speed and ease of maintenance.
- Making changes is usually cumbersome.
- Transparency in running or finished process instances is not innately given.
- Changes in market conditions can require extensive recoding.

Since differentiating business processes are a key factor in gaining or keeping a competitive advantage, we can see that finding more effective ways to address these issues could be vital to a company's success.

Maintaining that competitive advantage is more crucial than ever before, given the ever-increasing pace of change brought on by the pressure to innovate as a result of global digitalization. If companies miss new trends, they might be out of business very soon. In this dynamic environment, the need to address the challenges arising out of the business/IT alignment problem becomes ever more acute: in most cases, process experts in operating departments work out the to-be business processes using graphical notations such as EPC (Event-Driven Process Chain) or BPMN (Business Process Model and Notation). As part of the software specification, process models are exchanged with the developers who have to implement the new solution based on the process models. Experience at several companies has shown though that changes made to the models during implementation mean the original

models are outdated by the time the implementation is complete and they are rarely updated to reflect the reality of the implementation. This rather limits the usefulness of the models, and consequently they become "shelf-ware." This is highly frustrating considering the effort which has been spent on these models.

In order to gain transparency into running/finished processes, companies often then invest in additional software for process mining and process analysis. Process mining, for example, helps to determine which paths the finished processes followed during their execution. It sounds illogical that additional software is necessary to derive a process model out of the logged data, although originally the processes were implemented using process models.

Experience therefore shows that this approach has its shortcomings, and it is valid to search for alternative approaches which address all or at least some of the mentioned limitations—ideally without introducing new limitations at the same time. With the introduction of BPMN, we now have new options at our disposal, especially with version 2.0 of the BPMN specification because it explicitly contains execution semantics for business process engines which can now execute BPMN-based process models. The approach of developing business processes using a standardized notation and running the models on a process engine would seem to offer some potential for addressing the limitations mentioned. Model-based development is not in itself new, but the development of software based on models is seen critically by experts due to the fact that models get quite complicated and unmanageable when it comes to complex real-life business scenarios. One method of addressing this challenge is introduced in the book Process-Driven Applications with BPMN [4] published in 2014. It introduces a holistic approach for implementing complex real-life business processes based on BPMN models. The presented solution is named the "process-driven approach (PDA)" and describes precisely what needs to be done to successfully implement differentiating core business processes. The process-driven approach comprises the following:

- A collaboration model between business and IT called "BizDevs" to overcome the business/IT alignment problem (see Sect. 3.2.2)
- A new way of thinking about BPMN-based business process implementations (process-driven thinking; see Sect. 3.2.3)
- A new methodology for business process implementation projects (process-driven methodology; see Sect. 3.2.4)
- A specific software architecture recommendation for process-driven applications (process-driven architecture) and a suggested development approach (process-driven development; see Sect. 3.2.5)
- A recommendation for a technology stack which supports process-driven applications best (process-driven technology; see Sect. 3.2.6)

The feasibility of the approach in theory was proven in the book itself. Using a small example, the basic architectural and development details were explained. However, what remained open was the applicability of the approach for real-life scenarios.

The arguments in favor of the process-driven approach led SAP Language Services to decide to use this approach for a major project to fulfill their core business requirement: providing services for translations (in 40 languages) of a variety of items for SAP products, such as user interfaces, business reports, marketing materials, videos, and handbooks. The requirements for running the business processes supporting these services were so unique that no standard off-the-shelf translation management software could meet them. So the SAP Language Services team decided, after an intensive evaluation phase, to build their differentiating business processes following the PDA methodology. As part of this chapter, we will describe the project and will summarize the experiences made with the process-driven approach applied to a real-life scenario. It addresses in particular the following questions:

- Is it possible to use the model-based approach to build applications that fulfill complex, real-life business needs? What needs to be done to achieve this goal?
- Is it possible to preserve BPMN process models developed in operating departments following the BizDevs collaboration model during implementation?
- One of the main attributes of PDA is the separation of business process and technical artifacts. Is it possible to keep the obvious technical and business complexities under control if the process-driven approach is applied? What needs to be considered?
- Which benefits do companies gain by applying the process-driven approach, and which of the aforementioned shortcomings are being addressed by it?
- Finally, how does the BizDevs collaboration model contribute to overcoming the business/IT alignment problem?

The remainder of this chapter is structured as follows. In Sect. 3.2, we explain briefly the ideas behind the process-driven approach. Section 3.3 describes the project at SAP SE in more detail, and Sect. 3.4 summarizes the results for researchers as well as for practitioners and gives an outlook on further research topics.

3.2 The Process-Driven Approach

The spark for the process-driven approach came from the release of the BPMN 2.0 specification in January 2011[2]. For the first time, execution semantics were defined for a graphical process notation by a standards organization (OMG—Object Management Group). This created a clear definition of how a process should behave if executed by a process engine that complied with the BPMN 2.0 standard. Software vendors immediately started implementing the new process modeling standard, providing process engines that executed BPMN process diagrams. This was a big step forward and laid the foundation on which process-driven applications could prosper. The question was: How can this idea of running BPMN-based models using an engine be transferred into real-life projects? The research carried out for the book

on process-driven applications established that several aspects are required, which must work hand in hand for it to be successful. Although it is impossible to repeat all the details of the process-driven approach described in the book, in the forthcoming paragraphs, we will summarize the main ideas. More details can be found in [4].

3.2.1 Definition of a Process-Driven Application

The definition of a process-driven application is as follows [4, p.19]:

Process-driven applications are business-oriented applications that support differentiating end-to-end business processes spanning functional, system, and organizational boundaries by reusing data and functionality from platforms and applications.

The definition stresses already the importance of business requirements and process logic as the main driver for all decisions that need to be made while developing the application. The process-driven application is the result of applying the process-driven approach. If we take a closer look at business processes, we can distinguish between standard business processes and unique, company-specific, differentiating business processes. Standard processes are well covered by standard products, and it doesn't make too much sense for companies to implement these themselves. However, companies cannot do much to differentiate themselves from the competition by using standard processes, so the next question we have to answer is this: How can companies quickly and sustainably build, run, and monitor differentiating business processes? This is exactly where the process-driven approach fits into the picture by providing an effective and efficient implementation methodology. Key criteria for a process-driven application are independence from the IT landscape and process flexibility in regard to changing market conditions and competition. These criteria will be mainly supported by the process-driven architecture which will be discussed in Sect. 3.2.5.

3.2.2 Process-Driven Collaboration (BizDevs)

The main idea behind process-driven collaboration is overcoming the alignment problem between business and IT, with both sharing common responsibility for *one* BPMN model right from the beginning of a project. The traditional development process was very much dictated by business folks handing over software specifications which had to be implemented by their IT colleagues. Because of the potential misunderstandings caused by software specifications formulated using mainly prose, the results of the implementations rarely fulfilled the original requirements immediately. The typical ping-pong game between business and IT started, consisting of implementation (by developers) and review phases (by

business colleagues) until the final result was eventually reached. This "procedure" is time-consuming, error prone, and highly frustrating for both parties.

The process-driven approach targets those shortcomings, changing the collaboration between business and IT by stipulating that a well-defined notation (BPMN) must be used to depict the process logic precisely. In addition, the modeling of the business processes is done together right from the start of an implementation project. Both sides enter into a partnership of equals. Because both sides work together on one BPMN model, chances are very high that the implementation immediately fits the expectations, and that increases development productivity. This raises the question of whether a roundtrip of one BPMN model between the business and IT teams is possible or not. However, BPMN as the common language between business and IT allows work on new levels. The new collaboration model is based on collaborative work on *one* BPMN model, which is then executed, as it is, by a BPMN engine. The BizDevs collaboration model simply does not permit changes to the BPMN process model just to make it executable. Although this may sound challenging to achieve in practice, the goal can be reached if organizations are willing to follow the new collaboration model, where both sides are equally responsible for one BPMN model and where the focus is on the preservation of this model throughout the transition to execution. This preservation of one BPMN model is also supported by the process-driven architecture which will be discussed in more detail in Sect. 3.2.5. The responsibility for the executed processes is now extended to the business side, so there can be no more finger-pointing between the two camps. For this new kind of collaboration, the term BizDevs has been coined to describe the collaboration between business and development. The term is influenced by the term "DevOps," which describes the collaboration between development and operations.

BizDevs means that business people become an integral part of the process development cycle—a new accountability that the business folks have to get used to. In addition to defining how the process should ideally run, it is also important from the start to define exceptions—what should happen if an expected outcome is not reached. For example, if it is critical that a process participant responds within a certain time frame, what should happen if they fail to do so? Another example could be a technical error that prevents the process from moving on. Here again, the value of business-IT collaboration becomes obvious. Without this collaboration model, the implementation of BPMN-based process models becomes questionable at best. Hence, BizDevs is an indispensable prerequisite for successful PDA implementations.

3.2.3 Process-Driven Thinking

BPMN is not just another modeling notation for business processes. Unfortunately, many authors of books about business process management see it this way: they only describe BPMN alongside other modeling notations, reducing the comparison between them to just the different shapes supported by the notations. Process-driven

thinking uses the full shape set of the BPMN palette. For a thorough understanding of BPMN, it is crucial to consider the semantics of all shapes in the palette in order to apply them correctly in process models that can then be correctly interpreted by BPMN process engines at runtime. BPMN process engines in the end implement the semantics described in the BPMN specification. So thinking in "process engines" is a new challenge for modelers, business people, and developers, who have to design for execution right from the start. Process models need a new level of precision as engines require detailed information to make models executable. Because of this precision, there is no room for misunderstandings or ambiguities left. To increase this level of precision, modeling guidelines such as the ones described in Bruce Silver's book [3] are highly recommended. Together, modeling guidelines and the awareness that process engines rely on precise process models result in high-quality process models which can be understood from the diagrams alone.

Another aspect of process-driven thinking puts the business processes in the center of gravity. Every decision to be made during a project's lifetime always asks for the business requirements first. It is also at the heart of our next section, the process-driven methodology.

3.2.4 Process-Driven Methodology

Because of the importance of the business processes, one central question is: How should a process-driven project be started? Should we start with an actual analysis of the current (process) situation and derive the to-be processes from there (bottom-up)? Or should we start with the new to-be processes right away, without considering the current situation at all (top-down)? The answer for the process-driven approach is pretty clear: it's the second option. The problem with the bottom-up approach is the following: you will most probably spend a lot of time and money on the documentation of processes that you already know don't work satisfactorily and for very little benefit. If you try to improve the current process, you are working on symptoms, not on an overall process improvement that takes advantage of the latest technology options. Starting with the to-be processes gives you the freedom to innovate, both in terms of the business logic itself and of harnessing technical innovations.

One core rule of the process-driven methodology is not to let yourself be restricted by the current process implementation or by other technical or organizational constraints, such as the existing IT landscape, external systems, partners, suppliers or customers. The key question for decision-making in the process-driven methodology is always: What does the business logic require? From this point of view, it is possible to derive the business objects (e.g., a purchase order, an account, an employee) and their properties, the required services and their interfaces, user interfaces, decision rules, events, process steps, etc.—everything that is necessary to make a business process model executable. Applying the process-driven methodology sounds easy at first, but is sometimes hard to follow

because people tend to always think about their IT landscapes and the restrictions they imply. The clear recommendation is not to think too much about IT landscapes and systems because they are changing anyway, especially in times like these where the trend to cloud-based systems is increasing, where mergers and acquisitions happen, all contributing to an even more fragmented IT landscape. You simply cannot afford to depend on such a brittle foundation. It is better to abstract from specific systems and stay independent from them. Following this approach allows much shorter time to market cycles from concept to implementation. Remember that it is one of the major goals of a process-driven application to be as independent as possible from a company's IT landscape, and the process-driven methodology contributes to that goal. It is further strengthened by the process-driven architecture which will be discussed next.

3.2.5 Process-Driven Architecture and Process-Driven Development

In order to fulfill the promises of independence and flexibility for the process-driven application as described in Sects. 3.2.2 and 3.2.3 as well as the promise of preserving a BPMN model throughout the transition from the original model to execution, an architectural blueprint is required: the process-driven architecture. An architectural blueprint is needed because the usage of BPMN alone neither ensures a successful development project nor a sophisticated architecture for the resulting applications. The problems known from normal programming apply for BPMN-based developments as well and can best be explained using an example which is taken from [4, pp. 67–74]. Compare Fig. 3.1, the result of the traditional approach, with Fig. 3.2 which uses the process-driven architecture.

You can see from the model how the core processes that are so critical to the success of the company (the upper process in Fig. 3.2) are not obscured by the technical details because these are put into a separate layer: the service contract implementation layer (SCIL). The valuable business process stays intact and, most importantly, remains under the control of the business department. However, the process can be easily adapted for use in other regions; you simply need to adapt the service contract implementation layer (the lower BPMN model in Fig. 3.2). Of course, this adaptation does involve some effort, but applying this approach will be of benefit in the long term, as it releases you from the complex web of connections between back-end systems. Business processes (e.g., the upper BPMN model in Fig. 3.2) and technical processes (e.g., the lower BPMN model in Fig. 3.2) can be developed and modified independently, but remain connected by the service contract (e.g., the message flow between the two BPMN models). This architecture also helps you to keep the business processes in their original form as conceived by the business departments. The key question in an implementation project will be to determine exactly which activities belong in which layer, in other words, which

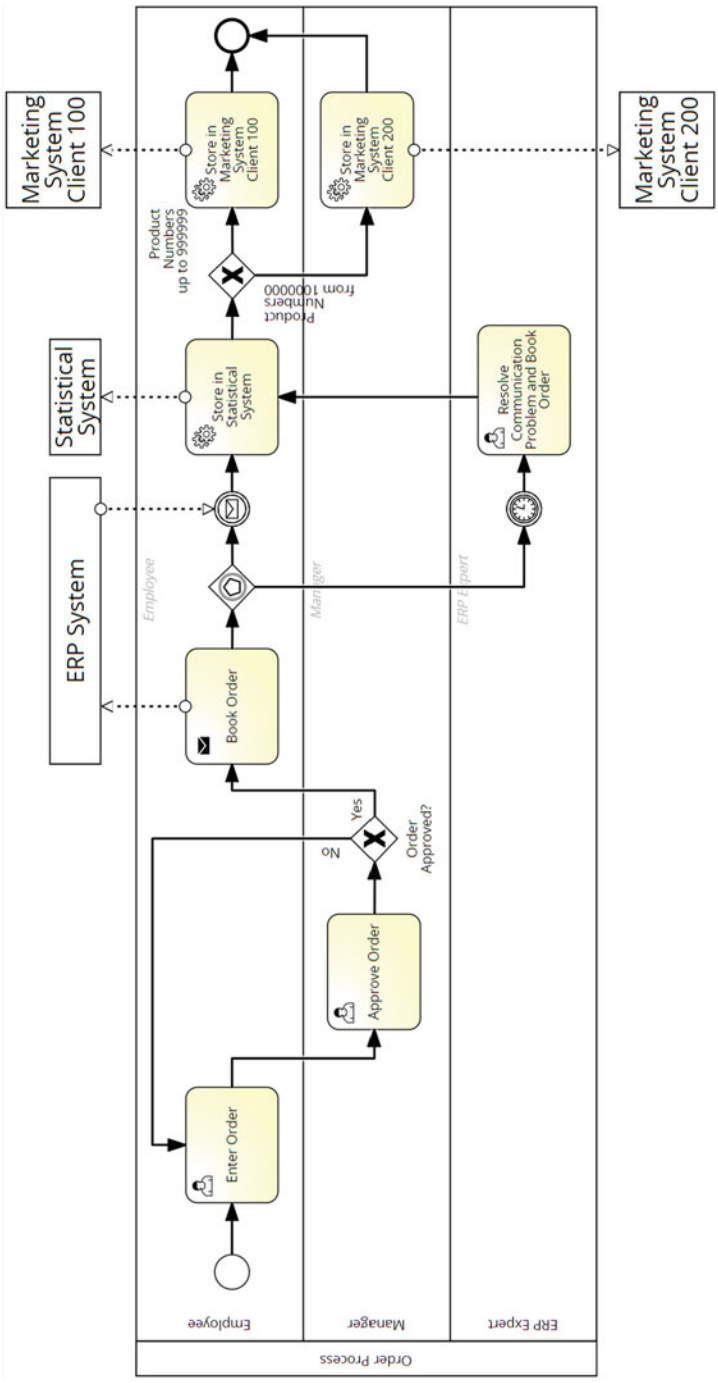

Fig. 3.1 Order process after processing by developer

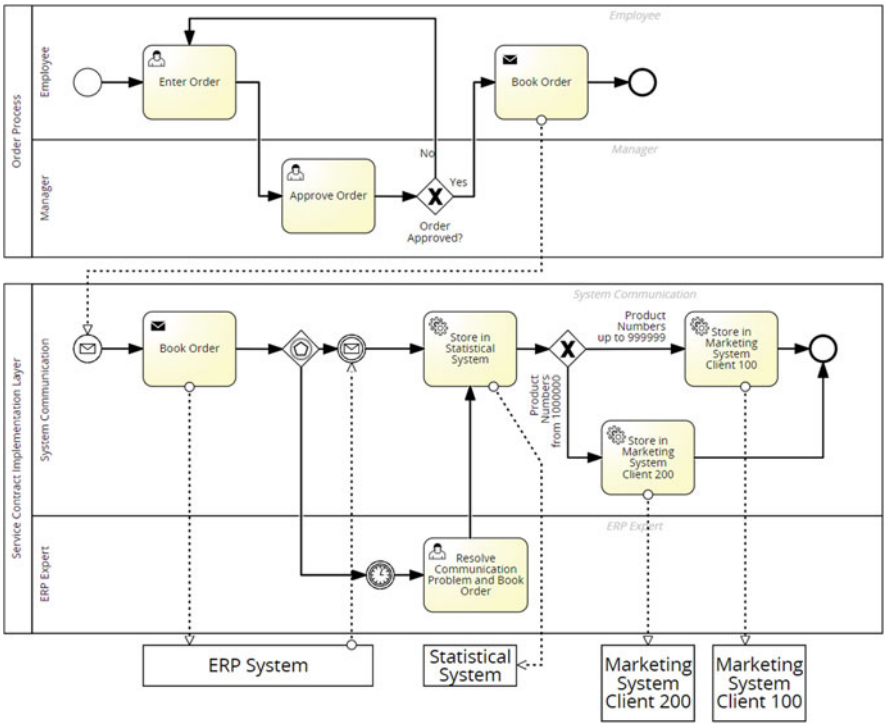

Fig. 3.2 Order process after separation of layers

activities are really part of the core differentiating process and which are supporting activities. Essentially, this determination will be made each time through business-IT collaboration, with the business side in the lead. Key criteria are as follows:

- Does the business see this activity as critical to the business process? Is it necessary, and does it add value from a business perspective?
- Can process participants easily understand the activity?

The basic idea presented above was refined, and this resulted in the reference architecture for process-driven applications depicted in Fig. 3.3.

The PDA layer comprises the business processes and everything needed to make the processes executable, e.g., local persistency for the business objects the processes work on, the user interfaces for BPMN user tasks, business rules for BPMN business rules tasks, events, etc.

Communication with the outside world is handled by the service contract layer. The interfaces described there (the fields and the data types needed for the technical implementation) just consider the needs from the view of the business processes and are defined in both directions: from the business processes to the external world and vice versa. The data types being used for the interface's descriptions

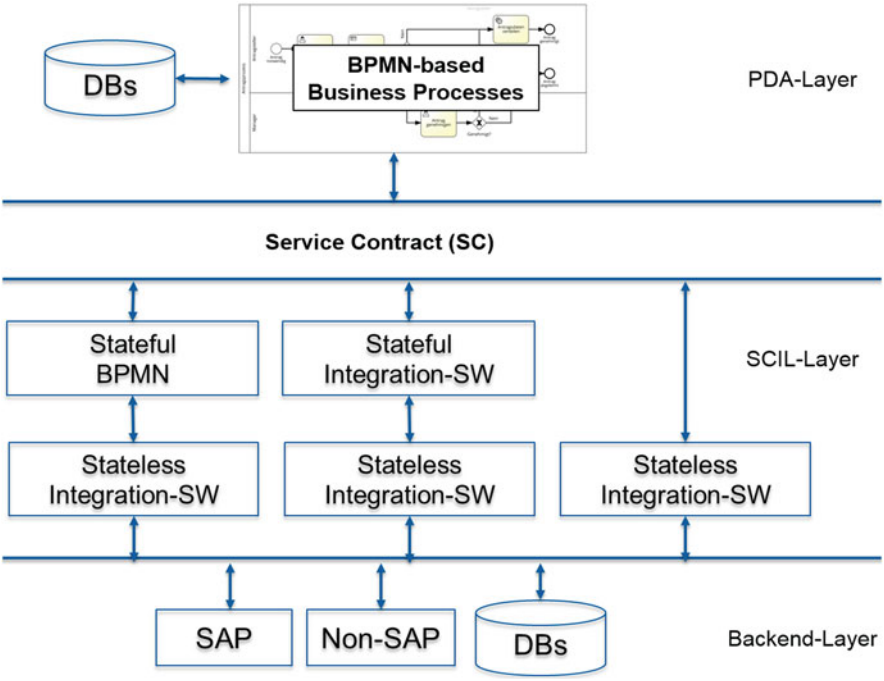

Fig. 3.3 Reference architecture for process-driven applications

are identical to the ones being used within the business process itself, avoiding mappings between different data types. The service contract layer is an abstraction from the specific back-end systems and shields the process-driven application from the IT landscape with its proprietary data types, interfaces, and technologies. A process-driven application never connects to one of the back-end systems directly. This is a typical pitfall in many BPMN models. They contain direct connections to back-end systems, and therefore a change in the system landscape makes a change in the process models necessary. The abstraction of the business model from the IT landscape is no longer given and makes adapting the model to new requirements unnecessarily complicated.

The actual implementation for each service contract is summarized in the service contract implementation layer (SCIL) which, for sure, looks different for each IT landscape the business processes should run on. It takes over the integration part of a process-driven application. As can been seen from Fig. 3.3, the SCIL differentiates between stateful and stateless integration. Stateful integration means the handling of wait states during integration. This is, for example, the case if an aggregation of several messages is necessary before finally sending the collection to a target system, e.g., a combined bank transfer. Stateful integration still relies on a harmonized data type system as it is used in the business process and the service

contract. However, stateful integration is not necessary for every service requirement from the business process. That's why a third option is shown on the right of Fig. 3.3.

The individual data type systems being used in the diverse applications are only relevant when connecting to specific back-end systems. Hence, mapping between the harmonized data type system being used so far and the proprietary data type systems being used in the back-end systems is only necessary in the stateless integration part of the SCIL. Therefore, routing and mapping are the main tasks of the SCIL's stateless integration part. The SCIL layer in Fig. 3.3 depicts three implementation alternatives for the integration:

1. On the left: Stateful integration is handled by a BPMN process, and the stateless integration is covered by specialized integration software. More and more companies are using BPMN for integration purposes as well, especially for stateful integration which can nicely be modeled using BPMN. Transparency is again the key argument in favor of using BPMN for stateful integrations because the BPMN process engines collect all data needed for monitoring the integration processes during runtime. This significantly simplifies operations. However, the usage of BPMN for stateful integrations is only recommended if the engines fulfill the performance requirements. For stateful integrations with millions of messages in short time periods (high-frequency scenarios), the recommendation is to use specialized integration software, leading to the alternative in the middle of Fig. 3.3.
2. In the middle: Both parts, stateful and stateless integration, are handled by specialized integration software. This is recommended for high-performance scenarios where an optimized integration engine is capable of managing the load.
3. On the right: As outlined above, a stateful integration part is not necessary in all cases. A simple transfer of a message (including routing) to the right target system(s) and mapping between data types is all that is required in this scenario. This is best covered by specialized integration software. It is not recommended to use BPMN engines for these use cases as BPMN software is optimized for executing business processes and not for integration. Even though vendors of BPMN engines claim to integrate with many systems out of BPMN processes, it is definitely not recommended to use this functionality. BPMN engines cannot connect to as many systems as specialized integration software can, and they (BPMN engines) are also not optimized for executing complex mappings between data types. Leave those tasks to optimized integration software.

We can conclude that a process-driven architecture relies on the "separation of concerns" principle allowing for a maximum of parallelism during development which increases development efficiency. This is the key principle of *process-driven development*. We gain process flexibility because we can easily adapt the BPMN process models in the PDA layer to changing market conditions and new competitors as the BPMN models are not polluted with technical integration flows. Hence, they are less complex and easier to maintain. The adaptability to changing IT landscapes is ensured by the service contract together with the service contract implementation layer. If there is a change in interfaces or systems, this can be

adjusted locally using the specialized integration software. And finally, we preserved the BPMN model during development—exactly what we wanted to achieve.

3.2.6 Process-Driven Technologies

In order to implement a full-fledged process-driven application, it is recommended to use the following technologies:

1. BPMN engines for the execution of the business processes as well as the stateful integration part of the SCIL. As was outlined in Sect. 3.2.5, the usage of BPMN engines for integration purposes is only recommended if the performance requirements for handling the message volume are met.
2. Business rules engines (or decision management systems (DMSs) as they are also known) complement process engines. BPMN engines concentrate on the execution of process logic, whereas a DMS executes decision logic and/or calculations.
3. Enterprise service bus for integrations—both stateful (e.g., aggregator pattern) and stateless (e.g., routing/mapping) integrations.
4. Although not discussed in detail in this chapter, event stream processing (ESP) software is recommended for new IoT (Internet of Things) scenarios with a multitude of sensors sending signals about, e.g., temperature, pressure, humidity, etc. which need to be filtered and analyzed for business-relevant information. The ESP solution is responsible for signaling business-critical events to the business processes. They are typically not directly connected with BPMN-based processes in the PDA layer of Fig. 3.3; instead, they send the business events to the SCIL which is then in charge of handing them over to the responsible processes. This is mentioned here for the sake of completeness—while not directly relevant to this case study, it gives an indication of potential further use cases for process-driven applications.

This setup ensures a very flexible environment for process-driven applications which is prepared for fast adaptability to changing conditions for a long time to come.

In this section, we've covered the basic ideas of the process-driven approach and proposed theoretical answers to the following questions raised at the beginning of this chapter:

- Is it possible to preserve BPMN process models developed in operating departments following the BizDevs collaboration model during implementation?
- Is it possible to keep the obvious technical and business complexities under control if the process-driven approach is applied? What needs to be considered?
- Which benefits do companies gain by applying the process-driven approach, and which of the aforementioned shortcomings are being addressed by it?

3.3 Implementation Project at SAP Language Services Using the Process-Driven Approach

In Sect. 3.2, we've described the main ideas behind the process-driven approach in reasonable detail because it is the foundation for the ongoing project at SAP Language Services. All the aspects discussed in Sect. 3.2 were completely applied during this project. We will now continue with a closer look at the situation at SAP Language Services before the project and how it was improved using the process-driven approach. The remaining part of this section is based on an article by Matthias Heiler, which was first published in the January-February-March 2016 issue of SAPinsider [1]. It was updated with latest numbers and slightly enhanced.

The SAP Language Services (SLS) department provides translation services (in 40 languages) for a variety of items for SAP products, such as user interfaces, business reports, marketing materials, videos, handbooks, and documentation. SAP Language Services collaborates with several translation agencies across the world and coordinates more than 2800 native speakers in order to achieve high-quality translations, even taking into account the local culture of the respective country for which a translation is needed. Just to give you an impression about the volume that needs to be translated, in 2016 more than 700 million words were translated (one Harry Potter book contains roughly one million words). The business requirements for running the processes supporting these services were so unique that no standard off-the-shelf translation management software could deliver what SAP needed. There are two main aspects that make this process so unique and differentiating: Firstly, the ability to simultaneously ship localized versions of software products and features in a high number of languages is a key competitive advantage for SAP. Secondly, in order to meet this goal and maintain that advantage over time and in changing market conditions, SAP Language Services has developed a range of approaches and processes that are fairly unique in the localization industry. So the SAP Language Services team decided after an intensive evaluation phase to build their differentiating business processes following the PDA methodology and to run them using an SAP product called SAP Process Orchestration.

The first step was to design an overall framework for the business processes. There were several factors to consider in building the business process framework. The services the SLS team has to deliver depend on specific translation scenarios. The process inputs can vary widely, as can the requirements of the process outputs, and the process must be able to produce the required result from these different inputs. There can also be variations within one project. For example, very high linguistic quality is required for the Japanese version—so in this case, machine translation will need to be reviewed by language experts. The Italian version however is only needed for test purposes; the quality need not be perfect, but it is needed much sooner—so here, just using machine translation without expert review would meet the goal better. The source text might be in German, so for Japanese it would make sense to produce an English version first and then translate into Japanese from English, as this will considerably lower the cost of translation into Japanese

(German-Japanese translators are much rarer and consequently significantly more expensive). In addition, every translation project must consider different types of text sources (there are a wide variety of formats and system types that need to be processed, such as typical software file formats, ABAP systems, Microsoft Office files, video files, etc.) as well as different types of texts, such as user interfaces for software, marketing materials, texts for internal communication, and even official financial statements. As a result, each translation process must factor in those requirements by variants in their execution—this was a key influencing factor in the design of the business process framework. In addition, the primary goals of the project included achieving a high degree of automation to improve operational efficiency and allowing for flexible adjustments of the services to accommodate new or changed requirements.

This difficult constellation, consisting of a multitude of different translation requirements, an overly complex IT landscape with several hundred systems to be integrated, and inefficient process implementations with many redundant manual tasks, caused the valuable and highly skilled people at SAP Language Services to spend the majority of their time just to keep the processes alive and running ("keeping the lights on"). Their capacity was obviously not available for innovations (compare Fig. 3.4, left side).

Therefore, one of the key goals of the project is to relieve the team from time-consuming, inefficient, redundant tasks and give them more room for business innovations in new language technologies such as neural and statistical machine translation and other natural language processing technologies (compare Fig. 3.4, right side). Optimizing the IT landscape by consolidating systems, high reuse of linguistic assets, and, last but not least, process automation are the main measures that support this goal.

During many workshops, a list of requirements for the new solution was collaboratively worked out and consisted of the following three main items:

- Best practices, which have been established over many years through cooperation with partners and customers, have to be considered in the new application as

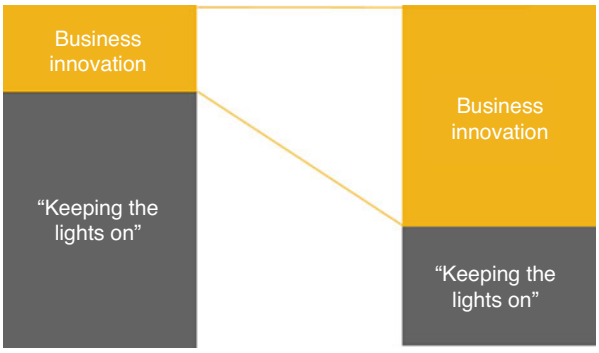

Fig. 3.4 Key goal of the SLS project: freeing capacity for business innovations

well. It requires the right balance between standardization and flexibility without compromising high service levels.
- Agile and sustainable process adaptations must be possible (e.g., adding new translation technologies to a process), even on short notice.
- Process documentation must always be up-to-date and must not be a separate step in a project's life cycle: process documentation must correspond 1:1 to the running processes. The goal is, on the one hand, to minimize effort and, on the other hand, to facilitate the exchange of best practices.

With so much complexity, easily understandable process models were essential to the project. Hence, it was decided to implement the business processes using the process-driven approach including the BizDevs collaboration model. Figure 3.5 shows, for example, the result of the collaboration between business and IT in a BPMN model that could be created using any business-friendly BPMN modeler, and Fig. 3.6 shows the resulting executable BPMN model in SAP Process Orchestration. Note that the two models are identical—exactly what we wanted to achieve using the process-driven approach.

The process-driven approach is generic and independent from specific tools and environments, so it works with any BPMN-based modeling tool. The PDA methodology, with its uncluttered and collaborative approach to process modeling, is an ideal fit for the SAP Language Services project, enabling business users, BPMN specialists, developers, and user interface designers to discuss process logic, business functionality, user interfaces, and services very precisely. The project took the approach of educating business users both in BPMN 2.0 and the PDA methodology. As a result, the business users quickly became BPMN specialists in their own right, capable of using the full BPMN palette. The only restrictions on shape sets used were those imposed by the process engine itself, where the palette was not completely implemented. For those cases, the business users were able to use the implemented shapes to achieve the same outcome, but of course a full implementation would eliminate the need for such workarounds.

As a result, the executable process is truly business driven; and thanks to the early, intensive involvement of key users, acceptance of the model is very high. Communication between the IT and business units is standardized through BPMN and is highly efficient because it virtually eliminates the risk of misunderstandings and decreases the time between concept and implementation. In fact, although this project required a standardized approach to handling multiple different language technologies, the creation and implementation of 65 process models following the PDA approach was achieved within 9 months. Compared to traditional methodologies for implementing processes using programming languages such as ABAP and Java, the PDA approach has an implementation time savings of roughly 75% (status in January 2016). One additional time accelerator in the project was to start directly with the design of to-be processes instead of struggling with legacy as-is processes. In theory, it would have been possible, following the bottom-up approach, to first create process models to reflect the as-is processes and then to use those as the basis for improvement. The team rejected this approach, as it was seen as very

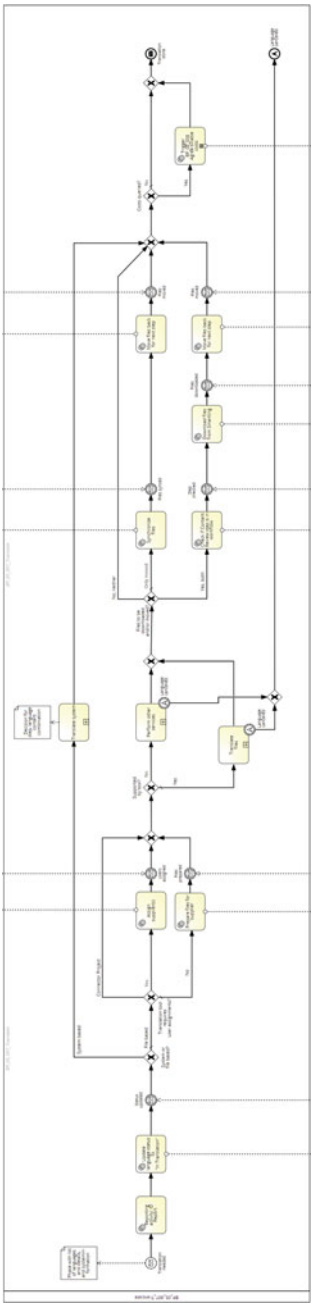

Fig. 3.5 The BPMN-based process model from the business perspective

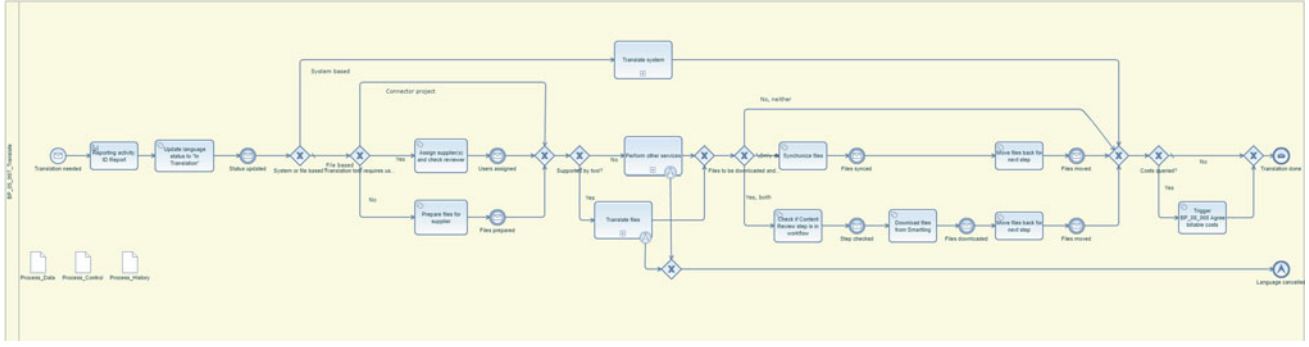

Fig. 3.6 The executable BPMN model in SAP Process Orchestration is identical to the structure of the business process model which was defined before

effort intensive with little to no benefit. Since the business experts were directly involved, they were already perfectly aware of the shortcomings of the existing processes, even without well-defined models or in-depth analysis. The consensus was also that spending time and effort to create those models would have a negative impact on their ability to define to-be processes and that they would risk carrying over undesirable elements and patterns from the as-is processes for the sake of expediency. Therefore, the approach selected was to start with the to-be processes and then to use those as the basis for further optimizations. The team found that they also faced the fairly common difficulty of abstracting from the given infrastructure and translation tools. From the first iteration, while it was relatively easy to define generic processes that could be used for all translation types for aspects such as project management, it was significantly harder to do so for the parts dealing with the actual translation itself in all the different tools. In fact for these processes, the first iteration did not succeed in completely separating the business process from the systems used. However, after gaining some experience in the practical application of the methodology, the team was able to achieve this goal, so that now changes can be made in either the business or the technical layer, without impacting the other layer. For example, it is possible to add or replace translation tools without changing the business process. Where changes need to be made that impact both layers (e.g., a new business activity is added and requires a new system), these changes can be implemented in parallel, increasing development efficiency.

The project is still ongoing and has evolved since then. The latest numbers after a total implementation time of 33 months are impressive (December 2017): 206 really complex nontrivial business processes, 169 integration processes (SCIL implementations), and 126 user interfaces speak for themselves. Process execution times have also been noticeably reduced: for example, the execution time for the end-to-end order process for marketing materials was reduced to one-third of the original execution time. As microservice architectures have become more common, SLS has also seen additional benefits to the PDA approach. On the one hand, it is easy and efficient to integrate new microservices into the process as they become available. On the other hand, the process-driven approach provides a highly effective framework for orchestrating diverse microservices to create business value in a range of different scenarios. Overall, SLS achieved the following:

- Improved efficiency in process execution
- Improved user experience
- Higher automation rate
- Increased flexibility and adaptability
- Increased transparency in operational business

Besides that, two more goals have been reached:

1. SLS took a major step toward active process management, where business and IT work closely together and can adjust their processes more quickly and consistently.
2. This in turn gives the team opportunities to expand the services offered and to develop and even commercialize business models relevant to the digital era.

Table 3.1 Aggregated metadata for the SLS process collection

Collection name	SLS
Process count	206 models, up to 3 versions/model
Domain	Managing of translation projects
Geography	Worldwide
Time	03-2015–12-2017 (ongoing)
Boundaries	Cross-organizational 26%, intraorganizational 24%, within department 50%
Relationship	Is being called/calls another 100%
Scope	Core 206, technical 169
Process model purpose	Executable
People involvement	None 45%, partly 55%
Process language	BPMN 2.0
Execution engine	SAP Process Orchestration 7.5
Model maturity	206 productive

Table 3.1 gives an overview of the implemented processes using the process collection template for categorizing business processes described in Chap. 2.

It should be noted that the process version count does not reflect the number or frequency of changes to the models. A new version is only created when this is technically necessary, e.g., in case of interface changes. Managing version compatibility can be a challenge—in addition to keeping the number of versions low, the team has developed mechanisms to automatically upgrade running process instances to the latest version on a new feature release (e.g., at key points, the process checks if it is running in the latest available version; if not, it cancels itself and restarts at the same process point in the latest version, with the same data). In terms of assessing maintainability or sustainability, the number of process instances is perhaps the more telling figure. For business processes alone, there have been a total of almost three million instances, with on average around 16,500 instances running at any given time. The figures for technical processes are significantly higher. Application support is facilitated by a dedicated process that provides support staff with relevant error data in the form of a human task in case of technical or business errors. One team role has dedicated responsibility for operations/maintenance, mainly in terms of oversight on tickets/tasks; remaining operations activities are carried out by all team roles on the fly—a BizDevOps model.

One further insight that the team has gained is that it seems much easier to manage this kind of implementation project using agile methodologies. A new team was set up for the project, consisting of business and IT specialists from SAP Language Services, supplemented with PDA and BPM experts from consulting partner itelligence AG. Initially, a hybrid development management approach was selected; however, as the team has matured and gained experience, this has evolved over time to an adapted version of agile development methodologies. While some elements (e.g., teams of ten) are not practical for our purposes, clearly defined user stories, sprints, and feature-based deliveries have proven valuable. The team

has found it helpful to describe how they work in detail using a process model. Overall, requirements from stakeholders are continuously added to the concept backlog. After prioritization, they are grouped into user stories that form coherent units of business value. Part of the concept work includes carefully examining dependencies (both business and technical) between the user stories—failure to do this early on can block deliveries of features that are in themselves complete, but cannot be deployed separately from parallel developments. In addition to creating process models and UI prototypes, the concept team (consisting of business and IT specialists) also prepares backlog items for implementation. Developers attach effort estimates to the backlog items; and these, together with the known dependencies, are the basis for development sprint planning. This approach has provided a great deal of flexibility in terms of delivering features as soon as they are ready, as well as providing greater transparency for all team members around the status. Delivery frequency is weekly, with larger updates reaching production on average approximately every 2 months.

In summary, therefore, we find that we have now been able to answer our five questions:

- Complex real-life scenarios can be completely covered using a model-based approach.
- BPMN models developed by business departments using the BizDevs collaboration model can be preserved 1:1 during implementation.
- The process-driven approach provides an effective methodology for mastering complexity, both business and technical.
- The benefits are listed above.
- The BizDevs collaboration model has proved to be a vital tool in addressing alignment issues, and the benefits proposed in theory are observed in practice.

3.4 Conclusions and Outlook

3.4.1 Conclusions for Researchers and Practitioners

This chapter of the book has outlined the fundamentals of the process-driven approach (PDA). As result of applying the process-driven approach, you get process-driven applications. These are defined as business-oriented applications that support differentiating end-to-end business processes spanning functional, system, and organizational boundaries by reusing data and functionality from platforms and applications. We can summarize the key aspects of the process-driven approach as follows:

- Process-driven *collaboration* between business and IT (BizDevs)
- Process-driven *thinking* that considers shape semantics and the process engine while modeling

- Process-driven *methodology* that develops process models top-down without considering restrictions—no analysis of the current process implementations
- Process-driven *architecture* including a reference architecture for process-driven applications
- Process-driven *development* that rigorously applies the "separation of concerns" principle to achieve a maximum of parallelism during development
- Process-driven *technologies* comprising a BPMN engine, business rules engine (or decision management system), integration software such as an ESB, and ESP software for scenarios relying on events

The approach is being applied in a complex project at SAP SE. SAP Language Services (SLS), part of the Globalization Services department at SAP, has to solve the challenge of standardizing their differentiating end-to-end language production processes while retaining broad flexibility to meet a wide range of changing requirements. So far, a total of 206 complex processes have been implemented within 33 months. The advantages gained to date by the application of the process-driven approach for this project can be summarized as follows:

- Time
  - Shorter development time due to parallel independent development
  - Shorter innovation cycle and faster time to market
  - Shorter strategy-to-reality cycle
- Money
  - No additional documentation necessary (modeled process = documented process = executed process)
  - Cost benefits during development *and* maintenance
  - No need to buy additional software for process mining or business activity monitoring if the process engine collects comparable data and provides relevant analytical tooling (depends on the engine used)
- Higher-quality implementation output (more precise, gets it right the first time)
- Increased flexibility on both sides: business process flexibility and flexibility regarding the integration of various IT landscapes
- Increased implementation efficiency as the first implementation is immediately fitting requirements due to early end user involvement resulting in an increased acceptance
- Transparency
  - Increased transparency during process execution
  - Increased transparency by analyzing automatically collected process execution data (via BPMN execution engine)

- Ability to act: PDA offering the best-possible management support in driving a company's strategy

It is advisable to use the process-driven approach in the following cases:

- Alignment of business and implementation requirements in a single BPMN model is important (only one common BPMN model for both sides, business and IT).
- Independence from the system landscape is critical for the resulting application.
- More than one system needs to be integrated.
- The system landscape on which the processes of the solution must run is not stable.
- The solution is complex and justifies the effort involved.
- The solution will provide a competitive advantage.
- The processes in the solution are expected to change frequently.
- The processes in the solution will be used in other organizational units, areas, regions, or other subsidiaries or companies.

However, if none of these statements apply to a development project, it is certainly worthwhile to consider alternatives. The application of the process-driven approach has proven (at least for the SLS project) the following:

- Real-life, complex business processes can be completely modeled and executed using a graphical notation (BPMN).
- BPMN-modeled business processes can really be executed as they were initially planned by the business (preservation of the business BPMN model during implementation).
- Business and technical complexities can be controlled using the right methodology and just one notation (BPMN).
- The BizDevs collaboration model achieves unprecedented efficiency and eliminates misunderstandings. (Thinking in process engines executing business processes forces a new level of precision as it requires that all details have to be made explicit. As a result, companies understand much better how they really work).
- The BizDevs collaboration model requires a thorough understanding of the complete BPMN shape set on both sides—both business and IT. Experience at SLS has shown that while this does require a learning effort especially on the business side, this investment more than pays off in terms of the results. The process-driven project has been invaluable in providing practical experience of the kind of lifelong learning that is fundamental to success in the digital era.

3.4.2 Outlook

The process-driven approach is still in its early stages. However, the results achieved in a first really complex real-life project are more than promising. It seems as if implementation efficiency can be significantly increased compared to common

programming approaches. Additionally, the process-driven approach is not only a solution for the first implementation. Due to its modular design, it also helps to reduce the maintenance effort after going productive. The transparencies gained during process execution and after finalization are further key arguments in favor of the approach. For sure, the results have to be confirmed in more projects of this complexity, and both aspects need to be analyzed in more detail: the initial development effort/efficiency and the maintenance effort/efficiency. Besides the mentioned aspects which are worth more research effort, the following list summarizes some ideas for further research questions:

- How suitable are current BPMN engines and their development environments for the development of applications following the process-driven approach?
- What does the ideal development environment for the process-driven approach look like?
- Which additional development guidelines can be given to PDA developers?
- Can the promises of the process-driven approach be confirmed by further projects?
- Can the BizDevs collaboration model be further detailed?
- Are BPMN choreography diagrams useful in the process-driven approach?
- Can BPMN collaboration diagrams be utilized to explicitly visualize the vertical process collaboration between the layers?
- What are the influences of latest IT trends (e.g., in-memory DBs, big data, cloud computing, mobile, Internet of Things, machine learning, NoSQL DBs) on the process-driven approach?
- How can the extensibility of process-driven applications be achieved (e.g., by extension points which are also applied if a new version of a process-driven application is shipped by a vendor)?
- Which prerequisites must be fulfilled for a roundtrip of BPMN models between business-oriented modeling environments and developer-oriented IDEs?
- How can customizing of process-driven applications be achieved?
- The process-driven approach involves a learning effort on the part of project team members that is representative of the type of lifelong learning needed to succeed in the digital era. How can organizations best harness the experience from process-driven projects as they seek to establish a culture and methodology of lifelong learning that fits their unique situation and needs?
- Successful process-driven projects result in substantial efficiency gains for an organization, creating room for further innovation and new business models. New business models will likely require then new process-driven projects to implement them, which the organization is now equipped to do. How can organizations structure these innovation cycles to maximum benefit?
- How can the PDA approach best be combined with agile software development techniques?

Obviously, there is more to explore in the domain of the process-driven approach. We hope that the publication of the results gained by the complex SAP Language Services project and the application of the process-driven approach for differen-

tiating business processes has provided interesting insights for researchers and practitioners alike and motivates them to invest more into this promising approach. It can be a starting point for a new wave of business process implementations helping companies to prepare themselves for the digitalization era.

References

1. M. Heiler, Managing modern business processes: how to use a process-driven architecture to achieve flexible, efficient processes. SAPinsider **17**(1) (2016). Also available online at http://sapinsider.wispubs.com/Assets/Articles/2016/January/SPI-managing-modern-business-processes. Accessed 28 Dec 2017
2. OMG, *Business Process Model and Notation (BPMN), Version 2.0* (2017), http://www.omg.org/spec/BPMN/2.0/PDF. Accessed 27 Dec 2017
3. B. Silver, *BPMN Method and Style*, 2nd edn. (Cody-Cassidy Press, Aptos, 2011). ISBN 978-0-9823681-1-4
4. V. Stiehl, *Process-Driven Applications with BPMN*, 1st edn. (Springer, Heidelberg, 2014). ISBN 978-3-319-07217-3