Structured Communication—Approaching S-BPM with Microsoft Technologies

14

Robert Singer and Stefan Raß

Abstract

Many enacted business processes in the field use (more or less intense) communication to forward work to the next participant in an activity chain. Communication can be oral (personal, phone) or technically supported (e-mail, phone). It can be unstructured using natural language—typically text or spoken word—or structured using formal language (business objects) typically stored in systems. Based on decades of research in the domain of the social sciences, we know that an understanding of how organizations work are based on communication and language. Therefore any technology to support the execution of business processes should support communication between process participants. This is the concept of S-BPM. Here, we present the results of work in the field to develop a platform to model and execute business processes as interaction between actors. As process models predefine work we call this way of interaction *structured communication* (using standard e-mail exchange). To enable also cross-company communication (process orchestrations) we technically implemented the platform as a so-called *multi-enterprise business process platform* (ME-BPP) using cloud technology. The contribution uses a real-world case to demonstrate the need for a communication-based view on business processes. The case reflects the situation typically for large-scale international companies with world-wide activities and with focus on processes related to order fulfillment, including manufacturing. Further on, an IT architecture to support the enactment

R. Singer (✉)
FH-Joanneum—University of Applied Sciences, Alte Poststraße 147, 8020 Graz, Austria
e-mail: robert.singer@fh-joanneum.at

S. Raß
StrICT Solutions GmbH, Plüddemanngasse 39, 8010 Graz, Austria
e-mail: rass@strict-solutions.com

A. Fleischmann et al. (eds.), *S-BPM in the Wild*,
DOI 10.1007/978-3-319-17542-3_14

of such *distributed* processes is discussed. The contribution is intended for practitioners with some IT background and/or interests.

14.1 Introduction and Motivation

In this section we will report and discuss typical situations in the field—related to business process management in general and the execution of business processes in particular. These situations will provide the context and the motivational background for the analysis: the use of S-BPM as business process modeling and execution paradigm.

For example, let's think about a typical work situation in a manufacturing company.

When the phone rings it is always something urgent, but Bob, the planning manager of the company, has no choice and picks up the phone. The friendly voice of Pieter wishes him a good morning, but the strange feeling in his stomach remains. Pieter is responsible for consolidating orders from several industrial customers; this includes orders from the own sales organization (brand) and from OEM[1] customers. Pieter is located in another European country. In principle, he could simply enter all requests (new or changed sales orders) into the company's order system, according to some simple business rules, and the factory, represented by Bob, has to answer via the system (accept or reject). Several key performance indicators are automatically recorded via the system, measuring the flexibility and reliability of the manufacturing site. But because of a trusted relationship, Pieter typically informs the factory in advance and asks for feasible solutions: can you do more of this product in week 24? Can we change some quantities from type A to B in this month?

On this day he asks to start two weeks earlier with the production of a new product for a very important customer and he needs an answer within two hours. That needs a lot of hectic personal communication and commitment from engineering, production, purchasing and logistics. Will the manufacturing equipment be ready (e.g., moulds), can we conduct a trial run in advance (including losing capacity because of a lost shift), can we bring in the needed material in time, etc. And, if we cannot handle this situation, are there additional options like moving orders between European and Chinese locations? The one and only tool to solve such riddles is communication. Some of the communication threads are serial (first check this, then that), some are parallel (each department checks). Additionally, the communication thread spreads over many people for the issue to be discussed personally, by e-mail, or by phone; they use, send and receive data using simple office documents and or systems; and it involves people from outside the organization as well, e.g., suppliers, engineering colleagues or the logistics department in the business unit headquarters in Taiwan.

[1]Original Equipment Manufacturer.

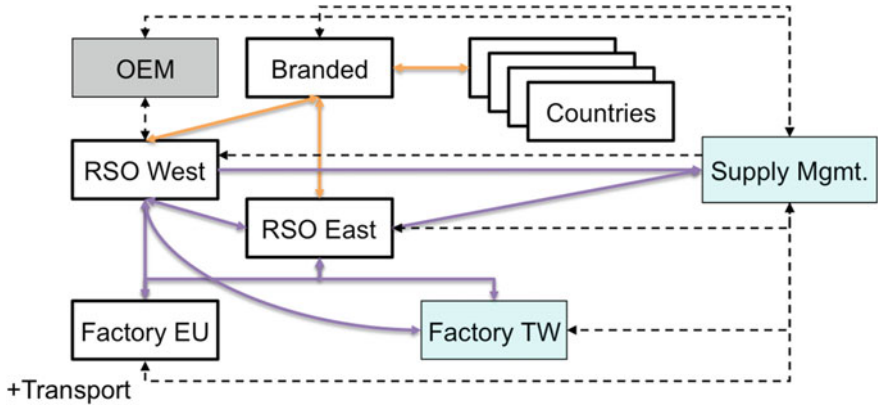

Fig. 14.1 Communication connections (relations) between the involved parties in the text case (business unit view). Each *box* represents a particular organizational unit, as described in the text. Abbreviations: *RSO* Regional Sales Office, *TW* Taiwan, *Countries* legal units in each European country. *Dashed lines* visualize further communication and line of command relations not explicitly discussed in the case

Bob will come up (as always) with an answer in time; after that, Pieter will discuss the committed proposal with the customer and, if they commit too, he will enter the agreed sales order (new or changed) into the system. Bob then will accept the changes in the system. To make it more complicated, the solution has to be communicated to and or committed with the business unit supply manager (Amy) located in Taiwan (possibly delayed because of time difference). Obviously, there are some interesting issues. An infamous point is that nobody knows what happens afterwards. There are private conversations and phone communications, notes on napkins and some or several e-mails all over the world. Maybe there will be trouble two months later with this order and the customer has to be informed that the order has to be postponed by two weeks. How to analyze what happened and why? The only visible fact is the acceptance of the new or changed order contradicting documented or undocumented policies or business rules. And—how to interpret the measured KPIs? If we try to visualize the "relations" and information flow between all involved **actors** of the *Order Fulfillment* process (on business unit level) we come up with Figs. 14.1 and 14.2.

Practice shows that such processes are the norm and can neither be modeled in full with "standard" modeling notations (such as *Business Process Model and Notation* (BPMN) or *Event-driven Process Chain* (EPC), for example) nor automatically executed based on these business process models—in this case, the organization has some documented business process description in RACI[2] form. As can easily be seen, we are confronted with a typical knowledge-intensive process; the main ingredient is **knowledge**, the output is a decision. But of course there is an

[2]Responsibility Assignment Matrix.

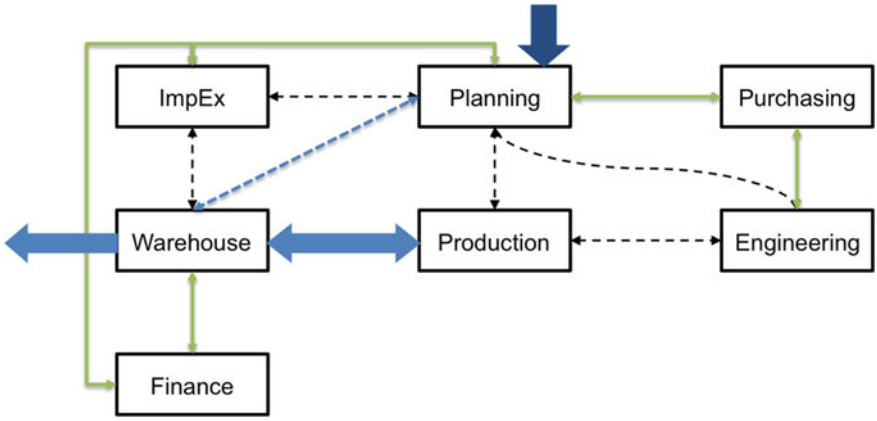

Fig. 14.2 Connections between the parties in the case (site view). Each *box* represents a particular department, as described in the text. Abbreviations: *ImpEx* Import and Export. *Dashed lines* visualize further communication relations not explicitly discussed in the case

inherent structure in such a change **case** and it should be noted that such change requests typically violate "standard" business rules and policies (fixed sales orders over a period of three months, for example). Any change therefore has to be evaluated on its own.

The question now is: Can we bring more structure into the work flow? Or, shouldn't we simply stick to a system, such as an *Enterprise Resource Planning* (ERP) system? The first question we will answer with "yes"—as we will discuss soon the second one with "no". The above case uses information and enters information into an ERP system, but that has no relation to any predefined work flow (but it is linked with the organizational structure and roles).

But there is actual IT support which works very well: E-mails! All involved parties (lets call them actors) can send any other actor a message. We can even cross organizational borders—and world-wide. It is also possible to send messages to people we do not even know, as long as somebody else knows them—so we can get answers to our questions from people we didn't know beforehand (we call this mobile messages). Additionally, we can send data together with our messages; any actor in the communication path can store or (depending on the data format) modify the data, which often are office documents: an actor, for example, can add a column in an *MS Excel* file and forward it to another actor. We can see from this that e-mail communication is a way for flexible—but unstructured—enactment of business processes. It can even be used to execute business processes we never thought about —it can *bootstrap* a process. Nevertheless, neither the communication thread nor the data is centrally stored—there is no central repository. It is therefore difficult to investigate what happened in the past; more or less forensic work—not so good if we are interested in compliance and process improvement.

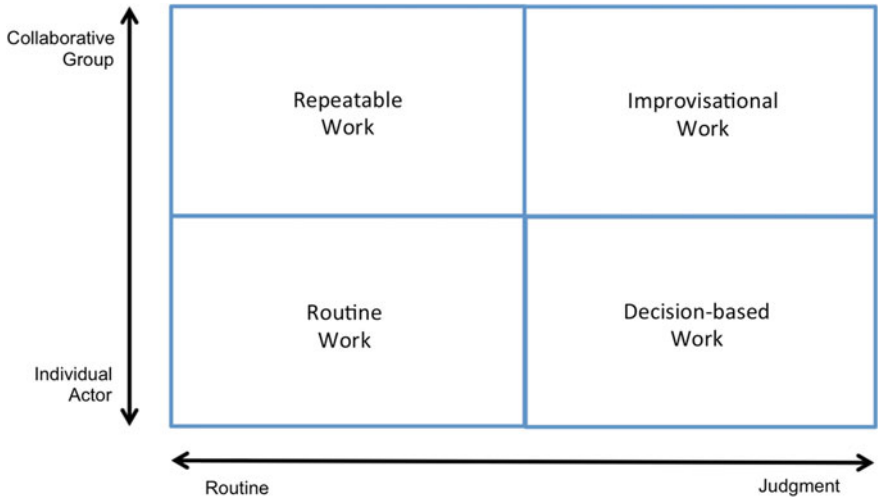

Fig. 14.3 A proposal of Davenport (2010) to categorize knowledge-based business processes

The predominant benefit of e-mail as business process execution tool is its flexibility and ease of use (a really flat learning curve); the disadvantage is that it is not efficient in cases when it is possible and wise to define a work flow (or parts of work flows) in advance. Not all business processes are purely ad hoc. These types of processes are typically expensive, as we need experienced and educated knowledge workers to evaluate the situation, develop a solution and make decisions—this can only be done by well-educated and empowered employees, who are typically also well paid. It is also difficult to agree on service levels for such processes.

We can see in Fig. 14.3 that business processes can be categorized (gradually) in two dimensions representing level of interdependence (number of actors) and complexity of work. Depending on the value of each dimension, a business process will have a more or less predefined structure.

14.2 Structured Communication

The question now is, how to create a system for structured communication? As methodology, *Subject-oriented Business Process Management* (S-BPM) seems to be the perfect foundation for such an approach—simply because it is structured communication: we have a network of actors (subjects are conceptualizations of actors), who are synchronizing their work through the exchange of messages. In the following paragraphs we will identify some (somewhat technical) requirements we have to consider for an implementation; and we want to create an IT platform for the execution of S-BPM models based as closely as possible on natural communication behaviors, and using as many already existing software platforms and applications as possible.

After collecting the requirements we will then be able to argue why a certain architecture is a feasible and useful approach to realize a solution for the execution of business processes. A business process execution system is an integral part of a business process management system and bridges the gap between models stored in repositories and their IT-enabled execution. It brings the information and knowledge, which is embedded in the model—the blueprint of the business process—to life.

In this context we understand a business process model as an entity for defining a plan to deliver services (or products) to customers, i.e., defining what we plan to do and in which logical order. If this sounds easy, let me assure you that it is not. Business processes are a means to manage service delivery, and as broad as the range of possible services is, the semantic spectrum of the term "logical order" is as broad. We therefore have to understand "logical order" in a more fuzzy way, as discussed in the previous section.

Therefore, business process execution systems need to support flexibility in the order of activities to be performed or needed. What we mean is, if we define a business process model, we define the future, how we plan to do the work. But from practical applications we know that we cannot plan all possible future situations. Depending on the type of service, organizational culture, or industry—the corresponding business process will create situations where we will not be able to stick to the predefined business process logic, simply because the concrete situation has not been considered appropriately. That means any business process execution system needs to facilitate this fact in some way. The typical case presented in the previous section is the context for the definition of the requirements of a fully featured business process execution system.

- Business process execution systems need to support **concurrency**. That means activities (or, synonymously, tasks) are executed simultaneously and potentially interact with each other; the simplest interaction would be synchronization after each concurrent activity has finished. In modeling notations this typically is reflected using symbols for AND-splits and -joins.
- Business process execution systems need to support **distributed** execution. Business processes cannot be seen as isolated workflows for administrative purposes only, but as a means to coordinate a value system with supply chain partners. That means actors in a business process are geographically distributed and not necessarily members of the same enterprise (e.g., manufacturer and supplier).
- Business process execution systems need to support **mobility**. This is a consequence of how we work today, but also leads to technical requirements for an implementation of a business process execution system.
- Business process executions systems need to support **flexibility**, i.e., the possibility of human process participants deviating from the predefined process path in case of an unexpected (and therefore not modeled) situation or exception—we need the capability to deviate from the path initiating so-called ad hoc activities while running a concrete instance of a process.

From these requirements we can conclude that BPMS based on any technology which executes business processes under the central control of some software (the process engine) cannot fulfill the criteria discussed above in its full consequence; this is especially true for the requirements *concurrency* and *distributed*, which lead to technical questions (Butcher 2014), which cannot be discussed here. Today such systems mainly focus on BPMN as modeling notation and more or less proprietary solutions to execute the models. Such "classical" workflow systems typically support office processes very well (for example the famous "application for leave" process) within one organization, but have serious difficulties executing real-world processes crossing organizational boundaries; additionally, from a socio-technical view on systems, we can also conclude that communication plays a central role in social interaction and therefore it is a natural way to think about the coordination of work.

Another issue we have to consider is the handling of data or **business objects**. Here we have the same issues as above: who stores the data and where? If we think of a process execution system as an ERP system it is clear that all data is centrally stored in exactly one database. This database is "owned" by one organization (even if it is located somewhere else) and the organization has full control over content and states of the datasets. But how do we handle the data we send to other process participants?

This demand is now reflected in new developments in the domain of BPM, such as *BPM Platform as a Service* (bpmPaaS), *Multi-Enterprise Business Process Platform* (ME-BPP), *Cloud BPM*, and *Social BPM*. The term bpmPaaS can be defined (Dixon 2012) as "*the delivery of BPM platform capabilities as a cloud service by a service provider*". An ME-BPP is defined Dixon (2012) as a "*high-level conceptual model of a multistakeholder environment, where multi-enterprise applications are operated. Multi-enterprise applications are those purposely built to support the unique requirements for business processes that span across more than one business entity or organization. They replace multiple business applications integrated in serial fashion*". Now, that is exactly what we are looking for: an ME-BPP. The next sections will discuss what we found on our *excavation* in the field.

14.3 How to Execute S-BPM Models

In this section we will sketch our journey towards a *Multi-Enterprise Business Process Platform* based on S-BPM (Singer et al. 2014), i.e., a so-called agent based approach. As already mentioned, one very important intention was to use as many available tools as possible in the field. Although this section contains some technical stuff, we do not have the intention to discuss things like code snippets in detail, but to give some deeper insight what is needed behind the scenes to **execute distributed and concurrent business processes**.

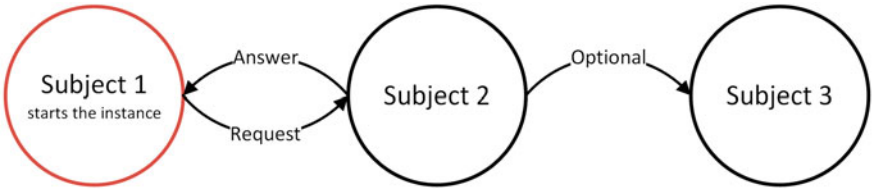

Fig. 14.4 An example of a *Subject Interaction Diagram*: it contains three subjects and all exchange messages. Subject 1 starts the process by sending a request to Subject 2; Subject 2 makes a decision, sends an answer back to Subject 1 and, in case of a positive answer, informs Subject 3 about the decision

An S-BPM process is defined via the communication exchange channels between subjects (agents are instantiated subjects in this context, or the other way round—subjects are generalizations of agents) (see Fig. 14.4). Additionally, each subject has a defined (but invisible to the outside world) internal behavior, which is determined as a process flow using states for receiving or sending a message (to another subject), and states in which the subject is doing some work (see Fig. 14.5). States can be flagged as starting or ending states and are connected using directed arcs.

A platform for enterprise use cannot be built from scratch, but has to be integrated with an available IT infrastructure (e.g., server platforms). Additionally, we need some business process execution technology we can use as a starting point; one prerequisite is that it must be usable in a software development platform (we need to write software using some functionality offered by others) and be able to run in a cloud environment (we will explain this later). Besides other points, and because there is already a platform available which is based on *Java* (but limited to running on *MS Windows*), we decided to start investigating other available technologies based on the *MS Windows* technology stack. Especially, the *Workflow Foundation* (a .NET programming framework) offers a promising starting point, as will be explained now. Principally any other workflow engine can be used, as long as it integrates with the used server platforms and offers similar functionality.

14.3.1 Workflow Technology

The *Workflow Foundation* (WF) workflow provides functionality to maintain state, get input from and send output to the outside world, provides control flow and executes code—this is done by so-called *Activities*. An *Activity* can be modified in any thinkable way and WF workflows can be graphically constructed within the development platform (*Microsoft Visual Studio* in our case). An example of a WF workflow is depicted in Fig. 14.6. The execution is done by the workflow engine, which is part of the operating system (the .NET environment).

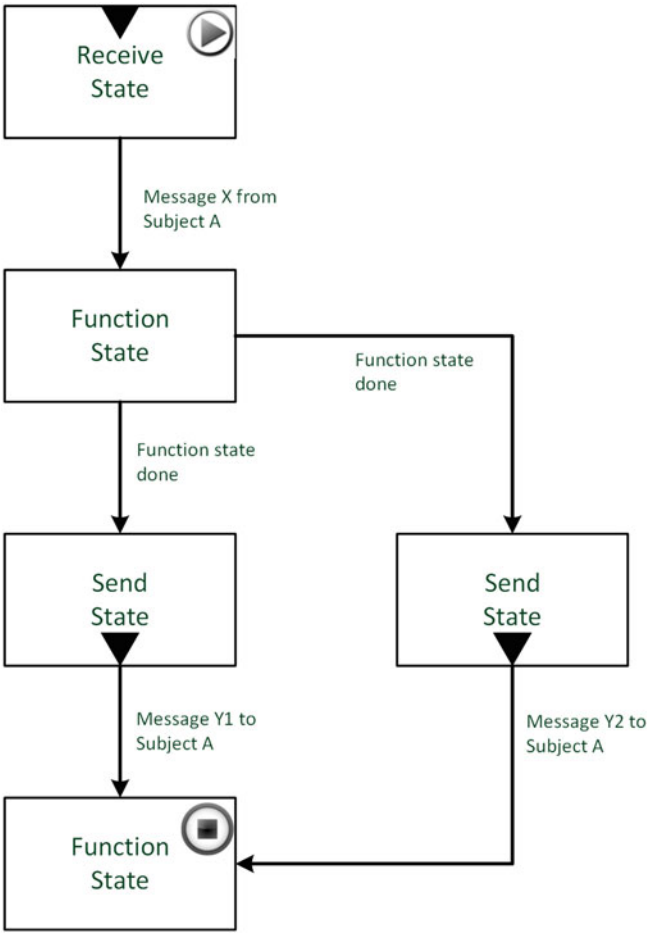

Fig. 14.5 An example of a *Subject Behavior Diagram*: *triangles* at the *top* symbolize *Sending*, *triangles* at the *bottom Receiving States*; other states are *Function States* and states in general are marked as starting or ending state (by *play* and *stop* icon). State transitions are modeled as directed arcs

If a process instance is not needed for the moment—e.g., because of waiting for a message from another process participant—the state of a workflow can be **persisted** (into a persistence store) and stored safely until the continuation condition (e.g., an arriving message) is met—an important functionality for **long-running processes**. Based on the requirements discussed above, a workflow might run on different threads in different processes and on different machines during its lifetime. Any application built on WF technology is therefore **scalable**, since it is not confined to a single process on a single machine. Furthermore, activities can be executed **concurrently**. The chosen *WF workflow* technology supports these requirements.

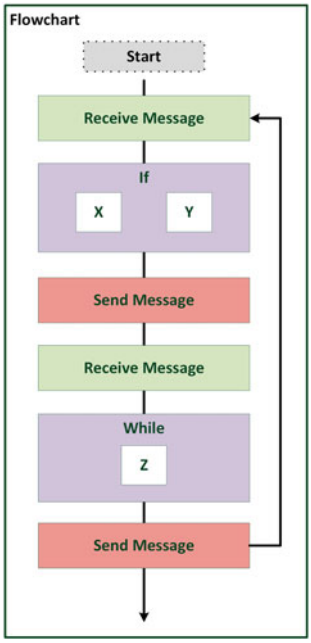

Fig. 14.6 The structure of a WF workflow; all work is done by activities. The *Flowchart Activity* is enacted by the WF runtime engine and process flow can be routed back to previous *Activities*

If we simply compare the representation of an SBD (see Fig. 14.4) with the representation of the WF workflow as depicted in Fig. 14.6, we can conclude that it may be possible to map any SBD to a WF workflow. This will be our first topic to study and we will show that this can be done. For the S-BPM methodology to work with WF, custom activities[3] are needed to perform the functionality of the S-BPM states: so we need to write some code to get a custom *Function*, *Receive* and *Send Activity* for standard S-BPM behavior, as defined by Fleischmann et al. (2012). Technically, we get a process model defined as an XAML file[4] (an XML-based language).

14.3.2 S-BPM as Windows Workflow Model

The mapping of an SBD onto a WF workflow can be done in the following way: there are four elements in S-BPM which need equivalents in WF workflows: subjects, states (send, receive, function), transitions and parameters (local, global). The WF equivalent for a subject in general is a WF *Flowchart Activity*. Each S-BPM state and its following transitions are a custom WF *Activity*. Parameters in S-BPM are

[3]From a programming point of view this means that we have to develop our own S-BPM classes using the WF Activity classes as base classes.

[4]The *Extensible Markup Language* (XAML) is a *Microsoft* format to store executable program code.

converted to variables in WF, which provide the same functionality. S-BPM parameters assigned to S-BPM states become WF variables assigned to WF activities.

As we need WF *Activities* with specific behavior, we need to "enhance" the standard *Activity* class (we use C#) with additional functionality. Not to forget, any *Function State* can include so called *Refinements*, that is, any additional functionality, for example, interacting with other applications or hardware.

All information about processes and their execution has to be stored in a proper way. Therefore, all defined processes as well as their running instances are stored within a central process repository on the server side. Additionally, we have to consider a mapping between organizational roles and subjects, i.e., a specific role is mapped onto a specific instance of a subject (an agent); roles are typically defined in the active directory structure of the IT infrastructure. Normally, a single user can be assigned to several subjects and a subject can be assigned to several users.

As we can see, it is rather straightforward to map an S-BPM model and it goes off without a hitch. It is therefore also possible to automatically transform S-BPM models from other platforms.[5] That means we have a general technology which is able to represent and enact *one* subject, i.e., we can map the internal behavior of a subject onto a Microsoft .NET workflow. The next step now is to find out how several subjects can interact with each other, or in other words, how we can map an entire business process onto WF workflows. As a reminder, any agent or actor in the case at the beginning represents a subject. It is important to understand, that the communicating subjects are *distributed*, that means we do not have a central control over them; each of them acts *independently* and *concurrently*.

14.3.3 The First Prototype (PROMI)

The basic component of our first architecture model is an application titled **Scheduler**, as it is responsible for scheduling all messages between the interacting subjects. The *Scheduler* represents the server-side execution environment for processes, while all necessary interactions with users are performed on the client side. The basic concept of this server component is depicted in Fig. 14.7. First of all, the *Scheduler* is acting as a host environment for all WF workflows. Each instance of an S-BPM process consists of several communicating subject instances (agents). The *Scheduler* manages loading, instantiating, termination, unloading, and the storing of workflows, including the synchronous or asynchronous execution of workflows. Furthermore, the *Scheduler* manages the message exchange between the subject instances (agents). Messages can be exchanged by the use of specifically designed activities from within the WF workflows. The *Scheduler* takes care that messages are delivered to the dedicated recipients.

The **message pool** concept is a central mechanism of S-BPM; in S-BPM all subjects have their own input message pool,[6] and message exchange between

[5]As proof of concept we imported a process designed in the Metasonic Suite (www.metasonic.de).

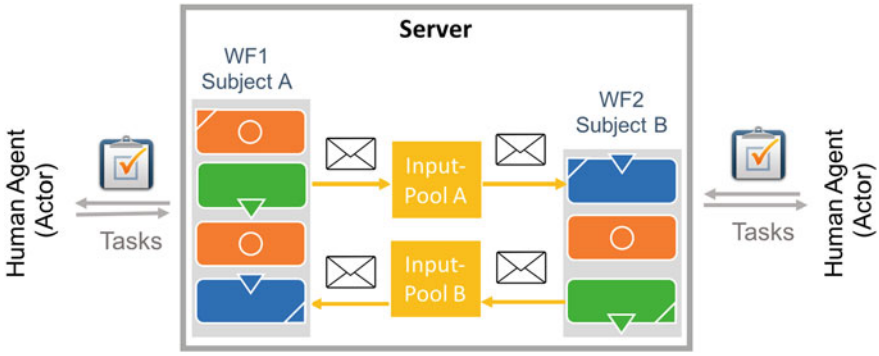

Fig. 14.7 The figure shows the execution of a process with two subject instances, i.e., agents (*Subject A* and *Subject B*). The behavior of each subject is defined by a sequence of custom activities defined by a WF workflow (*WF1* and *WF2*). The workflow activities can basically perform three different actions: send a message, do something and receive a message. Each subject has its own message pool. A workflow communicates with clients in the form of *Tasks*

subjects can be synchronous or asynchronous. We need both types, as subjects are instantiated as agents and an agent can be a human or a machine, or a service. Further, a subject has full access to all messages in its input pool and it can remove any of these messages for processing. This is a fundamental functionality for real-world business processes, reflecting the fact that a process participant (actor) decides which process to continue next (in general it allows setting of priorities).

Consequently, any agent can send messages to the message pool of another agent and take out messages from its own message pool. Workflow activities may require user interaction. In our implementation concept the user interaction is performed client-side. Therefore, the *Scheduler* generates a so-called *Task* for the responsible agent and also includes corresponding data fields (read and or write); in case of a human agent (user-task) this will typically lead to a form to be completed and returned to the *Scheduler*. The information flow between the components of the architecture is depicted in Figs. 14.8 and 14.9.

Based on these concepts we have built a platform to execute S-BPM process models. As *structured communication* is our motto, we use an enterprise e-mail infrastructure to start the processes and to answer the tasks. Following this approach there is no need to learn a new application and we can use the benefits of e-mail as a tool to execute processes, but in a (more) structured way. And, we have the possibility at any time to send an "unstructured" e-mail to anybody inside or outside the defined business process. This implements, in a very uncomplicated way, the S-BPM modeling approach "modeling by restriction" (Fleischmann et al. 2012): we gradually move from unstructured to structured communication.

[6]Since any technical implementation has limited resources, input pools are limited in their size. If a pool is full, no further message can be received and the situation has to be handled by the software. In worst case we have a deadlock situation: waiting for a message which never can be received.

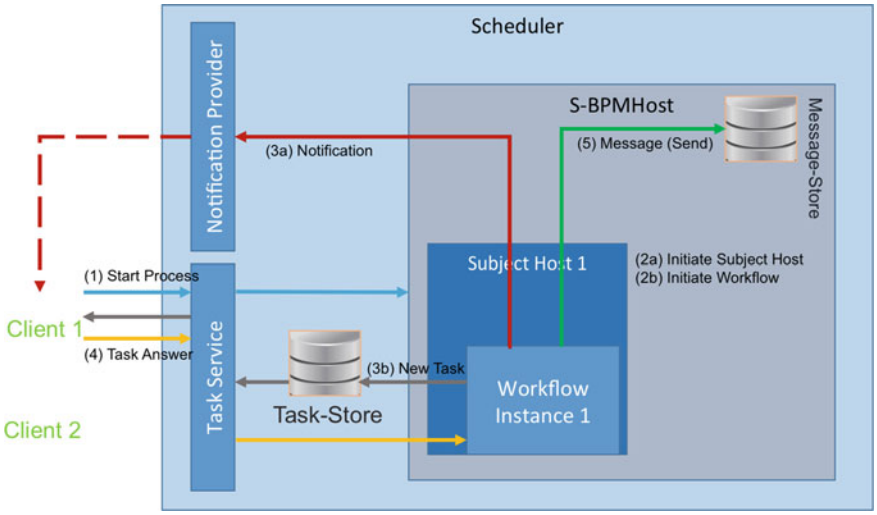

Fig. 14.8 Part 1: (*1*) a process is started by a user on a client. The system then (*2a*) instantiates the environment and (*2b*) the workflow, (*3a, b*) creates a *Task* for interaction with the user, (*4*) handles the answer and (*5*) generates a *Message* to be forwarded to the next *Subject* in the process

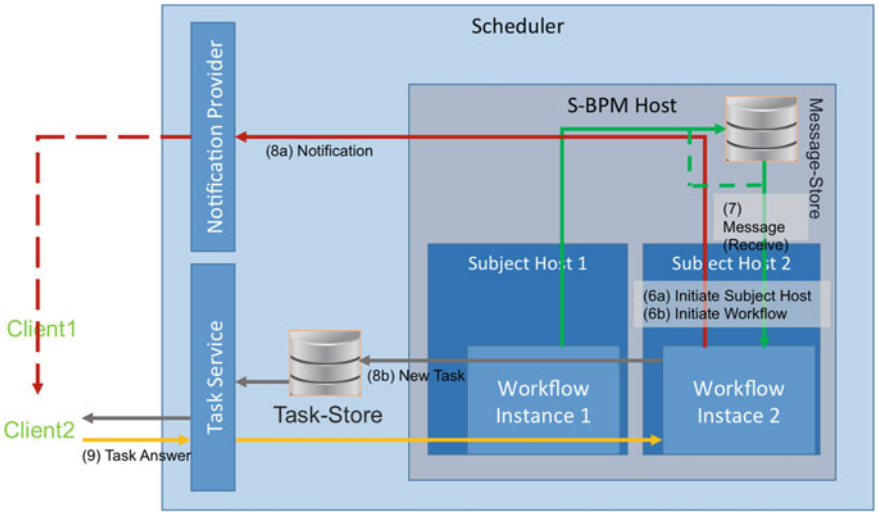

Fig. 14.9 Part 2: now, (*7*) the *Message* is forwarded to the next *Subject*, which means that it has to be instantiated first (*6a, b*), if needed. Then (*8a, b*) a *Task* is generated for interaction with an user via a client application (*9*). And so on

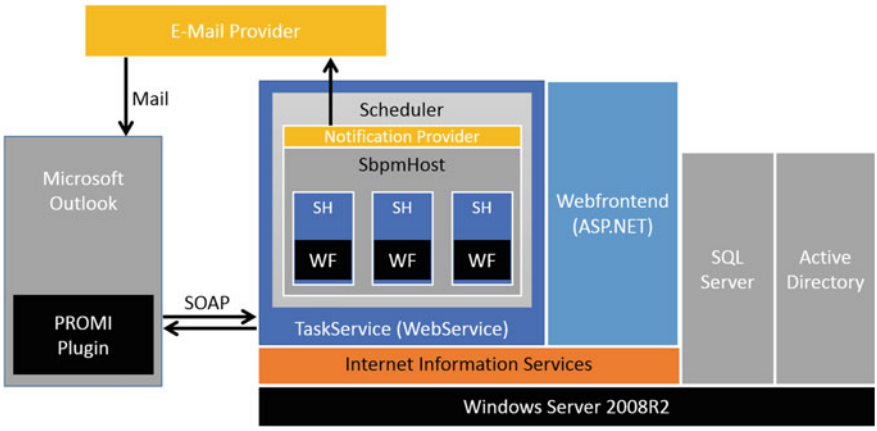

Fig. 14.10 Structured communication: overview of the S-BPM execution architecture

The corresponding prototypical application and architecture (named PROMI) is depicted in Fig. 14.10. Some components were not mentioned yet:

- A web-frontend can be used to answer *Tasks* via any web browser, if preferred (instead of using *MS Outlook*, or when using a mobile device).
- In this prototype we use *MS InfoPath* for more sophisticated data exchange types (business objects); so we can design rich forms to be sent to communication partners (basic forms can hold only basic data types).

As we have learned, *InfoPath* is not a very practical candidate as foundation for the exchange of forms; there are many restrictions (safety or documentation issues). It principally works, but we think that other solutions are needed to achieve the required ease of use, flexibility, performance and a cost-efficient implementation. *MS Outlook* is also not an easy candidate, because of programming restrictions.

S-BPM processes can be uploaded for execution (JPP or XAML format), all data (persisted instances, process models, etc.) is stored in a SQL database, a web server instance works as application host, role models from an active directory can be directly used to model the organization, and a mail server instance is used to handle message exchange between the subjects in the form of e-mails. The architecture resides on a server running *MS Windows Server 2008 R2 Datacenter* with *Hyper-V*; on this platform there are two virtual *MS Windows Server 2008 R2* servers running (one for active directory, DNS, IIS, SQL and the PROMI application, the other one hosting the *MS Windows Exchange* server). On clients we need *MS Outlook*, which uses a plugin to start S-BPM models (appears as a separate menu entry).

One very important aspect of using such a setting should not be neglected: integration of other software or hardware components; for example, we used *MS Dynamics NAV 2009* to demonstrate the integration of customer data via web service calls. Any software built on the *Microsoft* technology stack can be integrated without big hassle, as long as the interface is documented.

14.3.4 Moving into the Cloud

The PROMI architecture has some substantial limitations, thus we have to rethink some assumptions. Nevertheless, the core idea—to translate S-BPM models into WF workflows and use this as a foundation for an enterprise application to execute business processes—remains.

What do we need? To recap, we need an infrastructure which can be used by more than one company to define and execute integrated business processes crossing organizational boundaries. That means we have to create an architecture which does not run on only one company's server; from a technical point of view this means that processes running on the infrastructure of one company need to interact with processes running on the infrastructure of another company. Other requirements yet not or not fully considered:

- The platform needs to be **scalable**; that means it must be capable of handling processes with a small and a large number of instances and transactions per time frame.
- There must be a **security** concept which allows fine granular steering of user rights and visibility of business process models or instances, and access to data (business objects).

We believe that the only way to implement a *Multi-Enterprise Business Process Platform* is the use of an agent-based approach (in our case the S-BPM methodology) built on proper infrastructure. This can be for example a **public** or **private** cloud; the installation, running and managing of a cloud infrastructure as discussed in the following needs serious capabilities of an organization (money and people). We think that a public cloud has some beneficial features related to cost and as a foundation for new services and business models. We especially think, that a public cloud could have some advantages for SMEs. But there are also some drawbacks of a public cloud, as it needs additional efforts to integrate locally hosted applications with S-BPM processes hosted. If deep integration with other applications is needed, a local installation is preferred.

The whole new architecture is depicted in Fig. 14.12. Processes are hosted on an instance of the *Workflow Manager* (WFM), which is responsible for the hosting, administration and configuration of the subjects based on scopes (see Fig. 14.11), such as a *Company Scope* (1) for the processes of one organization, a *Process Scope* (2) for each process and a *Management Scope* (3). Each company has its own *Process Store* (4) and *Subject Store* (5); the same for *Message Store* (6) and *Task Store* (7). Each company has *Task Handler* (9) instances to generate new tasks and each process has *Message Handler* (8) instances to manage message exchange. *Task* and *Message Handler* are implemented as workflows itself. The mechanism of *Scopes* ensures full encapsulation of one company or organization by the other. Further, it allows rights management on a very fine granular basis for each activity; depending on the rights of a role, activities can be visible or not, and activities can be executed or not.

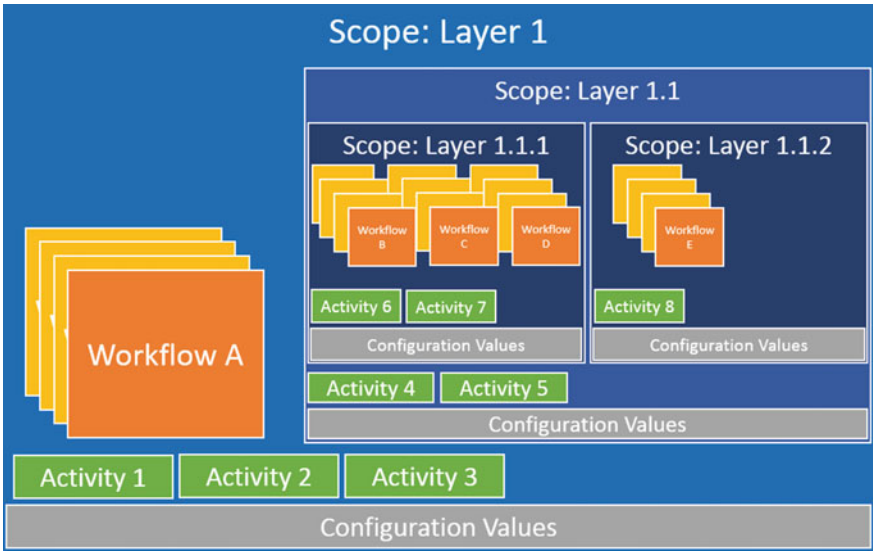

Fig. 14.11 Scopes are containers that may contain *Scopes*, *Activities*, workflow definitions, workflow instances and configuration settings

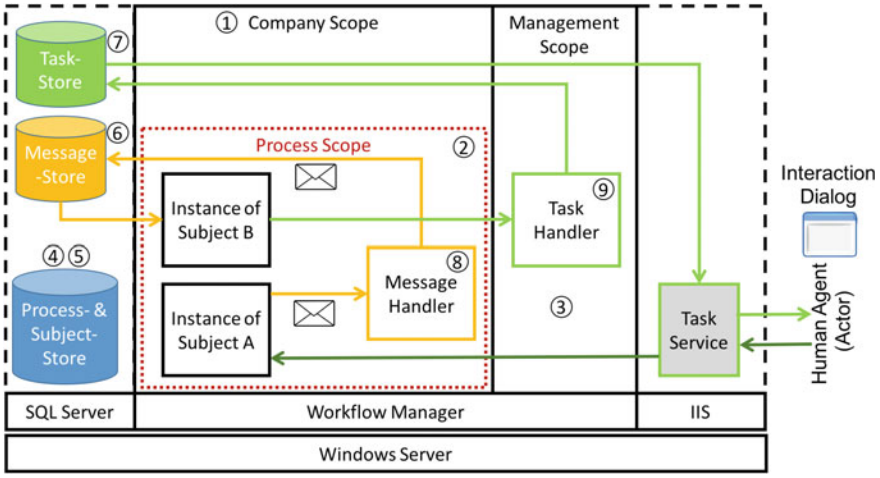

Fig. 14.12 StrICT architecture. The processes are executed server side and the workflows are coordinated through message exchange (*orange*). Task requests (*light green*) and task answers (*dark green*) are routed to a client via the task service

The new S-BPM architecture heavily uses fundamental functionality of the *MS Workflow Manager* (hosting of workflows) and the *MS Service Bus* (exchange of messages). The service bus provides relay and broker messaging functionalities that enable the exchange of messages between different services (see Fig. 14.13). It is

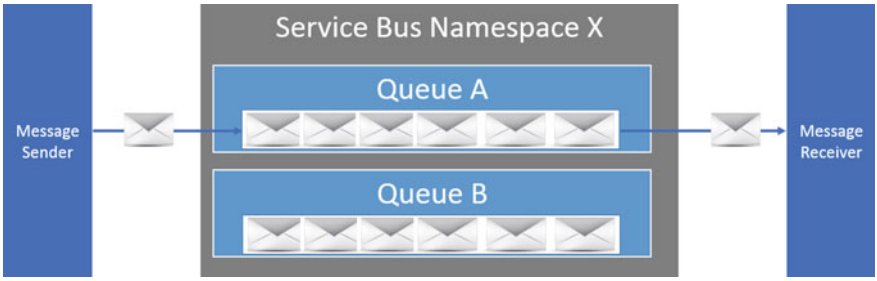

Fig. 14.13 Service bus namespace with service bus queues. Send messages to a transmitter queue, from which they are taken in the order received by the recipient

important to understand that the S-BPM architecture can be hosted on a server or a server farm (if scalability is needed) under the control of a company's IT department, or it can be hosted on a public cloud infrastructure provider such as *Microsoft Azure*. Any needed service (exchange, etc.) is available as a service in the *Azure Cloud*. A public cloud offers some additional possibilities for inter-company process execution, as will be explained later.

Communication between subjects—Messages to other subjects are routed via the internal *Service Bus* (part of *Workflow Manager*). The *Message Handler* is instantiated after receiving a message and forwards it to the correct input pool (*Message Store*) of the receiving subject instance; afterwards the instance is canceled. Subject instances have access to their own message pool and can choose any available message. Now, there is no central scheduler component any more; any subject conceptualizes an independent agent. This realizes an environment for distributed execution of concurrent business process tasks, which are synchronized via the exchange of messages. Messages are containers for data models, which means process actors exchange relevant business data (e.g., customer order, production order, invoice, …) via message exchange.

User interaction—Interaction with process participants is done via the *Task Service*. A *Task* is a request to be processed by a user, typically to fill in some data into a form (or anything else). A user has full access to its list of tasks. Tasks can be routed as regular e-mails to a user according to the role in an S-BPM process. A task can then be answered again using a standard e-mail protocol.

Figure 14.14 depicts the routing of messages via an external service bus. In this way messages can be routed from the IT infrastructure of one company to another one. This realizes an execution scenario of cross-company business processes. Of course, the processes need to have an agreed common name (technically we also have to send a Globally Unique Identifier (GUID) to identify the instance, so we know for which running instance the answer is) and a compatible and agreed interface.

The modeling of process collaborations can be a difficult task, as it is not an easy task to check whether a model can be executed without dead- or life-lock (because of the distributed and concurrent nature we have no execution path under central control). Principally, there are methods to accomplish that in an automatic or semi-

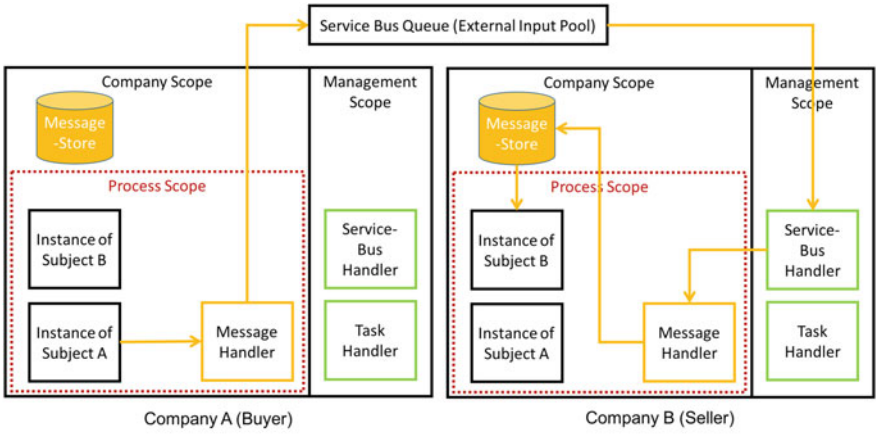

Fig. 14.14 Messages for external entities are passed to the input of the external subject

automatic way. As the processes are executable, they can be validated via simple execution (but, depending on the complexity eventually, not in a systematic way).

14.4 Results

14.4.1 Impact of Actions

Using the S-BPM methodology and the developed process execution platform it is an easy task to develop an enterprise application based on business process models. Now, we get an executable business process without coding; nevertheless, promising any complexity without coding is not credible. Especially if we need communication with other applications we will need to develop interfaces, or, if there are standard interfaces, we may need some coding to pack the data accordingly and make the correct calls writing some lines of code. Data, or business objects in general, typically call for technical skills for the design of the data models. Even if there is a tool to design nice forms for user interaction (generated by the *Task* service) knowledge about different data types is needed.

In the case of our example, in the beginning no highly sophisticated data model or extra lines of code are needed to demonstrate the functionality of the discussed and developed application. Nevertheless, a fully functional implementation of business processes needs interfaces to existing applications, such as an ERP or sales systems where customer data, purchase or manufacturing orders and other data is stored (depending on the needs of each collaboration participant). To design and discuss the process in collaboration between the interacting *Subjects*, simple data types can be used. If the process participants—we mean the people doing the process—have

developed a common understanding over the supply chain, IT can support the business process by connecting process activities with other systems; but only if needed.

Now, back to the introductory case. There are several involved subjects which coordinate work through the exchange of messages. For clarity we focus on the exchange of messages only and do not discuss the internal behavior of all subjects as it does not add any additional experiences. Following the modeling guidelines discussed by Fleischmann et al. (2012) we focus on the interface behavior, i.e., message exchange. If one is interested in similar scenarios as discussed here, we suggest also having a look at the book *S-BPM Illustrated* from Fleischmann et al. (2013).

For a compact visualization[7] of the process participants (*Subjects*) we use a BPMN conversation diagram as depicted in Fig. 14.15. This visualizes the case in a similar way as a *Subject Interaction Diagram* but without a detailed view of all *Messages*. It is interesting to see that the S-BPM concept to define business processes corresponds with a very similar representation defined in the BPMN standard; we see that the S-BPM concept is not something esoteric, but contrary to BPMN conversation diagrams S-BPM allows for a direct enactment of the modeled business processes.

The presented cloud implementation as discussed in the previous sections is capable of fully realizing an IT-based implementation of the case process. In Fig. 14.15 the subjects *Customer*, *Regional Logistics*, *BU Logistics* and *Factory Planning* represent cross-company process partners. Each of them has its own *Scope* in the cloud architecture, which represents its very own area to model and execute processes; no data can be interchanged between different scopes. It is also possible that any involved organization hosts its own copy of the architecture on separate hardware. Each organization models only its own processes and defines its communication partners as interface subjects.[8] Messages during execution are routed to the correct process partner (subject) as elaborated in Sect. 14.3.4 and depicted in Fig. 14.14. Further, the process execution is done via e-mail exchange (the client side). That means starting a process, interacting with tasks via forms and message exchange are done via e-mail (we developed a Microsoft Outlook plugin for this, as discussed in Sect. 14.3.3).

14.4.2 Open Issues

At the end of this practical case, some words about open technical issues.

Performance: we did not execute any performance tests, specifically performance depending on the number of running instances in total and of a process model. Even the architecture *should* be scalable, this has to be confirmed based on scientific and technical best-practice principles.

[7]The case discusses a so-called *Process Network* as described in Chap. 5 in Fleischmann et al. (2012).

[8]*Interface Subjects* regulate cooperation and facilitate the synchronization of process network partners (Fleischmann et al. 2012).

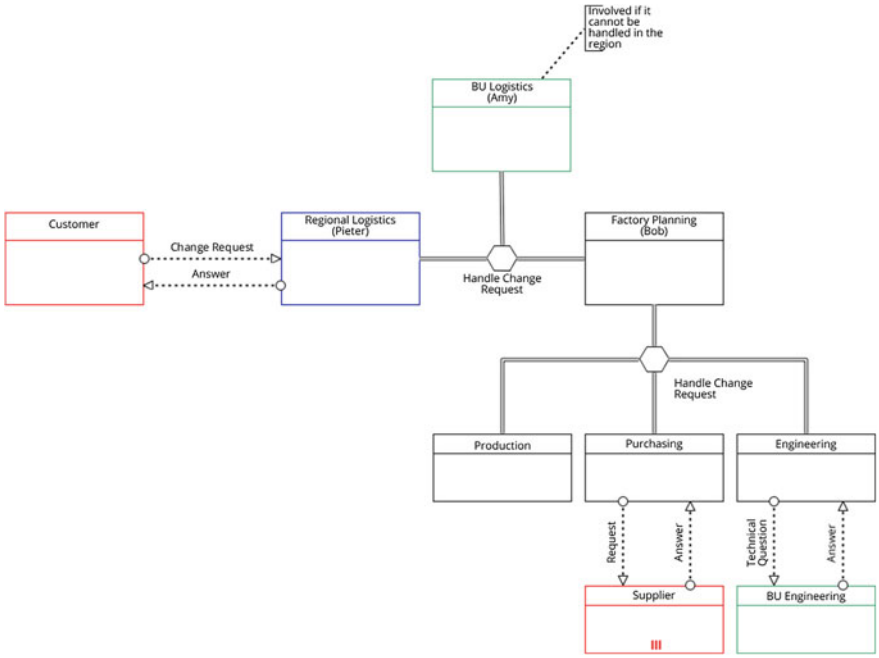

Fig. 14.15 The case visualized as BPMN conversation diagram. This representation cannot be directly converted into an S-BPM process representation and is therefore not directly executable

Modeling: at the time of writing, the modeling tools are not in a mature state; the development of models is done in a browser window based on *jsPlumb*[9]; the models are stored in the cloud infrastructure and can be uploaded for execution.

Business objects: there are two ways to work with data; messages can hold simple data types (numbers, text) or complex data types based on *JavaScript Object Notation* (JSON) data structures, a readable and compact data format designed for the exchange between applications. In this context there are some open technical questions to be solved.

Usability: as the application is intended to be used by *non-technical* people, heavy research towards usability has to be done. This includes questions about *semantic transparency* of the modeling language as discussed by Singer (2014), the design of data models and forms for user interaction.

[9] http://jsplumbtoolkit.com.

14.4.3 Takeaway

Practical work based on real-world problem settings has shown that all tools (applications) to build a multi-enterprise business process platform are available. In our case we have demonstrated that using the available server and programming tools from *Microsoft* it is possible without large effort to build an S-BPM platform based on cloud technology. The benefit and intention of choosing these platform products was to get an architecture for any size of business, even the largest. All used server applications are available as services on the *Microsoft Azure Cloud* and are therefore highly scaleable. The drawback of the presented approach is that it may be a too "fat" (i.e., not lean) solution for small and medium enterprises. Especially, if there are only simple processes based on some few subjects, the platform could be too expensive based on cost per transaction.

But we also have learned that execution is not everything—there is also a great need for modeling tools, not only for process models, but also to design data structures and forms. The execution platform is not visible to any stakeholder, but only to IT staff. Nevertheless, for daily and practical use also well-designed interfaces for using a business process system are needed with plenty of functionality; for example, users want and need to *search* for transactions (instances) and the related data. A very good approach is the idea of exchanging messages via e-mails. This leads to a mix of structured (the processes) and unstructured communication with high acceptance by the involved users.

References

Butcher P (2014) Seven concurrency models in seven weeks. The Pragmatic Programmers. LCC, www.pragprog.com

Davenport TH (2010) Process management for knowledge work. In: vom Brocke J, Rosemann M (eds) Handbook on business process management, vol 1. Springer, Berlin

Dixon J (2012) Hype cycle for business process management. Gartner, Stamford

Fleischmann A, Schmidt W, Stary Ch, Obermeier S, Börger E (2012) Subject-oriented business process management. Springer, Berlin

Fleischmann A, Raß S, Singer R (2013) S-BPM illustrated. Springer, Berlin

Singer R (2014) User centered development of agent-based business process models and notations. arXiv:1404.2737

Singer R, Kotremba J, Raß S (2014) Modeling and execution of multi-enterprise business processes. In: 16th IEEE conference on business informatics, workshop on cross-organizational and cross-company BPM (XOC-BPM), Genf, Switzerland, July 14–17