

Modeling Processes in a Subject-Oriented Way

5

5.1 To Go

Our processes are sometimes quite complicated. I would like for this high complexity to actually get implemented, and for everything to nevertheless remain well structured. As simple as possible, but not simpler, as Albert Einstein once said.



If we want to solve complex problems, we need to look for proper structuring possibilities. The best case would be to have just a handful of symbols to describe the process structure and the dynamics. It would also help if we could use a small number of additional complementary symbols to represent constantly recurring behavior sequences in a compact and structured way.



Everybody should be able to understand the basic constructs and these should be based on what we do. What are we really doing? We communicate with colleagues to exchange information and synchronize tasks, and we use the information or things which we receive from others to do something.



Yes, this is exactly the type of reality that we find in processes. We can capture this perceived reality through models. Such process models describe who is communicating with whom, and how stakeholders interact.



In the following, we will discuss the S-BPM bundle of modeling activities in detail.

As the distinction between design time and runtime of models is essential to the understanding of modeling, we first distinguish between models and instances. Then, we explain what role S-BPM stakeholders play in the course of modeling. Subsequently, the individual modeling constructs are described. We distinguish here between basic and extension constructs.

Using basic constructs, processes can be described completely from a subject-oriented perspective. However, for the compact and concise representation of complex affairs of humanly perceived reality, the subject-oriented method has been extended with corresponding constructs. These allow a much shorter and more transparent representation (notation) of certain constellations in processes. These constructs are not fundamental extensions enriching the expressiveness of the S-BPM specification language, but rather merely a means of simplifying the notation to handle complex cases, as each extension can be expressed completely using the basic constructs. The extension constructs result from practical experiences with the subject-oriented approach. While continuing S-BPM practice, it may be useful to add other constructs as well. However, such extensions have always to be traceable to basic constructs.

5.2 Process Models and Process Instances

In business process management, there is a distinction between process models and process instances. Subject-oriented process models describe the behavior of parties involved in business transactions, in particular, which activities are performed by whom to yield a result of value. Such models represent generalized situations, in particular, of how a business transaction is managed and which tasks need to be accomplished. Subjects are abstract resources, which represent active agents in a process.

For instance, a process model describing the request for a business trip contains the subjects involved, what the people responsible for those subjects do and in what order, and how they communicate to achieve a result.

However, a process instance is a concrete occurrence of the process described by the model. It is created when a transaction is actually triggered. For example, a process instance is initiated in the case of the business trip application when an employee, e.g., Mr. Schulz, submits a respective request.

Process instances contain concrete data: actors, activities, and affected business objects, as well as messages that are exchanged between actors for accomplishing a task. All of these are described in abstract form in process models.

A process model is created independently of specific organizational units or actors. Similarly, the model is independent of the tools or application programs that are available to execute the process. Thus, a business trip application may be submitted by any employee of an organization. The activities to be carried out are usually the same for all: they are performed in the roles/subjects “employee” (applicant), “manager” (approver), and “travel office” (clerk). In addition, different IT support for the same process could be used. A central organization could manage business trip requests with an SAP system, while in remote offices homegrown applications could be used.

A process model is therefore implemented on the one hand several times in the organization and on the other hand possibly in different computing environments. Although this complicates the aim of many organizations to achieve standardization and homogenization, it corresponds to reality, since heterogeneous organizational and system landscapes, which have either grown historically or are the result of corporate mergers often, have to be taken into consideration. A process model should therefore be largely independent of these environmental conditions.

The initiation of process instances can be done in different ways. In a first variant, a user creates an instance by interacting with an IT system. For example, employee Schulz creates a business trip request because he needs to visit a client. This process instance is executed in accordance with the process model and with the help of the specific people and respective tools assigned while embedding the process model into the organizational and IT environment. A second variant is the instantiation according to time constraints. For example, every Thursday a business trip request is automatically generated for a regular meeting in the branch office. A third possibility is the instantiation as a result of certain constellations of data. For instance, if the negative account balance of a checking account exceeds the associated overdraft line of credit for this account, an appropriate handling process is instantiated. Or in another example, the trigger could be a certain stock price: if the value falls below a certain mark and a bank customer is assigned to a certain risk class, a process is automatically initiated to respond to this situation. This is realized by a so-called complex event processing system.

In the following, we introduce the S-BPM-conform description of models. In subsequent chapters, we discuss the embedding into the organizational and IT environment of an organization, as well as the formation and execution of process instances of models.

5.3 Modeling Procedure

A subject-oriented process model describes, in contrast to existing approaches to BPM modeling, business processes primarily from the perspective of communicating actors or systems. It captures which tasks of a business process have to be

performed by whom using which tools, what result is produced in doing so, and for whom the result is intended.

A process model is considered a basic pattern that enables generating process instances for specific situations. A model of the process “business trip application” captures how the process basically works, while an instance of the process, e.g., Mr. Schulz’s application for a business trip, reflects the actual execution of his specific trip request, pursuant to the process model.

When modeling according to the subject-oriented approach, subjects are in the center of attention. They represent participating actors in a process. The modeling procedure essentially is a sequence of the following steps in which the associated level of detail increases moving forward:

- Identification of processes in an organization: The result is a process map with the processes and their mutual relationships.
- Specification of the communication structure: Based on the identification of the subjects and their interactions, in this step the communication structure of a business process is specified including the messages exchanged between the subjects.
- Specification of the behavior of the subjects involved in the process: The steps for accomplishing individual tasks of the subjects and the rules to follow thereby are specified.
- Description of the information all subjects involved in the process edit locally and mutually exchange via messages.

Actually, an organization is an ongoing process, a continuous chain of communication, regardless of whether both partners are coordinated in time or not (i.e., interacting synchronously or asynchronously).

Since the identification of processes and their constituent elements have already been discussed in the context of subject-oriented process analysis (see Chap. 4), we will detail the procedure from step 2 onwards in the following. The model constructs used for modeling are exemplified in the process “business trip application” of an organization.

This chapter reveals the fundamental constructs of S-BPM, namely subjects, their interactions via messages, their behavior, and the business objects they handle and exchange via messages. For each of the constructs, a diagrammatic symbol is available. This set of basic constructs is sufficient to model settings observed in perceived reality.

5.4 S-BPM Modeling Stakeholders

5.4.1 To Go

May I remind you that every endeavor requires specific roles. Hence, I see it as my duty to make you familiar with the various hats stakeholders are wearing when performing S-BPM activities. The roles help to clarify responsibilities. Each person is assigned to a role or receives a corresponding sticker.



An old story, in the end there is chaos anyway.



Thanks to de Bono, we know that four hats are sufficient for a balanced procedure.



Alongside myself as Governor, the Facilitator will interact with Experts and Actors in S-BPM.



I have my doubts whether the job will be easier then.





... I wish it would be that easy for me!

Yes and No, on the one hand, all stakeholders wear only one hat at a certain point in time. But when they try to simultaneously wear more than one hat, we need intervention and clarification. This actually depends on the state of affairs as reflected by the Facilitator. This explains his function quite well.



In principle, it is that easy: Actors are the key figures. Their communication behavior determines the success of the business of an organization. Without them, an S-BPM project cannot be legitimized.



If we need special methods to represent knowledge, or rather require domain-specific information, experts can help us. With their cooperation, we can pursue the goal of an organization following S-BPM.



Various stakeholders are involved with different levels of intensity in the activity bundle “modeling”, as already indicated in the previous section. In the following, we detail their tasks along the various activities.

5.4.2 Governors

The Governors (drivers and managers) determine the constraints for a process, and thus the rules for creating and maintaining process models. The Governors determine above all the process scope stakeholders need to consider in the project, and which methodology and tools they should use.

Specifications of the scope for modeling include process boundaries, i.e., how a process (domain) is distinct to others, and the representational structure, namely in which subprocesses a process should be decomposed. In addition, it should be specified which results from a previous activity bundle (e.g., analysis or

monitoring) should be mainly addressed when revising or rebuilding a model. This ultimately represents a prioritization.

Finally, a Governor decides, whether and when a model is complete and should be passed on to the activity bundle “validation” or another one.

Consequently, Governors set standards for different aspects of modeling. Thus, for the scope, depending on the importance of the process, top management or middle management takes responsibility, while the method and tool guidelines often stem from the Organization Department. Affected stakeholders traditionally encounter standards set by other bodies with mixed feelings, and with different, often insufficient levels of acceptance. Therefore, in particular with regard to rules which have been defined by the executive board level, but for which this in itself is not enough to grant them a strong binding effect, at least the moral support from top management is required to increase acceptance.

5.4.3 Actors

The Actors (work performers) are the active agents in the process. They can either be people in specific roles or machines that perform the individual actions in the respective processes and process instances. Process descriptions are essential for Actors because they indicate their behavior in the process or in its sub processes, i.e., what activities they perform and when.

S-BPM enables the Actors to create this description, within the guidelines specified by the Governor, themselves, and thus to actively design the development of the respective processes. Since, in principle, each employee of an organization is involved in at least one process as an Actor, this holds for every member of the organization. The behavioral specification for an Actor in a process corresponds to his subject description. Hence, in modeling, all directly and indirectly involved stakeholders, representing the process as such, have to be incorporated. They usually know well, what they have to do in a process, when they have to do it and in what order, and also how they can perform their work tasks effectively and efficiently. The Actors also know with whom they need to communicate during the execution of a process instance, and what data they need to exchange to enable a smooth process flow.

If necessary, the Actors or the Facilitator ask Experts to assist in coordination and modeling.

No Actors—no process description. S-BPM models should be semantically correct—models should reflect the work for each stakeholder in a coherent way.

5.4.4 Experts

An Expert (specialist) supports the Governor, Actor, and Facilitator with methodological and technical knowledge (see Sect. 9.4.3). Experts are consulted on specific technical or functional issues to introduce effective and efficient solutions. By selecting and using appropriate methods, they can help to find solutions to problems.

Experts can assist Governors in the formulation of modeling requirements, such as convention manuals. On request of a Facilitator, Experts can also perform method and tool trainings to qualify Actors.

Experts can also help Actors with modeling of processes, for instance by using reference process models. In such cases, the experience of an Expert can help to describe specific task sequences in a transparent and understandable way and to ensure compliant modeling.

The addressed expert competence in modeling and tool handling is often concentrated in the organization department of an organization.

Finally, the implementation of processes or parts of processes often requires the help of IT Experts.

5.4.5 Facilitators

Facilitators (guide during development) coordinate the various tasks within the activity bundle of modeling. This means they manage the communication between the Governor, Actors, and the Experts. They ensure that the Governor provides the required modeling guidelines in time, and that all Actors understand them.

If necessary, the Facilitator identifies the appropriate Experts for specific tasks and then puts in a request for their support, e.g., a tool specialist might be requested for solving a problem with the modeling tool.

The Facilitator ensures that the Actors' communicate with their colleagues and that they coordinate their activities in the course of modeling. The Facilitator also checks repeatedly by himself, or with the help of Experts, whether a model meets the requirements of the Governor, and whether the requirements resulting from a previous activity bundle are incorporated. Ambiguities are clarified together with responsible Governors and involved Actors.

Together with the Governors of the organizational development, the Facilitator guides the transition from modeling to validation, and thus initiates the subsequent activity bundle. Facilitators mostly belong to the organization department or the middle management and have temporarily taken on the function of a project manager for a process change project. They may be responsible for a complete process change or be appointed only for a particular activity, such as the modeling bundle. In this case, the role of a person as Facilitator is completed with the transition to validation. Such a scenario is especially common in modeling because here the Facilitator is often also the Expert for the modeling methodology.

S-BPM managers should signal from the very beginning to their coworkers their desire to communicate, point out to them the objectives of change processes, and inform them in the course of development of each step.

5.5 Basic Constructs of Subject-Oriented Modeling

5.5.1 To Go

I regularly receive requests to deliver data or to collaborate in modeling of processes. What does this mean?



What is the goal of the project?



The Governor has started a project.



The objectives have been defined by the Governor. We want to improve some of our processes.

We want to check whether our existing business processes correspond to our actual work tasks, which should help us to improve our processes in a targeted way.



Not before we have clarified to what extent and how to proceed. Actors should have a clear understanding of their involvement in processes.



And you want me to contribute?



5.5.2 Subject

In the simple scenario of the business trip application, we can identify three subjects, namely the employee as applicant, the manager as the approver, and the travel office as the travel arranger.

The definition of which subjects should be part of a process is a leadership decision—this is why the Governor needs to be involved. On the one hand, the necessary subjects result from the actual (as-is) situation, as it has for example already been described in the process analysis. On the other hand, the subject scoping, i.e., the question of what subjects there are and what tasks they roughly perform, can be adjusted to the envisioned or desired (to-be) situation.

Depending on the required or desired division of labor in a process, a corresponding number of subjects are necessary. This division is a design decision that must be taken in accordance with business needs. It influences the necessary granularity of a process model (see Sect. 5.5.6).

In case there are many specialized subjects involved in a process, it may lead to many potentially complex interactions between the subjects. This can be a problem, since the communication between process participants always carries the risk of delays and misunderstandings. In case of few subjects, however, the subject carriers often cover a too wide a range of activities, which puts high demands on the

participants. The decision with respect to subject scoping therefore has far-reaching consequences. It is complex, represents a major challenge, and requires extensive experience and care.

5.5.3 Subject-to-Subject Communication

After the identification of subjects involved in the process (as process-specific roles), their interaction relationships need to be represented. These are the messages exchanged between the subjects. Such messages might contain structured information—so-called business objects (see Sect. 5.5.7).

The result is a model structured according to subjects with explicit communication relationships, which is referred to as a Subject Interaction Diagram (SID) or, synonymously, as a Communication Structure Diagram (CSD) (see Fig. 5.1).

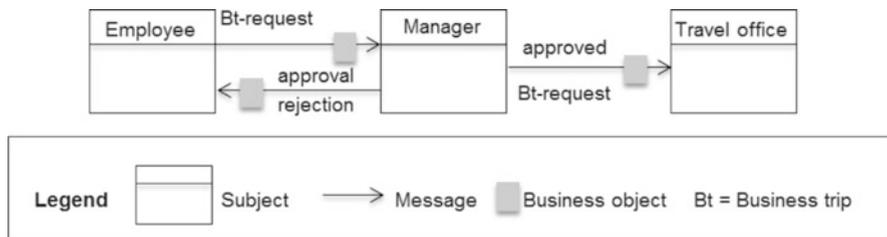


Fig. 5.1 Subject interaction diagram for the process “business trip application”

Messages represent the interactions of the subjects during the execution of the process. We recommend naming these messages in such a way that they can be immediately understood and also reflect the meaning of each particular message for the process. In the sample “business trip application”, therefore, the messages are referred to as “business trip request”, “rejection”, and “approval”.

Messages serve as a container for the information transmitted from a sending to a receiving subject. There are two options for the message content:

- **Simple data types:** Simple data types are string, integer, character, etc. In the business trip application example, the message “business trip request” can contain several data elements of type string (e.g., destination, reason for traveling, etc.) and of type number (e.g., duration of trip in days).
- **Business Objects:** Business Objects in their general form are physical and logical “things” that are required to process business transactions. We consider data structures composed of elementary data types, or even other data structures, as logical business objects in business processes. For instance, the business object “business trip request” could consist of the data structures “data on applicants”, “travel data”, and “approval data”—with each of these in turn containing multiple data elements.

5.5.4 Synchronization of the Technical Message Exchange

In the previous section, we have stated that messages are transferred between subjects and have described the nature of these messages. What is still missing is a detailed description of how messages can be exchanged, how the information they carry can be transmitted, and how subjects can be synchronized. These issues are addressed in the following subsections.

5.5.4.1 Synchronous and Asynchronous Exchange of Messages

In the case of synchronous exchange of messages, sender and receiver wait for each other until a message can be passed on. If a subject wants to send a message and the receiver (subject) is not yet in a corresponding receive state, the sender waits until the receiver is able to accept this message. Conversely, a recipient has to wait for a desired message until it is made available by the sender.

The disadvantage of the synchronous method is thus a close temporal coupling between sender and receiver. This raises problems in the implementation of business processes in the form of workflows, especially across organizational borders. As a rule, these also represent system boundaries across which a tight coupling between sender and receiver is usually very costly. For long-running processes, sender and receiver may wait for days, or even weeks, for each other.

Using asynchronous messaging, a sender is able to send anytime. The subject puts a message into a message buffer from which it is picked up by the receiver. However, the recipient sees, for example, only the oldest message in the buffer and can only accept this particular one. If it is not the desired message, the receiver is blocked, even though the message may already be in the buffer, but in a buffer space that is not visible to the receiver. To avoid this, the recipient has the alternative to take all of the messages from the buffer and manage them by himself. In this way, the receiver can identify the appropriate message and process it as soon as he needs it. In asynchronous messaging, sender and receiver are only loosely coupled. Practical problems can arise due to the in reality limited physical size of the receive buffer, which does not allow an unlimited number of messages to be recorded. Once the physical boundary of the buffer has been reached due to high occupancy, this may lead to unpredictable behavior of workflows derived from a business process specification. To avoid this, the input pool concept has been developed for S-BPM (see Sect. 5.5.5.2).

A typical example of a message exchange is the business trip as a business transaction. It is triggered by an event such as a scheduled customer visit. The application for the business trip can take place far in advance of the actual commencement of the journey. Before this, a hotel needs to be booked and travel arrangements need to be made—processes that can run in parallel or interlocked.

Once the trip starts, the process has not yet been completed. Billing and application for reimbursement may still need to be requested. A permanent synchronization of all the steps is not only expensive but usually not necessary

because a coherent processing scheme for business trips can be derived according to the causality given in the business process specification “business trip”. This represents an ideal scenario for an asynchronous message exchange.

5.5.4.2 Exchange of Messages via the Input Pool

To solve the problems outlined in asynchronous message exchange, the input pool concept has been developed. Communication via the input pool is considerably more complex than previously shown; however, it allows transmitting an unlimited number of messages simultaneously. Due to its high practical importance, it is considered as a basic construct of S-BPM.

Consider the input pool as a mail box of work performers, the operation of which is specified in detail.

Each subject has its own input pool. It serves as a message buffer to temporarily store messages received by the subject, independent of the sending communication partner. The input pools are therefore inboxes for flexible configuration of the message exchange between the subjects. In contrast to the buffer in which only the front message can be seen and accepted, the pool solution enables picking up (= removing from the buffer) any message. For a subject, all messages in its input pool are visible.

The input pool has the following configuration parameters (see Fig. 5.2):

- **Input pool size:** The input pool size specifies how many messages can be stored in an input pool, regardless of the number and complexity of the message parameters transmitted with a message. If the input pool size is set to zero, messages can only be exchanged synchronously.
- **Maximum number of messages from specific subjects:** For an input pool, it can be determined how many messages received from a particular subject may be stored simultaneously in the input pool. Again, a value of zero means that messages can only be accepted synchronously.
- **Maximum number of messages with specific identifiers:** For an input pool, it can be determined how many messages of a specifically identified message type (e.g., invoice) may be stored simultaneously in the input pool, regardless of what subject they originate from. A specified size of zero allows only for synchronous message reception.
- **Maximum number of messages with specific identifiers of certain subjects:** For an input pool, it can be determined how many messages of a specific identifier of a particular subject may be stored simultaneously in the input pool. The meaning of the zero value is analogous to the other cases.

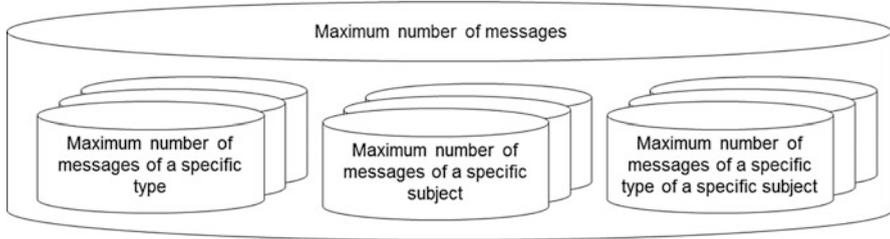


Fig. 5.2 Configuration of input pool by parameters

By limiting the size of the input pool, its ability to store messages may be blocked at a certain point in time during process runtime. Hence, messaging synchronization mechanisms need to control the assignment of messages to the input pool. Essentially, there are three strategies to handle the access to input pools:

- Blocking the sender until the input pool's ability to store messages has been reinstated: Once all slots are occupied in an input pool, the sender is blocked until the receiving subject picks up a message (i.e., a message is removed from the input pool). This creates space for a new message. In case several subjects want to put a message into a fully occupied input pool, the subject that has been waiting longest for an empty slot is allowed to send. The procedure is analogous if corresponding input pool parameters do not allow storing the message in the input pool, i.e., if the corresponding number of messages of the same name or from the same subject has been put into the input pool.
- Delete and release of the oldest message: In case all the slots are already occupied in the input pool of the subject addressed, the oldest message is overwritten with the new message.
- Delete and release of the latest message: The latest message is deleted from the input pool to allow depositing of the newly incoming message. If all the positions in the input pool of the addressed subject are taken, the latest message in the input pool is overwritten with the new message. This strategy applies analogously when the maximum number of messages in the input pool has been reached, either with respect to sender or message type.

5.5.4.3 Sending Messages

Before sending a message, the values of the parameters to be transmitted need to be determined. In case the message parameters are simple data types, the required values are taken from local variables or business objects of the sending subject, respectively. In case of business objects, a current instance of a business object is transferred as a message parameter.

The send process attempts to send the message to the target subject and store it in its input pool. Depending on the described configuration and status of the input pool, the message is either immediately stored or the sending subject is blocked until a delivery of the message is possible.

In the sample business trip application, employees send completed requests using the message "send business trip request" to the manager's input pool. From

a send state, several messages can be sent as an alternative. The following example shows a send state in which the message M1 is sent to the subject S1, or alternatively the message M2 is sent to S2, therefore referred to as alternative sending (see Fig. 5.3). It does not matter which message is attempted to be sent first. If the send mechanism is successful, the corresponding state transition is executed. In case the message cannot be stored in the input pool of the target subject, sending is interrupted automatically, and another designated message is attempted to be sent. A sending subject will thus only be blocked if it cannot send any of the provided messages.

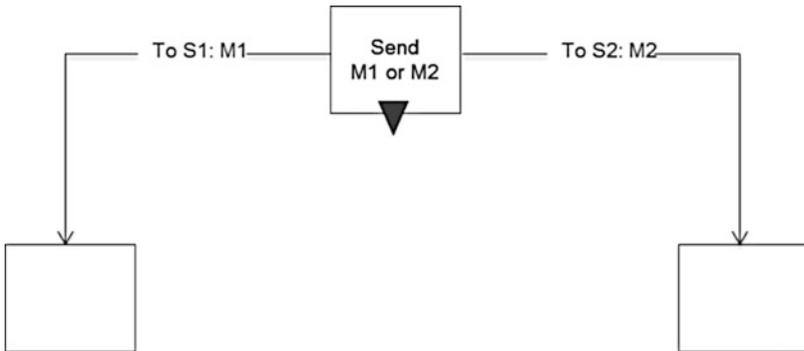


Fig. 5.3 Example of alternative sending

By specifying priorities, the order of sending can be influenced. For example, it can be determined that the message M1 to S1 has a higher priority than the message M2 to S2. Using this specification, the sending subject starts with sending message M1 to S1 and then tries only in case of failure to send message M2 to S2. In case message M2 can also not be sent to the subject S2, the attempts to send start from the beginning.

The blocking of subjects when attempting to send can be monitored over time with the so-called timeout. The example in Fig. 5.4 shows with “Timeout: 24 h” an additional state transition, which occurs when within 24 h one of the two messages cannot be sent. If a value of zero is specified for the timeout, the process immediately follows the timeout path when the alternative message delivery fails completely.

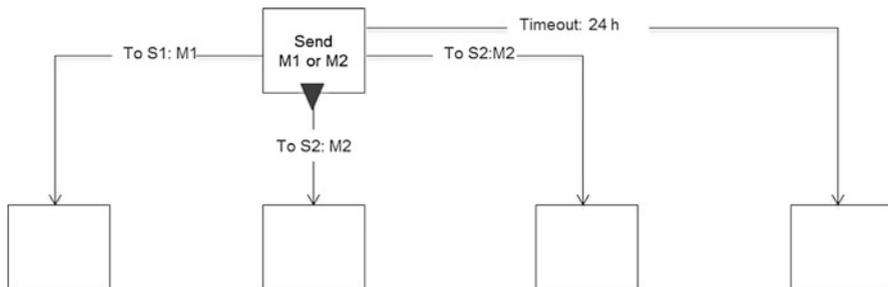


Fig. 5.4 Send using time monitoring

5.5.4.4 Receiving Messages

Analogously to sending, the receiving procedure is divided into two phases, which run inversely to send.

The first step is to verify whether the expected message is ready for being picked up. In case of synchronous messaging, it is checked whether the sending subject offers the message. In the asynchronous version, it is checked whether the message has already been stored in the input pool. If the expected message is accessible in either form, it is accepted, and in a second step, the corresponding state transition is performed. This leads to a takeover of the message parameters of the accepted message to local variables or business objects of the receiving subject. In case the expected message is not ready, the receiving subject is blocked until the message arrives and can be accepted.

In a certain state, a subject can expect alternatively multiple messages. In this case, it is checked whether any of these messages is available and can be accepted. The test sequence is arbitrary, unless message priorities are defined. In this case, an available message with the highest priority is accepted. However, all other messages remain available (e.g., in the input pool) and can be accepted in other receive states.

Figure 5.5 shows a receive state of the subject “employee” which is waiting for the answer regarding a business trip request. The answer may be an approval or a rejection.

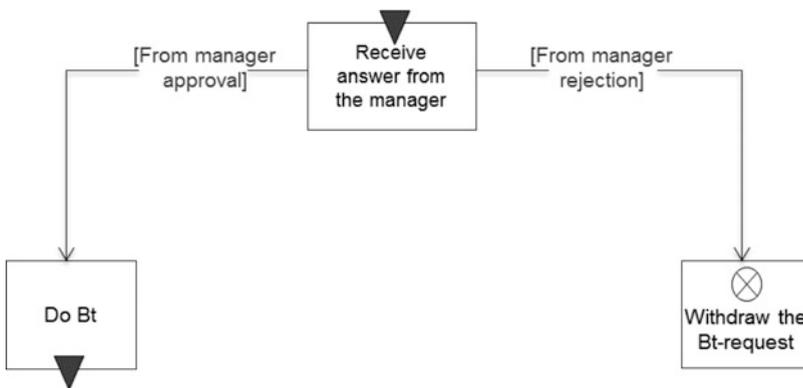


Fig. 5.5 Example of alternative receiving

Just as with sending messages, also receiving messages can be monitored over time. If none of the expected messages are available and the receiving subject is therefore blocked, a time limit can be specified for blocking. After the specified time has elapsed, the subject will execute the transition as it is defined for the timeout period. The duration of the time limit may also be dynamic, in the sense that at the end of a process instance the process stakeholders assigned to the subject decide that the appropriate transition should be performed. We then speak of a manual timeout.

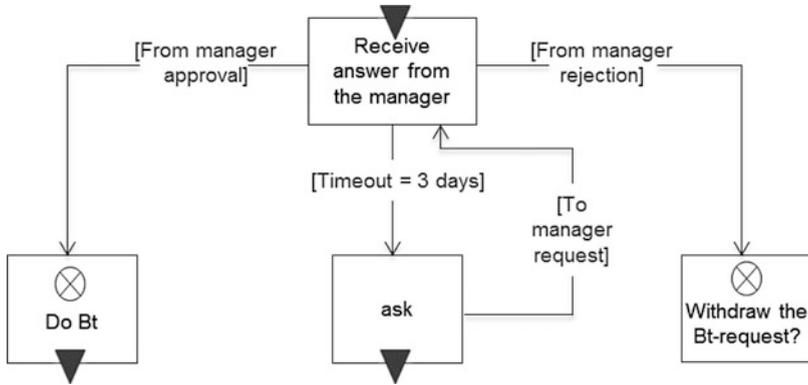


Fig. 5.6 Time monitoring for message reception

Figure 5.6 shows that, after waiting 3 days for the manager’s answer, the employee sends a corresponding request.

Instead of waiting for a message for a certain predetermined period of time, the waiting can be interrupted by a subject at all times. In this case, a reason for abortion can be appended to the keyword “breakup”. In the example shown in Fig. 5.7, the receive state is left due to the impatience of the subject.

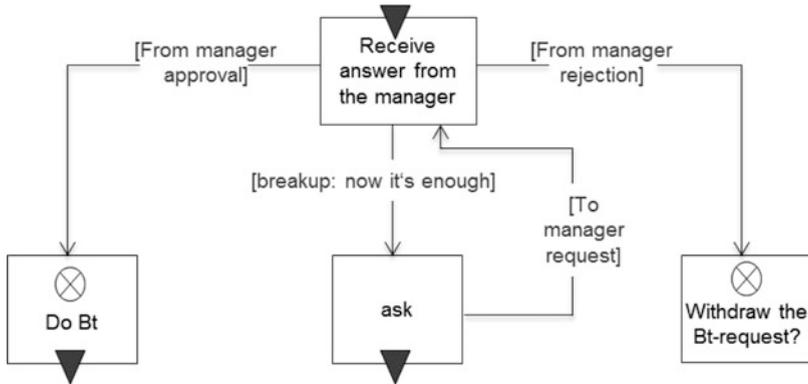
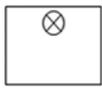


Fig. 5.7 Message reception with manual interrupt

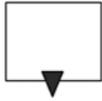
5.5.5 Subject Behavior

The possible sequences of a subject’s actions in a process are termed subject behavior. States and state transitions describe what actions a subject performs and how they are interdependent. In addition to the communication for sending and receiving, a subject also performs so-called internal actions or functions.

States of a subject are therefore distinct: there are actions on the one hand, and communication states to interact with other subjects (receive and send) on the other hand. This results in three different types of states of a subject:



Performing functions (function state)



Sending messages (send state)



Receiving messages (receive state)

In S-BPM, work performers are equipped with elementary tasks to model their work procedures: sending and receiving messages and immediate accomplishment of a task (function state).

In case an action associated with a state (send, receive, and do) is possible, it will be executed, and a state transition to the next state occurs. The transition is characterized through the result of the action of the state under consideration: For a send state, it is determined by the state transition to which subject what information is sent. For a receive state, it becomes evident in this way from what subject it receives which information. For a function state, the state transition describes the result of the action, e.g., that the change of a business object was successful or could not be executed.

The behavior of subjects is represented by modelers using Subject Behavior Diagrams (SBD). Figure 5.8 shows the subject behavior diagram depicting the behavior of the subjects “employee”, “manager”, and “travel office”, including the associated states and state transitions.

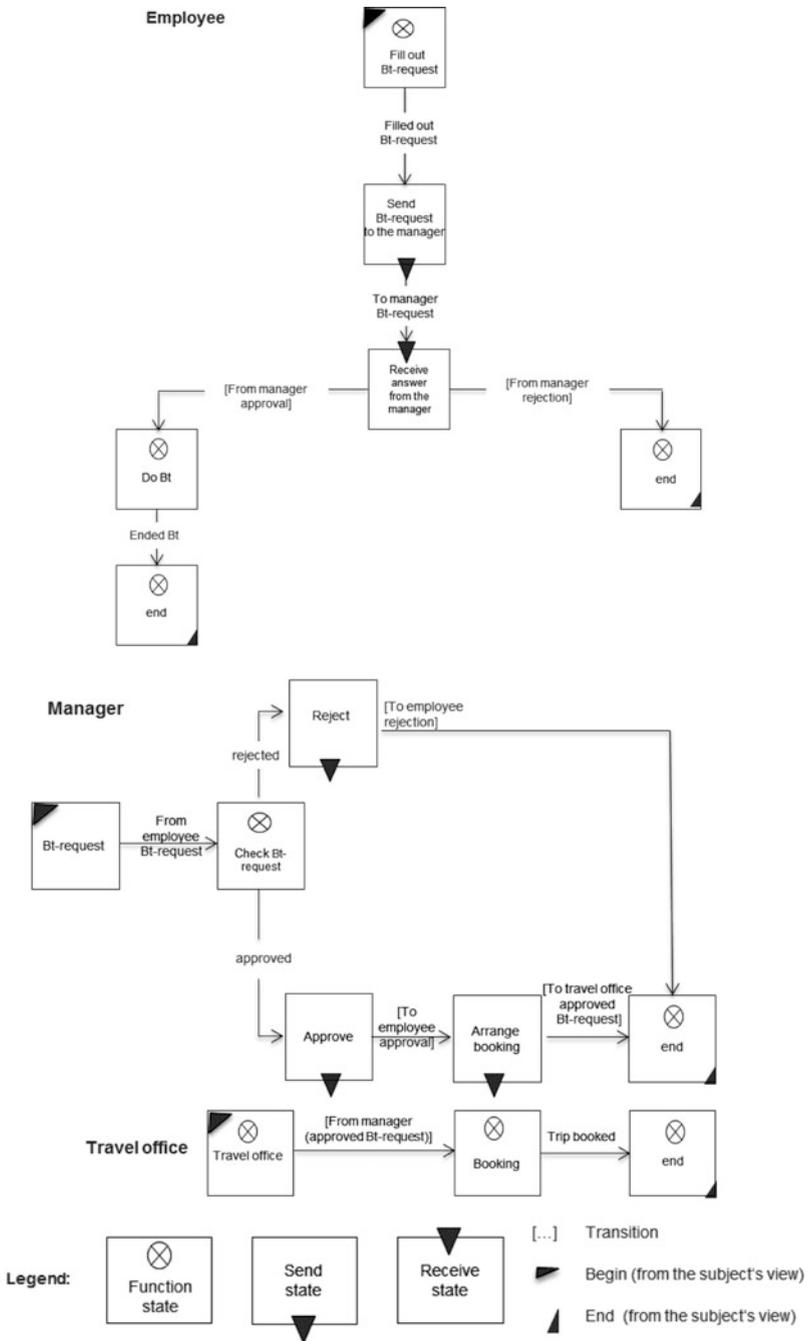


Fig. 5.8 Subject behavior diagram for the subjects “employee”, “manager”, and “travel office”

5.5.6 Normalization

The default behavior of a subject is represented by its action behavior (performing functions) and communication behavior (sending and receiving messages).

Action behavior can in principle contain many internal functions to be performed in sequence, in order to capture the individual work steps of a subject. In these sequences of internal functions, no sending and receiving nodes are included. This is crucial as work regulations for individuals or roles representing a subject but can lead to extensive and therefore confusing behavior diagrams. Moreover, these sequences of internal functions are not important for communication, and therefore, not relevant for the communication partners.

To simplify the presentation, we can use the fact that neighboring subjects, which interact during the course of process execution with the subject momentarily under consideration, and the behavior of which is currently being described, are mainly interested in the communication behavior of this subject (Do I get the desired result?) and less in its action behavior. The action behavior is of interest only insofar as it affects the communication behavior. Given this background, we can define a so-called normalized behavior, merging a sequence of functions into a larger function. By hiding functional details, the subject behavior, from the perspective of neighboring subjects, becomes much more transparent, without having to change the, for those neighboring subjects so important, description of the communication behavior.

Figure 5.9 shows in the upper half the detailed behavioral representation for the subject “employee”, as it is given as a work requirement for the affected employees. In the bottom half of the figure, the two actions “withdraw business trip request” and “change business trip request” (with a double-lined border) were combined into a larger action.

For a normalized behavior, in principle, any function states between their encompassing send and receive states can be combined to form other ones that remain visible to their neighboring subjects. Exceptions are end states. Consequently, it is not possible in the example to group the functions “do business trip” and “end”. This normalized behavior also provides indications for the level of detail of a process model.

An important issue in BPM projects is the question of the level of detail needed to describe the steps of a process. This issue was already addressed in the chapter on analysis (see Sect. 4.4). The normalization of subject-oriented modeling is a suitable tool to determine that normalized behavior is sufficient for complete representations.

This construct allows solving the problems identified for finding proper granularity using either a top-down or bottom-up approach. The appropriate level of granularity in modeling can be determined, once it is known which subjects are involved and what tasks they will perform in a process.

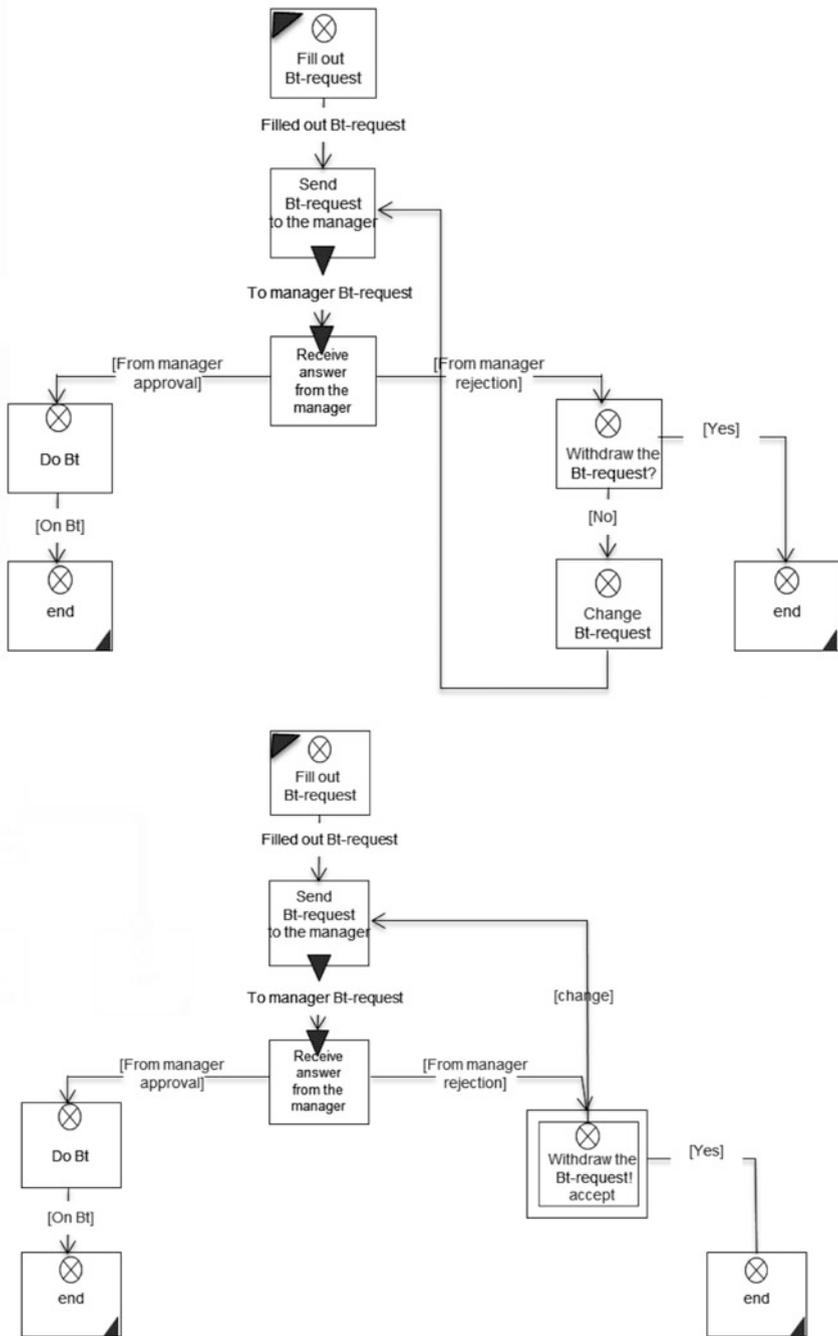


Fig. 5.9 Normalized behavior of the subject employee

To illustrate this issue, we use again the business trip as an example. Its resulting activities are shown in Fig. 5.10 in different levels of granularity.

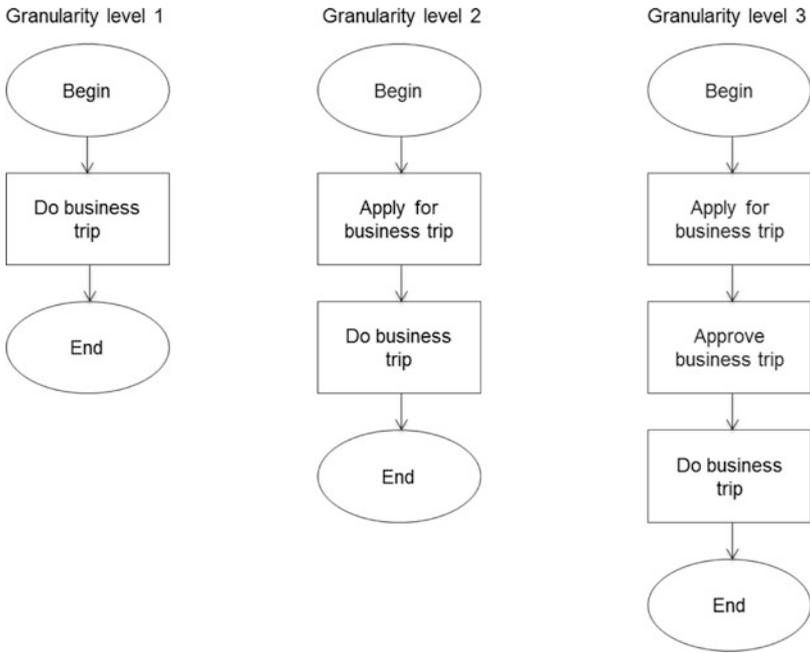


Fig. 5.10 Actions in the business trip application process in different levels of granularity

Figure 5.11 shows that only level three of granularity allows assigning activities to the three subjects “employee”, “manager”, and “travel office”. Otherwise, the activities were formulated too coarsely to be able to do this.

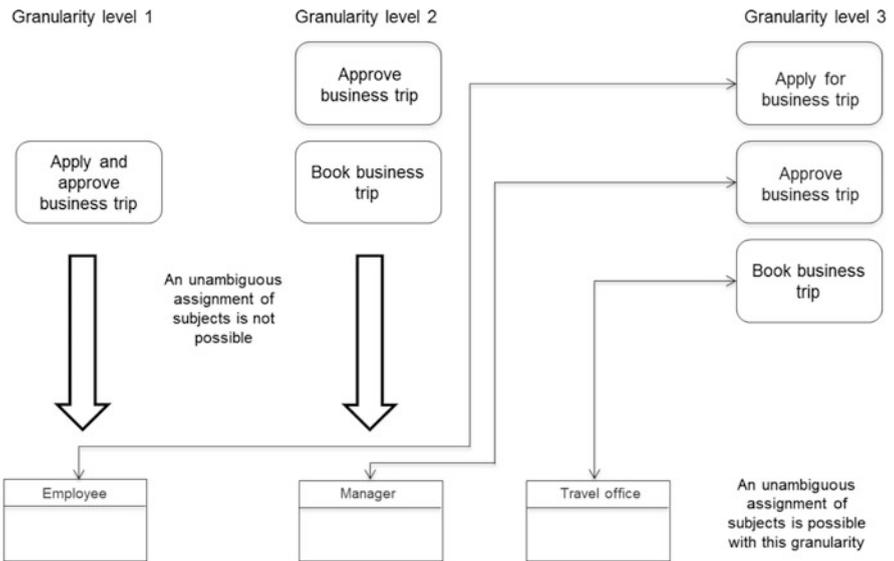


Fig. 5.11 Assigning tasks to subjects

The example illustrates that the granularity of actions in a process description is defined by the parties, or the active agents, involved in the process. The individual actions need to be clearly assigned to active agents.

The normalization thus determines the crudest possible description of a process as well as the minimal required granularity of process descriptions. The normalization of subjects is also required to identify the observable behavior of a process (see Sect. 5.6.7).

Discover matching processes; establish them as the ultimate guide on how to accomplish tasks through normalization. This helps stakeholders with orientation.

5.5.7 Business Objects

5.5.7.1 Understanding of Business Objects

In natural language, sentences are usually composed of a subject, predicate, and object (e.g., “Robert plays ball”). An object is not mandatory for a grammatically correct sentence structure, although if the object is missing, the sentence lacks the information on what or whom the predicate is acting upon (e.g., “Robert plays, but using what?”). This is transferable to a process:

A business process consists of actors who perform specific actions in a certain sequence, so-called predicates, and objects on which the predicates are defined. In this particular case, sending and receiving represent special predicates with the message as a direct object, and the addressee and sender as indirect objects.

Business objects are those things that are needed to provide outcome of business processes. Consequently, they are things that are used in a process. Business objects are passive, i.e., they do not initiate interactions or actions. Business objects are processed by subjects (cf. Grässle et al. 2004). They can outlast the execution of a process instance and can be used in process instances initiated later on as sources of information.

In the following, we deal with modeling of business objects and operations, which are processed on them in the course of executing process instances. The focus is less on physical business objects (e.g., a product which is delivered) than on logical business objects (such as the associated information for service delivery or a business trip application).

5.5.7.2 Structures of Business Objects

A basic structure of business objects consists of an identifier, data structures, and data elements. The identifier of a business object is derived from the business environment in which it is used. Examples are business trip requests, purchase orders, packing lists, invoices, etc.

Business objects are composed of data structures. Their components can be simple data elements of a certain type (e.g., string or number) or even data structures themselves.

For better understanding, it is recommended to describe the semantics of the data elements in more detail, especially if these cannot be unequivocally deduced from the identifiers.

Figure 5.12 shows an example of a business trip request. It consists of the data structure “data of requester” (employee) with data elements for name, first name, and personnel number, and the structure “data of trip” with the data elements for the start, end, and purpose of the trip.

Data structure	Meaning	Data type	Can/must	Value range/Default
Data of requester				
Name	Last name	Character	M	
First name	First name	Character	M	
Personnel number	...	Integer	M	
Organizational unit	...		C	
Pay group	...		C	
Data of trip				
Start trip	...	Date	M	Within 1 year from current date/ current date
End trip	...	Date	M	Start trip plus 1 year/ start trip
International trip	...	Boolean?	C	y/n, n
Travel destination (city/country)	...	Character	M	...
Reason for traveling	...	Character	M	...
Desired advance money	...	Integer	C	...
Data of approval				
Approval	Approval comment	Boolean?	M	y/n, n
Cost center	...	Integer	M	...
Desired advance money	...	Integer	C	...

Fig. 5.12 Data structure of the business object “business trip request”

5.5.7.3 Status of Business Objects and Their Instances

In many cases, the semantics of a business object changes during process execution, such as when a delivery slip is transferred into an invoice. Therefore, for a business object several different statuses can be defined. If a status changes, only those data structures or data elements, which are required for the new status, are transferred from the previous status, and new components are added as needed, or existing removed if no longer necessary. This ensures that a subject receives only those data elements for its work that it really needs. This will facilitate compliance with data protection regulations.

In the example of the business trip application, the status “booking business trip” can be derived from the original status “business trip request” of the business object (see Fig. 5.12). In particular, data elements with internal information such as

employee number, category of salary, reason for travelling, and the complete data structure for approval are removed. They should not be visible, e.g., outside the organization, and are not relevant for the (external) travel agent to book the trip. Therefore, as shown in Fig. 5.13, a new data structure “data of booking” is added. It contains data elements, which allow the travel agent to set a deadline for the latest possible receipt of the confirmation of booking while specifying certain hotel chains which have been contracted.

Data structure/ Data element	Meaning	Data type	Can/must	Value range/Default
Data of requester				
Name	Last name	Character	M	
First name	First name	Character	M	
Data of trip				
Start trip	...	Date	M	Within 1 year from current date/ current date
End trip	...	Date	M	Start trip plus 1 year/ start trip
Travel destination (city/country)	...	Character	M	...
Data of booking				
Contracted hotel chains	Approval comment	Character	M	...
Deadline of booking confirmation	...	Date	C	...
Booking confirmation	...	Date	M	y/n

Fig. 5.13 Business object “business trip request” in the status “booking business trip”

Using status information, a form template can be constructed. First of all, a status is defined as a business object type, from which different variants of business objects for use in other business process environments can then be derived. For instance, it would be conceivable that the travel office provides booking of private tours as a special service to staff members. In such a case, a business object “private travel booking” could be generated from the previous status of the business object “business trip request” by removing data fields irrelevant for private trips (e.g., reason for the trip, advance payments, etc.), and supplementing with others (e.g., in case a travel insurance is requested).

5.5.7.4 Views of Business Objects and Their Instances

Besides the definition of statuses for business objects and their instances, it may be necessary to define different views for different subjects. In contrast to status changes, in views the data structures or elements are not physically removed from a business object and its instances, but rather only different access rights are assigned to it. This is done for each subject in its respective process context, i.e., for the particular behavior status of the subject. As usual, read access (read) means that a subject can only see data elements and their content. In case of an assigned write permission, values can additionally be changed (read/write).

Figure 5.14 shows the views of the subjects “employee”, “manager”, and “travel office” in the status “business trip application” of the business object “business trip request”. The applicant can read all the data elements but is not able to fill in approval data, the cost center, and the amount of an advance payment. This is reserved for the manager. The view of the travel office includes only read permissions, and not even these for certain data elements. Thus, the reason for the trip and the advance payment requested by the employee are not accessible for the travel office at all, as they are not relevant for the actions of this subject.

Data structure/ Data element	View of employee	View of manager	View of travel office
Data of requester			
Name	R/W	R	R
First name	R/W	R	R
Personnel number	R/W	R	R
Organizational unit	R/W	R	R
Pay group	R/W	R	R
Data of trip			
Start trip	R/W	R/W	R
End trip	R/W	R/W	R
International trip	R/W	R	R
Travel destination	R/W	R	R
Reason for traveling	R/W	R/W	-
Desired advance payment	R/W	R	-
Data of approval			
Approval	R	R/W	R
Cost center	R	R/W	R
Allowed advance payment	R	R/W	R

Fig. 5.14 Views on the business object “business trip request” in the status “business trip application”

Let us have a look at the views of the business object “business trip request” in the advanced status “trip booking” (see Fig. 5.15). This status is relevant to the travel office, as it monitors the receipt of the confirmation from travel agents, and if necessary, changes travel dates in case of availability problems. The employees, however, are only interested in information on whether the trip has already been successfully booked, whereas the manager does not need a view on this status at all.

Data structure/ Data element	View of employee	View of manager	View of travel office
Data of requester			
Name	R	-	R
First name	R	-	R
Data of trip			
Start trip	R	-	R/W
End trip	R	-	R/W
Travel destination	R	-	R
Data of booking			
Contracted hotel chains	-	-	R/W
Deadline of booking confirmation	-	-	R/W
Booking confirmation	R	-	R/W

Fig. 5.15 Views on the business object “business trip request” in the status “trip booking”

5.5.7.5 Access Privileges to Business Object Instances

For business object instances, the modeler can specify whether only a single subject, namely, the one initiating the instance, can access them directly, or also other subjects. Accordingly, we distinguish between local and global business objects.

Local Business Object (Private Business Object)

A subject creates a local instance of a business object. Its data elements can only be read or modified by the generating subject. Other subjects can acquire access to an instance of a business object when a copy of that instance has been explicitly sent to them in a message.

Local business objects are appropriate for business transactions with external partners, such as suppliers and customers, because external subjects should not have direct access to business objects for reasons of security. Changes that are required in accordance with a certain business logic can also be returned by message exchange and lead to controlled modification of the data of the private business object by the designated and authorized subject.

In Fig. 5.16, only the subject “employee” can access its copy of a business object “business trip request”. The manager can only add his information once he has received the message with a copy of the business object. Similarly, the travel office can only handle the case after it has received a copy of the business object from the manager in the new status “business trip request approved”. By sending or receiving messages, a copy of the required business object is transferred to the respective partner.



Fig. 5.16 Business trip request as a private business object of the subject “employee” (transmission via message)

Global Business Object (Shared Business Object)

A global business object, when being defined, can be assigned to several subjects (“object owner”) can edit data elements in instances of the global business object according to their access rights controlled by views. A corresponding example is shown in Fig. 5.17.

Since all the involved subjects can access the business object “business trip request”, it is sufficient that the employee fills in the form (business object) and then informs his manager by sending a message, without transferring the business object. The manager can then directly access the application request and make his amendments. This also applies in the later phases of processing the trip, e.g., by the travel office.

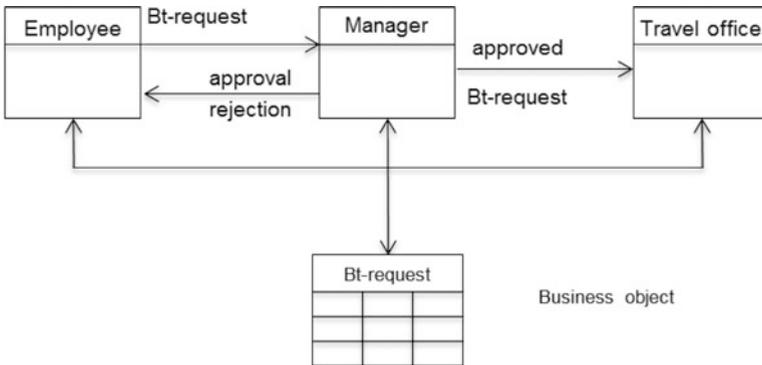


Fig. 5.17 Business trip application as a global business object of the subjects “employee”, “manager”, and “travel office”

Global business objects can be shared by any number of subjects of an organization in a complex process network (see Sect. 5.5.5). The benefit is that various subjects can access a common database with secure transactions, as there are not multiple copies of a business object in use. The disadvantage is that the subjects need to be able to access common business objects. In interorganizational processes, this often cannot be achieved without substantial effort.

However, using the concept of global business objects, complex access right issues can be clarified elegantly: a subject only has access to a business object when

it has a task to accomplish within the process instance which is associated with the business object under consideration.

In the example of the business trip application, it is not necessary for the travel office to have access to all personal and trip data permanently. However, when using static structures, no other solution is possible. On the other hand, if the data access is implemented through a global business object, the travel office only has access to the data when this is required for processing their associated tasks. At the latest, the data is protected again upon completion of the process.

Not all task performers need to see and manipulate all data. They should take a certain view of business objects. These “glasses” should reflect the information that they need to have to accomplish their tasks—no more and no less, but in association with time, namely just-in-time.

5.5.7.6 Operations on Business Objects

When executing business process instances, subjects perform operations on business object instances as part of their task and communication profile. Depending on the privileges of a subject, the following operations are possible:

- **Generate business object instances:** A subject can generate a business object instance by deriving it from the general business object definition, or copying it from an already existing instance.
- **Assign values to business objects:** Once business objects have been instantiated, the values required for the execution of the process need to be assigned to the individual elements by the authorized subjects. How these values are entered, shall be determined as part of the implementation of a process when being embedded in the organizational and IT environment. Examples include the identification and manual entry of data by people (e.g., quantity in an order position) or the saving of an automatically computed result by an IT system (e.g., VAT amount of an invoice).
- **In case of the business trip process,** for instance, a concrete object is generated from the specification of the business object “business trip request”, once an employee requests a trip. When filing a request, it is conceivable that the employee himself manually enters his personnel number into an electronic form and the IT system uses this to determine his name, first name, and category of salary and automatically enter them into the appropriate fields.
- **Duplicate business object instances:** Business object instances can be duplicated, e.g., to preserve a certain status of a business object. In the example of the business trip, the status can remain the same after completion of the form by the employee, until, for instance, the manager performs changes, e.g., changes the date. Each duplicate is given a unique name in order to distinguish it from other instances. This is defined in the status in which the copy is created.

- **Transfer data elements from a business object instance:** From a business object instance, field values can be transferred to data elements in instances of other business objects. Only the types of the data elements need to match. Such a mapping of values must be defined in function states of the process description of a subject. In case there are already duplicates of the target instance, it must also be specified to which duplicates the mapping refers to.
- **Change status of a business object instance:** A status change in business objects has been introduced as a variation of the initial business object by means of dismissing and/or adding data elements. Here, in a function state at runtime, i.e., for business object instances, retained data elements are transferred with their values to the new status. Data items no longer needed are deleted along with their values for the new status, while added data elements are initially empty and waiting to be entered. Here too, it must be specified in case of multiple instances to which instance the change of status refers to.
- **Send business object instance:** This operation can be performed only in a send state. As a result, a copy of a business object instance is sent. In case there are multiple copies of the instance, it must be specified to which instance the send operation refers to.
- **Receiving a business object instance:** A subject as addressee of a message with a business object instance must be in a receive state to accept the message. Once it takes this message from the input pool, a uniquely identifiable copy of the business object instance is created.

How the respective operations will be run on a business object is specified in the context of the IT implementation of a business object (see Sect. 10.5.1). In the course of modeling, it is only specified which operations are performed on a business object and which of its content parts need to be changed when tasks are accomplished.

With the view comes the privilege. The access rights to business objects are derived from the required task support. It has to be clarified whether a stakeholder requires access to a business object at all, and if so, whether he is only allowed to read it, or possibly even change it.

5.6 Extension Constructs for Process Networks

5.6.1 To Go

This sounds quite convincing. But what about complex processes involving a large number of subjects? I need to decompose them into related sub-processes.



In each of the processes, there are then subjects communicating with other subjects in other processes.



We can decompose a large process into sub-processes, and each of the involved parties describes his sub-process.



Yes, these are the interface subjects.

Does this mean I only need to know the interface subject of another process?



In this way, we can construct entire networks of processes. We can recognize all the mutual dependencies between processes in our organization.



Exactly, it is represented in your process as a so-called external subject.



5.6.2 Interface Subjects and Process Network

So far, we have considered only individual processes. However, processes are generally mutually dependent, i.e., subjects in one process communicate with subjects in other processes. In this way, networks of processes are created. Conversely, large and complex processes can be decomposed into smaller subprocesses. In the following sections, we introduce the various concepts for the formation of process networks.

Networked organizations especially benefit from S-BPM. This approach enables the structuring of the flow of information in a transparent form across the boundaries of an organization, and the disclosure of those parts of participating organizations that are required by network partners for successful cooperation.

The process “business trip application” represents only a portion of the entire business trip process. In reality, this process can consist of a whole series of small, highly interrelated processes. For instance, after approval by the manager, a subsequent process could address the travel office, booking through a travel agent a train ticket and a hotel room for the employee (applicant). When modeling using

the basic S-BPM constructs, this results in the subject interaction diagram extended by the booking process, as shown in Fig. 5.18.

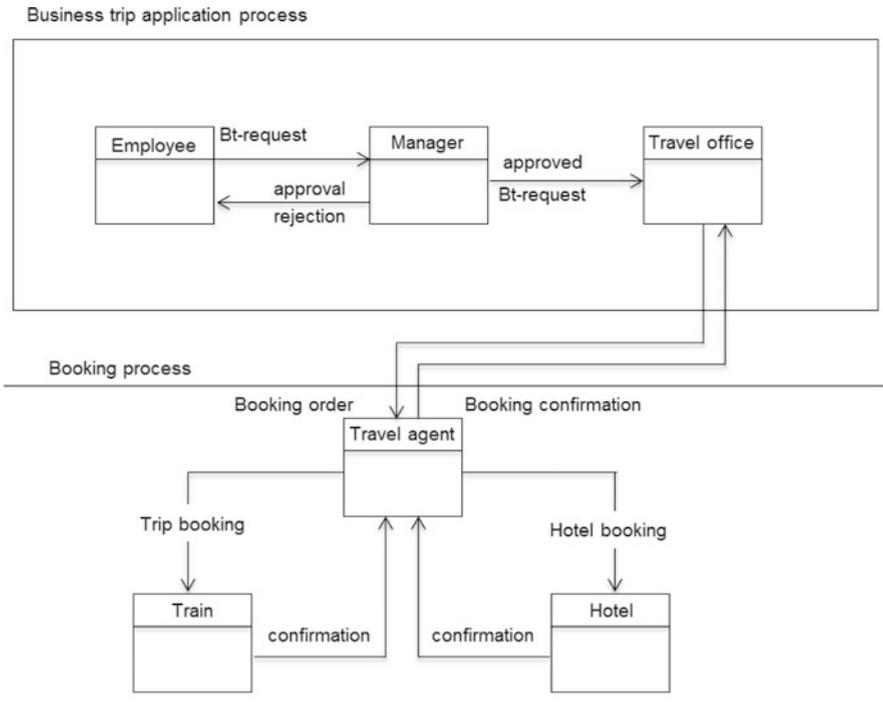


Fig. 5.18 Extended subject interaction diagram for the process “business trip application”

In order to structure and simplify the representation, the overall process can be decomposed into the two coupled subprocesses: “business trip application” and “booking”. Subprocesses describe specific, logically self-contained aspects of a complex process. The overall process is denoted as a process network. In this network, it is required that subjects of the subprocesses are linked across their process boundaries and communicate with each other.

A link between two processes is represented through interface subjects that reference one another. The associated interface subject of the respective other process is represented in the considered process through a so-called external subject.

Interface subjects regulate cooperation and facilitate the synchronization of processes of the network partners.

In the example, from the perspective of the subprocess “business trip application”, the travel agent is the interface subject. In the subject interaction diagram in

Fig. 5.19, it is indicated by gray coloring as an external subject. The reference symbol also contains “booking process” as the name of the process which contains the referenced subject. From the perspective of the booking process, the travel office is the interface subject, and “business trip application process” indicates the process containing this external subject.

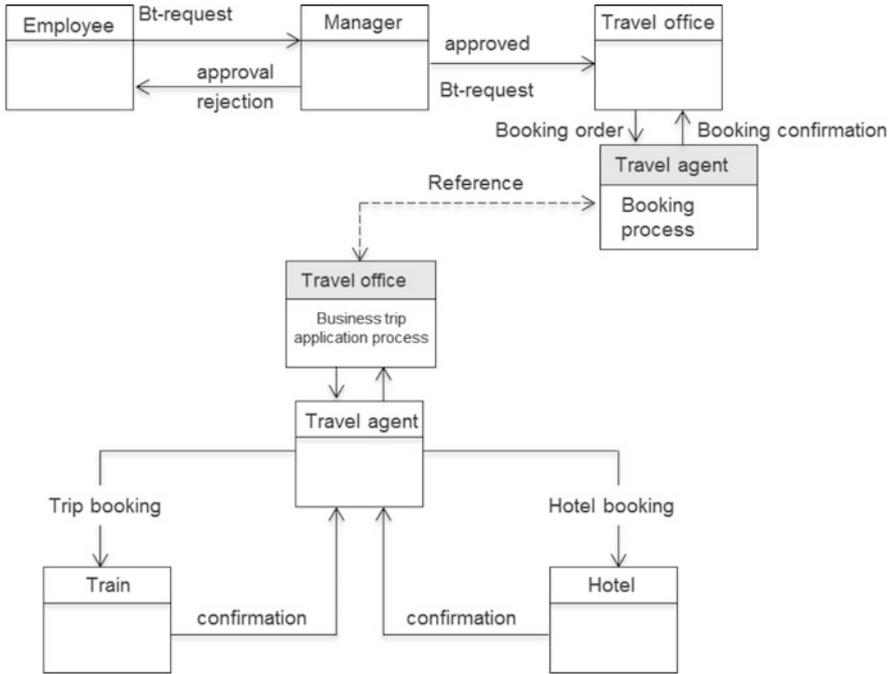


Fig. 5.19 Subprocesses “business trip application process” and “booking process” linked via interface subjects

Using mutual referencing, subject interaction diagrams (SIDs) can be consolidated into process network diagrams (PNDs), which only show processes linked in a process network and the messages exchanged across their borders. We refer to these as horizontal process networks. Such a network is presented in Fig. 5.20 as a Process Network Diagram for the entire business trip process in its currently developed form.



Fig. 5.20 Horizontal process network for the business trip process

5.6.3 Service Processes

In operational reality, there are (sub) processes which deliver defined results which can be encapsulated as a service process. Several other processes call this process to take advantage of its results.

For coupling the calling process with the service process, a so-called general external subject is introduced for the service process. It represents all the processes that use the service process. In this way, all sorts of calling subjects are implicitly referenced in the course of modeling, instead of setting explicit references to the respective subject in the calling process.

In the example of the business trip process, the booking process can be implemented as a reusable service process and thus made available to other calling processes. This could be useful, e.g., if an organization offers its employees booking of private tours through the travel office with special conditions. Then, the employees use the respective service process not only for booking business trips but also for vacation trips.

In such a service process, the utilizing process needs to know the interface subject of the service process. It will communicate with it as usual, so that nothing changes for the description of the behavior of the utilizing processes.

Figure 5.21 shows the booking process as a service process using “booking customers” as a general external subject.

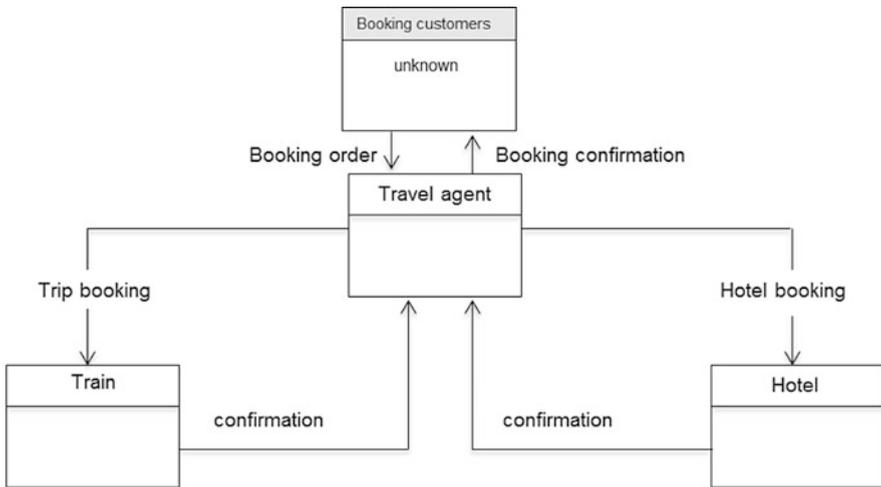


Fig. 5.21 Booking process as a service process with a general external subject

At the time of modeling, the service process neither knows the interface subject nor the utilizing process to which it belongs. Therefore, the external subject representing the interface subject in the processes calling the service process needs to be provided with a formal name. In this way, the messages, which are sent by the subjects of a service process to the utilizing processes, can be addressed.

In our example, the formal name “booking customers” is given. The process name “unknown” in the external subject identifies the considered process as a service process.

5.6.4 Multiprocesses

In a business process, there may be several identical subprocesses that perform certain similar tasks in parallel and independently. This is often the case in a procurement process, when bids from multiple providers are solicited. A process or subprocess is therefore executed simultaneously or sequentially multiple times during overall process execution. A set of type-identical, independently running processes or subprocesses are termed multiprocess. The actual number of these independent subprocesses is determined at runtime.

Multiprocesses simplify process execution, since a specific sequence of actions can be used by different processes. They are recommended for recurring structures and similar process flows.

An example of a multiprocess can be illustrated as a variation of the current booking process. The travel agent should simultaneously solicit up to five bids before making a reservation. Once three offers have been received, one is selected and a room is booked. The process of obtaining offers from the hotels is identical for each hotel and is therefore modeled as a multiprocess.

As a result, the representation is changed first on the abstract level of the process network diagram as shown in Fig. 5.22, where the nesting expresses that the “hotel offer and booking process” is a multiprocess.

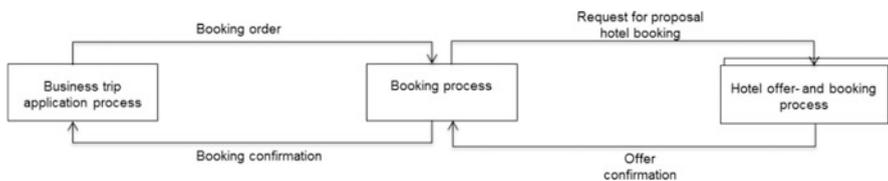


Fig. 5.22 Process network diagram “business trip application with hotel selection”

On the next level of detail, the subject interaction diagram, the nested symbol for the interface subject “hotel” shows that it belongs to a multiprocess (see Fig. 5.23). Every time the subject “travel agent” sends the message “request for proposal” to the subject “hotel” from the multiprocess “hotel offer and booking process”, a new copy of this process is generated. Each copy corresponds to a specific hotel inquiry.

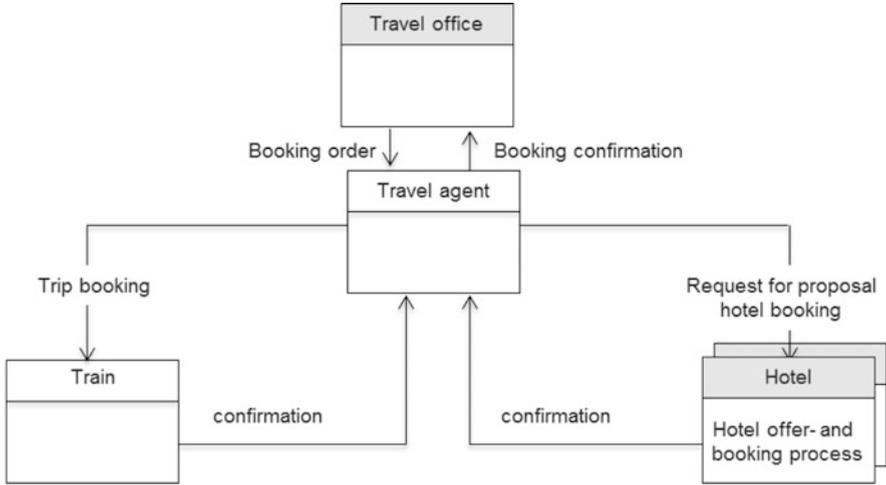


Fig. 5.23 Subject interaction diagram for the “booking process” with the “hotel offer and booking process” as a multiprocess

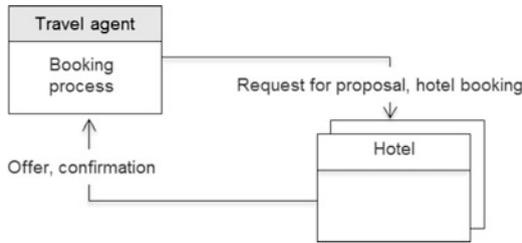


Fig. 5.24 Subject interaction diagram for the “hotel offer and booking process”

The “hotel offer and booking process” contains only the subject “hotel”, which communicates with the external subject “travel agent” in the booking process (see Fig. 5.24).

Multiprocesses are described in the same way as other processes of a process network. A supplement is required for the commissioning subject that communicates with a subject of a multiprocess. It needs to know how many and which copies of a multiprocess it has produced. Therefore, when describing its behavior, the respective copies are indexed like elements of a field, in order to identify the relevant copy for process state transitions. In case a subject wants to communicate with a subject of a particular process copy from the multiprocess field, it specifies the proper index of the process copy when sending or receiving. In our example, in the action “select hotel” the index for the best bid is saved in the parameter “selected”. This allows communication with the corresponding bidding hotel.

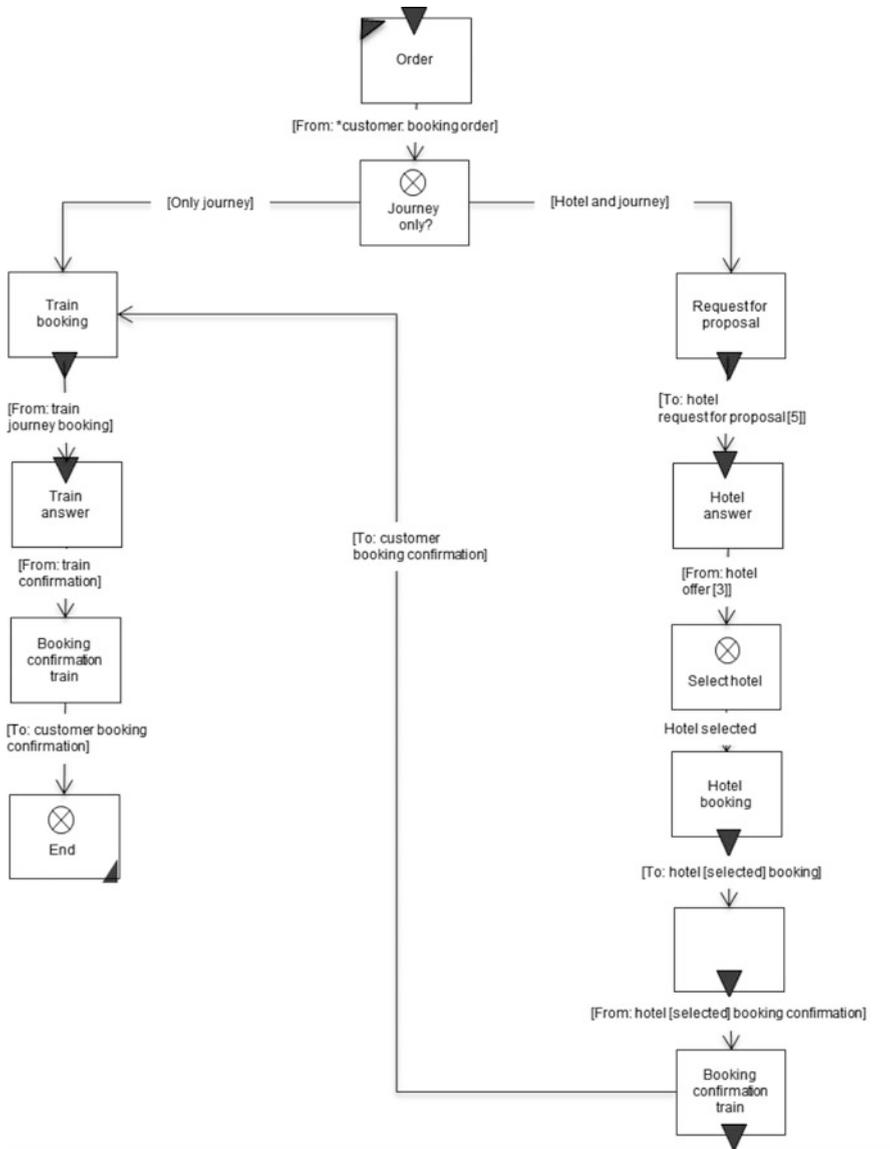


Fig. 5.25 Behavior of the subject “travel agent” with a multiprocess for selecting hotel

Figure 5.25 illustrates this situation with the state transitions [to: hotel request for proposal [5]] and [from: hotel offer [3]]. This specification expresses that offers from five hotels need to be obtained, and that a hotel will be selected and booked as soon as three bids have been received.

5.6.5 Complex Process Network Topologies

So far, we have mainly considered process networks with two or three processes and have illustrated the methods for linking processes. However, it is possible to expand networks to arbitrary complexity and to structure them hierarchically. Hereby, hierarchical structuring is not an extension of the means for representation, but rather a structured application of the previously described capabilities for linking processes. Process links in complex process topologies can be vertical or horizontal and can be constructed with “vertical” and “horizontal” subjects.

We will now demonstrate such a case for the “business trip application” process. It could be embedded into a more comprehensive process network termed

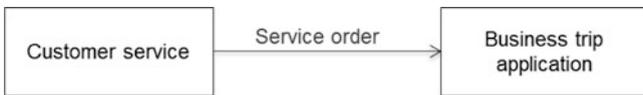


Fig. 5.26 Processes of the process network “customer care”

“customer care”. In such a network, customer reports could be received and edited by the process “customer service”. In some cases, to handle the customer request, a customer visit by a service employee could be required. This is initiated by sending the message “service order” triggering the process “business trip application”. Figure 5.26 shows this process network.

Messages, according to the S-BPM methodology, are not exchanged between processes, but always between subjects in processes. This results in the example in a refinement in which the subject “service desk” from the process “customer service” sends the message “service order” to the subject “employee” of the process “business trip application” (see Fig. 5.27). Both subjects are external subjects from the respective viewpoint of the other process and are not interested in the behavior of

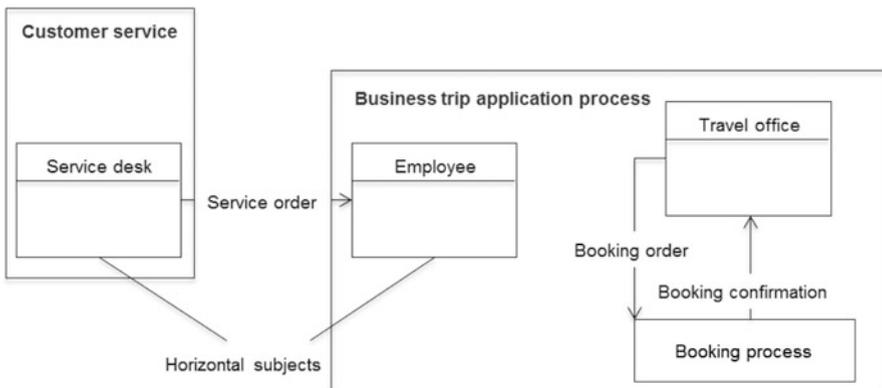


Fig. 5.27 Linking processes in process networks using interface subjects

their communication partner in the other process. Other subjects occurring in both processes, such as the manager in the business trip application process or a service dispatcher in the customer service process, therefore remain hidden at this level. These subjects are not visible from the respective perspective of the other process.

In the process “business trip application”, the subject “travel office” sends the message “booking order” to the booking process and receives the message “booking confirmation” in return. The booking process is not visible to the process “customer service” as a whole; it will be encapsulated by the process “business trip application”. This puts the booking process one level lower than the processes “customer service” and “business trip application”, which are on the same hierarchical level and are connected by the subjects “service desk” and “employee” through horizontal communication relationships. Subjects communicating with subjects of other processes on the same level are called horizontal subjects.

A refinement of the booking process by introducing the subject “travel agent” as a communication partner to the travel office leads to the representation shown in Fig. 5.28. Due to their vertical communication relationship, the travel office and travel agent are referred to as vertical subjects. All subjects of a process which communicate with subjects in processes in a higher or lower hierarchical level are termed vertical.

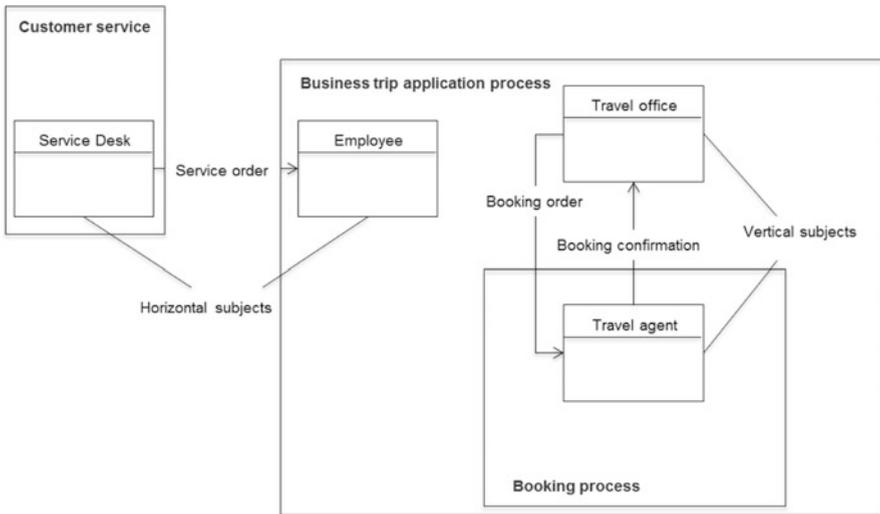


Fig. 5.28 External subjects in multilevel hierarchical process networks

Our example illustrates that for the structure of a process network, only those subjects are essential which communicate with subjects in other processes. Interface subjects thus define relationships between processes, and in this way, the process network. From the perspective of subjects of a particular process in the network, it does not matter whether their perceived external subjects are involved in

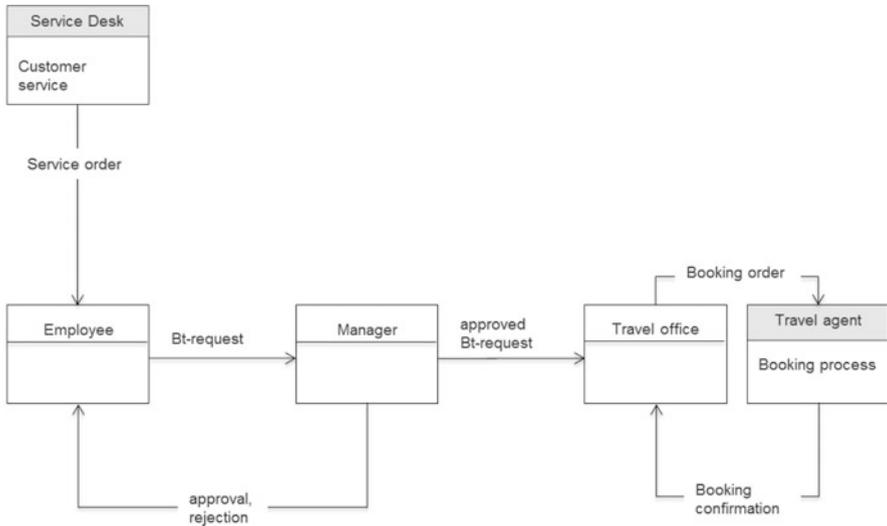


Fig. 5.29 Business trip application process with the associated external subjects

communication relations with other subjects of their process or not. In Fig. 5.29, this indifference becomes evident.

The previously presented concepts for the construction of hierarchical process networks will now be detailed using a complex, however abstract example. Figure 5.30 shows a process network with the three processes A, B, and C, with each of these in turn representing a process network in itself.

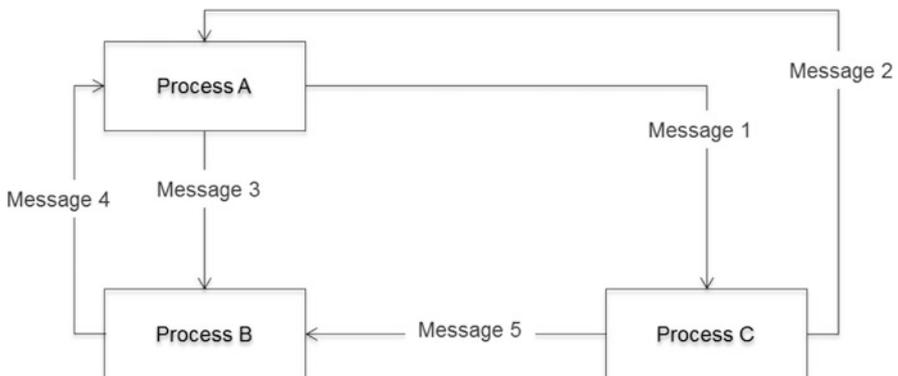


Fig. 5.30 Example of a complex process network

In the process network in Fig. 5.31, “process A” consists of the processes “A1” and “A2” and the external subjects (interface subjects) “SA1” to “SA4”. The subjects “SA3” and “SA4” represent “process A” with respect to “process B” and

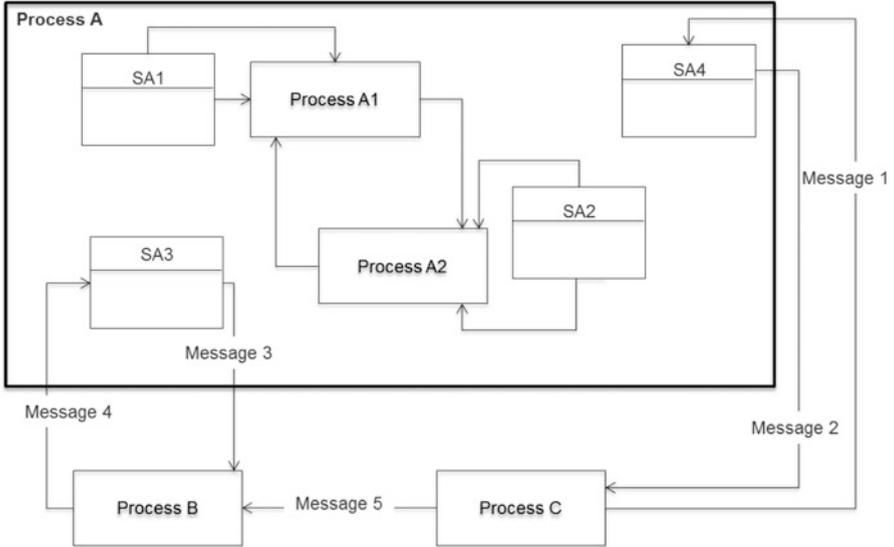


Fig. 5.31 Internal structure of a hierarchical level of a complex process network

“process C”, while “SA1” and “SA2” communicate with “process A1” or “processA2”, respectively.

In addition to the interface subjects, “process A” may contain other subjects which interact internally, but are not relevant for other processes, and therefore are hidden. In Fig. 5.32, the refined subject interaction diagram of process “process A” is shown. Instead of the partner processes “B” and “C”, the corresponding external subjects “SB1” and “SC1” are included. The processes “A1” and “A2”, which are only visible in “process A”, are represented as the external subjects “SA11” and “SA21”. The relationship between the processes “process A1” and “process A2” is not relevant for the subjects of “process A” and is therefore not included in Fig. 5.32. For reasons of intelligibility, in this figure, as well as in the subsequent diagrams, the messages exchanged between subjects are shown exclusively by arrows (without labeling them).

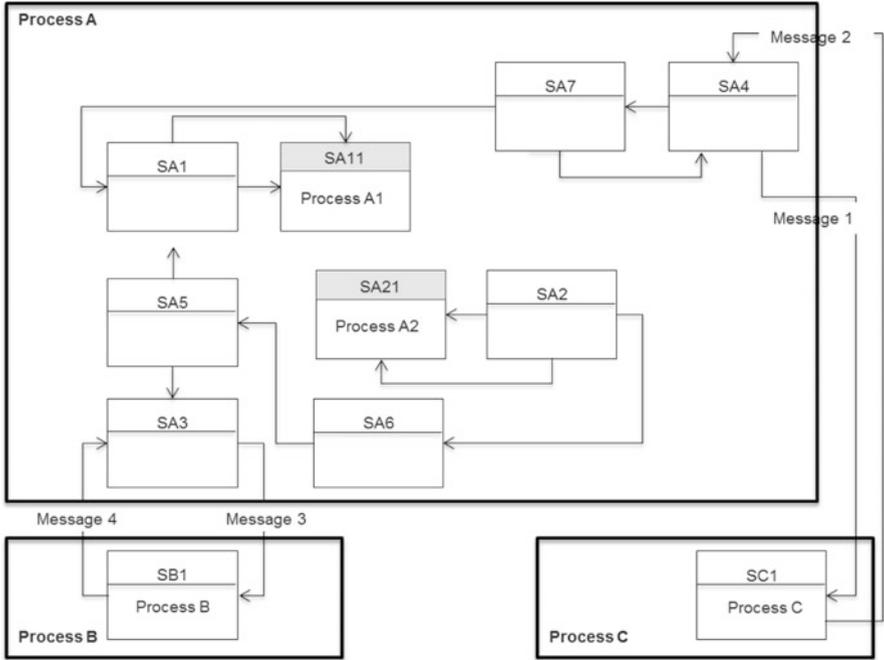


Fig. 5.32 Communication structure of “process A”

In Fig. 5.33, we take a closer look at the communication structure of “process A1” and “process A2”. In the upper part, we see for “A1” that its partner “process A” is represented by the interface subject “SA1” (vertical relationship), and its partner “process A2” by the interface subject “SA23” (vertical relationship). Accordingly, in the lower part for “process A2”, the interface subjects “SA2” and “SA12” connect it to its partner “process A” (vertical relationship) and “process A1” (vertical relationship).

After having detailed the individual sections of the network, Fig. 5.34 shows the hierarchy of the complex process system. It includes only those subjects which communicate with subjects from other processes. They are recognized as interface subjects in these processes.

“Process A” communicates via the horizontal subjects “SA3” and “SB1” with “process B”, and via “SA4” and “SC1” with “process C”, respectively. The processes “process A1” and “process A2” are subordinate to “process A”. Subjects in these processes can therefore only be reached via processes of “process A”, e.g., via the connections of the vertical subjects “SA1” and “SA11”, or “SA2” and “SA21”, respectively.

Figure 5.34 shows the external subjects of “process A”.

Analogous to the hierarchical structuring of “process A”, “process B”, and “process C” can be further decomposed. Figure 5.35 shows the processes embedded in “process B” and “process C”, and the associated horizontal and vertical subjects.

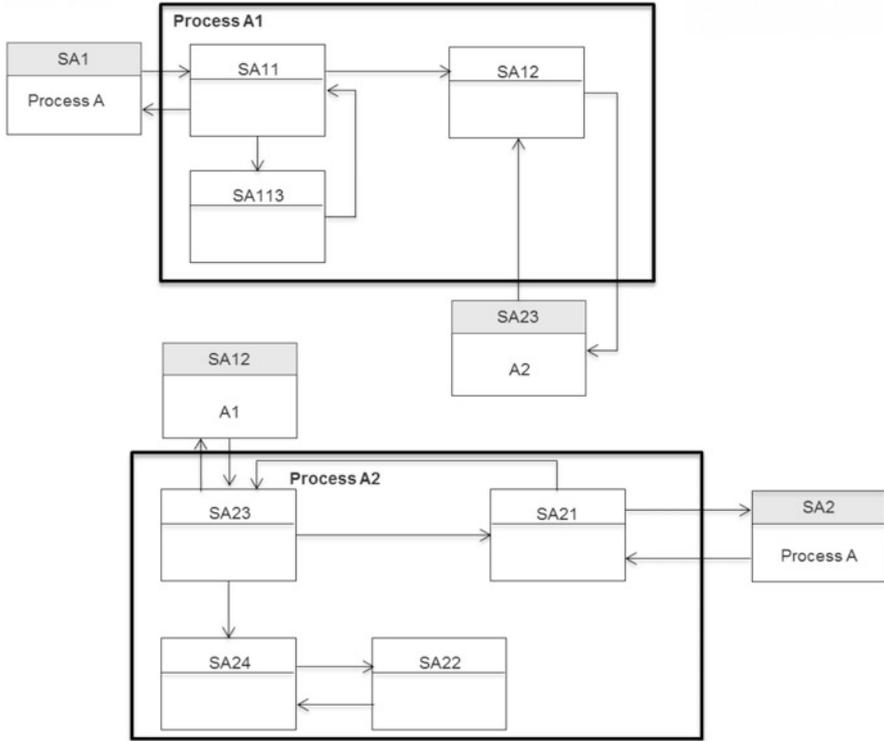


Fig. 5.33 Communication structure of “process A1” and “process A2”

Here, “process A” is again reduced to its external subjects from the perspective of processes “B” and “C”.

The above-mentioned concepts have revealed the S-BPM capabilities to structure a complex process system in subsystems as efficiently as possible. If we combine Figs. 5.30 and 5.31, a communication structure of the complex process system emerges, including all horizontal and vertical subjects. A complete representation would additionally include all of the internal subjects, which are not visible to subjects of other processes and were therefore hidden.

Such a fully resolved structure is usually very confusing. For a compact overview of a complex hierarchical process network, we therefore introduce the Process Hierarchy Diagram (PHD). It allows the consolidation of the representation into the hierarchical and communication relationships between processes. Figure 5.36 shows the process hierarchy diagram for the example used.

We now consider subjects of processes, which are embedded in a process network. They can only indirectly communicate with the other subjects of the subject network. As seen in the process hierarchy diagram in the figure above, a subject of “process B1” can only communicate with a subject “C1” via a subject of “process B” and “process C”.

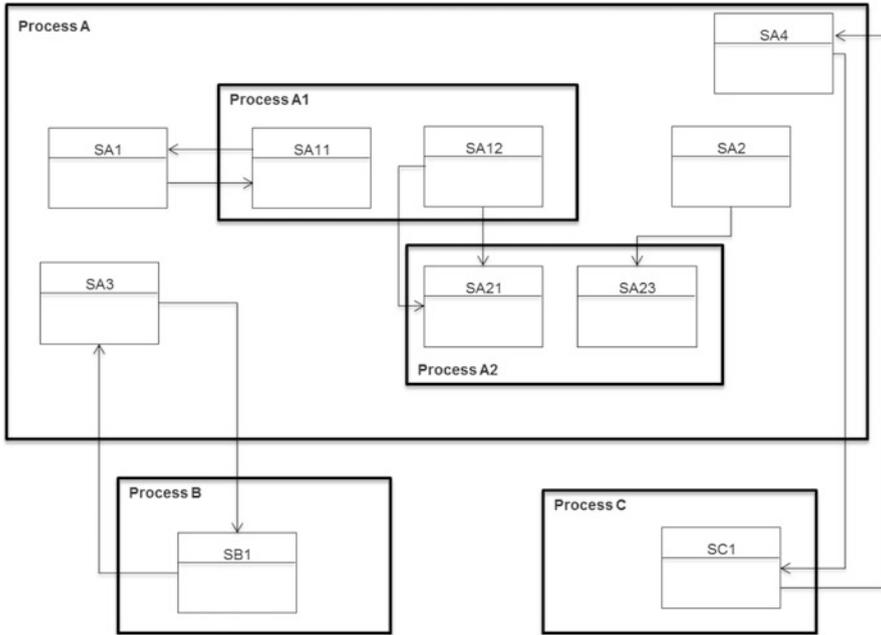


Fig. 5.34 External subjects of “process A”

5.6.6 Business Objects in Process Networks

What is the impact of hierarchical relationships of processes on the joint ownership (joint access rights) of subjects with respect to business objects? In the context of hierarchical process networks, Shared Business Objects can be defined as follows:

- Joint ownership of all subjects of a particular hierarchy level: All subjects of a certain process network on a particular hierarchy level can access a specific business object for reading and/or writing if views with appropriate access rights are available. This is not possible for subjects on levels above or below the addressed one. In Fig. 5.37, “Bo-1” is a Shared Business Object. It is in joint ownership of the subjects “SC1”, “SC3”, and “SC6”, as well as of all other subjects at this level. The latter do not appear in the figure, as they are not interface subjects.
- Joint ownership of all subjects from a particular hierarchy level downwards: In this case, in addition, all subjects of the processes “process C1” and “process C2” are joint owners of the business object “Bo-1” in Fig. 5.37.
- Joint ownership of all subjects from a particular hierarchy level upwards: With such a definition, business object “Bo-2” in Fig. 5.37 is in the joint ownership of all subjects in the process “process C2” and all subjects of the parent process.
- Joint ownership of all subjects in the entire hierarchical process network: Each subject of the hierarchical process network has access to such a business object, according to its views.

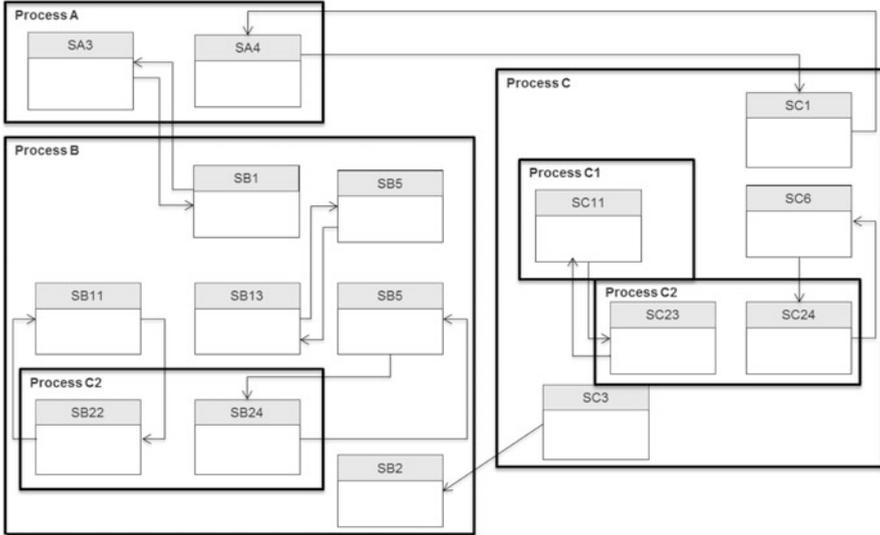


Fig. 5.35 Process hierarchy of “process A” and “process B”

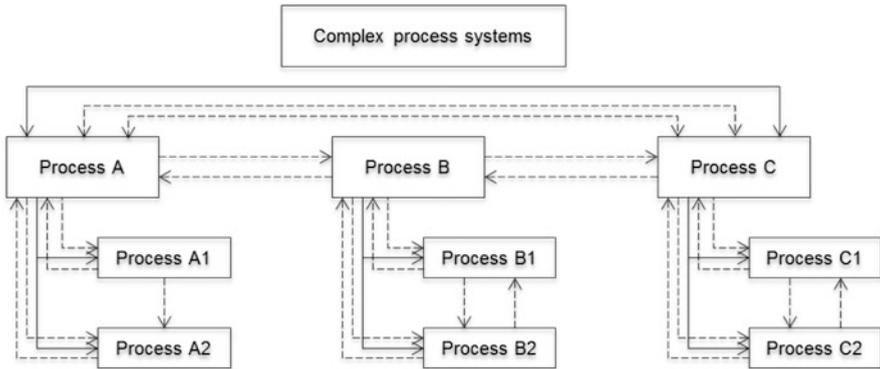


Fig. 5.36 Process hierarchy diagram

Intelligibility can be achieved by applying the following guiding principle during the modeling process: “As simple as possible, but as complex as necessary”. Process networks may lead to a hierarchical structure. They facilitate “stepping through” by introducing generalization and refinement.

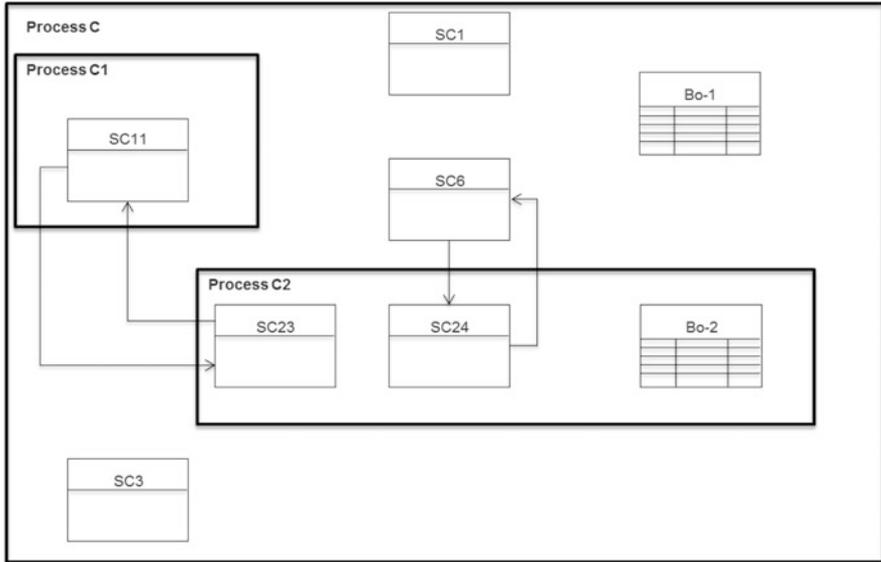


Fig. 5.37 Joint ownership of business objects in process networks

5.6.7 Reduction to Observable Behavior

Following the “as simple as possible” principle again, we can often reduce behavior to what is visible in the network.

The previously discussed simplifications when representing horizontal process networks refer to the interaction structure. In addition, even at the level of subjects, behavior representations are necessary to derive the externally visible behavior of an interface subject. In this context, we exploit the fact that subjects, which belong to different processes but yet are interacting with each other in process networks, are not interested in the internal behavior of the partner subject.

The functional behavior of an external communication partner and its interactions with other subjects in its native process are generally not relevant to subjects in linked processes. A subject is only interested in its partner’s communication behavior in the other process to the extent that it is directly affected by this behavior. Therefore, the partner’s behavior can be reduced to that interface when modeling. This is first done by replacing all those send and receive states of its communication partner used to simply interact with process-internal subjects with so-called pseudo-internal functions. In this way, the subject is shielded from communication behavior of the partner subject that does not directly affect it. In a second step, parts of the action behavior of the partner subject can be hidden by normalizing its behavior as shown in Sect. 5.5.6. Subject behavior reduced in this

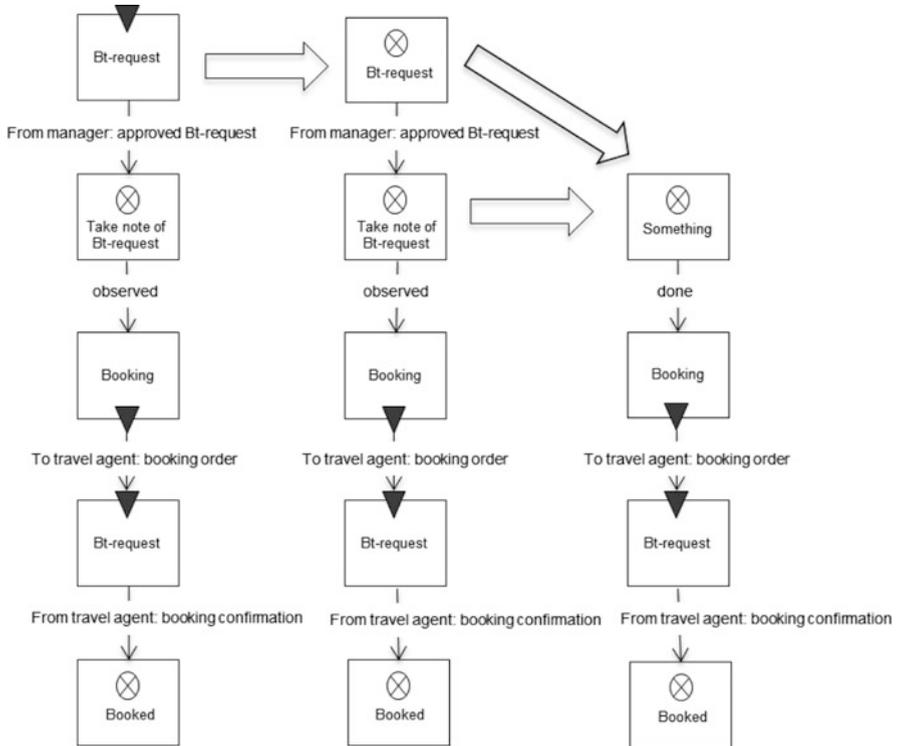


Fig. 5.38 Deriving interface behavior of the process “business trip application” with respect to the external subject “travel agent” from the behavior of the subject “travel office”

way is externally observable, and ultimately, represents the interface description of a process toward the partner process.

We introduce behavior reduction therefore as restriction on the behavior of a subject to aspects, which need to be recognizable by another subject in a linked process.

In the example, it is not relevant for the interface subject “travel agent” that the “travel office” communicates within its subprocess “business trip application” with the manager of the applicant. The travel agent is interested only in the communication behavior of the travel agency referring to him directly, i.e., the fact that they order him to book. The original behavior of the travel office, as shown in the left part of Fig. 5.38, can therefore be reduced from the view of the travel agent to the behavior visible in the far right part.

The first step is to replace the receive state “business trip request” by the pseudo-internal function “business trip request”. The result is shown in the middle of the behavioral description. This can then be further simplified by normalization: Both internal states “business trip request” and “take note of business trip request” are summarized to the function “something”. The right description emerges, representing the interface behavior of the process “business trip request” with respect to the subject “travel agent” in the booking process.

5.7 Extension Constructs for Subject Behavior Specifications

5.7.1 To Go

The structuring concepts for large process structures work quite well, but we still have problems describing complex subject behavior.



What do you mean by complex subject behavior?
Can you provide concrete examples here?



Yes, take for instance recurring patterns of behavior - we have these for approvals, or when messages come in unexpectedly (out of sequence), such as cancelations. Basically, we can model such situations using existing constructs. However, such representations cannot be constructed in a straightforward way and they are not really transparent.



There are modeling capabilities to handle this. Macros can be used for similar behavior structures. And exception handling can be used in case reactions to specific messages need to be modeled.



That sounds good. Did it help you?



Yes, we could actually use macros for recurring behavior structures, and exception handling to represent proper reactions to important messages.



5.7.2 Behavior Macros

Quite often, a certain behavior pattern occurs repeatedly within a subject. This happens in particular, when in various parts of the process identical actions need to be performed. If only the basic constructs are available to this respect, the same subject behavior needs to be described many times.

Instead, this behavior can be defined as a so-called behavior macro. Such a macro can be embedded at different positions of a subject behavior specification as often as required. Thus, variations in behavior can be consolidated, and the overall behavior can be significantly simplified.

The brief example of the business trip application is not an appropriate scenario to illustrate here the benefit of the use of macros. Instead, we use an example for order processing. Figure 5.39 contains a macro for the behavior to process customer orders. After placing the “order”, the customer receives an order confirmation; once the “delivery” occurs, the delivery status is updated.

As with the subject, the start and end states of a macro also need to be identified. For the start states, this is done similarly to the subjects by putting black triangles in the top left corner of the respective state box. In our example, “order” and “delivery” are the two correspondingly labeled states. In general, this means that a behavior can initiate a jump to different starting points within a macro.

The end of a macro is depicted by gray bars, which represent the successor states of the parent behavior. These are not known during the course of the macro definition.

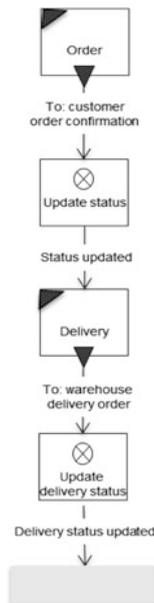


Fig. 5.39 Behavior macro “Order processing”

Figure 5.40 shows a subject behavior in which the modeler uses the macro “order processing” to model both a regular order (with purchase order), as well as a call order.

The icon for a macro is a small table, which can contain multiple columns in the first line for different start states of the macro. The valid start state for a specific case is indicated by the incoming edge of the state transition from the calling behavior. The middle row contains the macro name, while the third row again may contain several columns with possible output transitions, which end in states of the surrounding behavior.

The left branch of the behavioral description refers to regular customer orders. The embedded macro is labeled correspondingly and started with the status “order”, namely through linking the edge of the transition “order accepted” with this start state. Accordingly, the macro is closed via the transition “delivery status updated”.

The right embedding deals with call orders according to organizational frameworks and frame contracts. The macro starts therefore in the state “delivery”. In this case, it also ends with the transition “delivery status updated”.

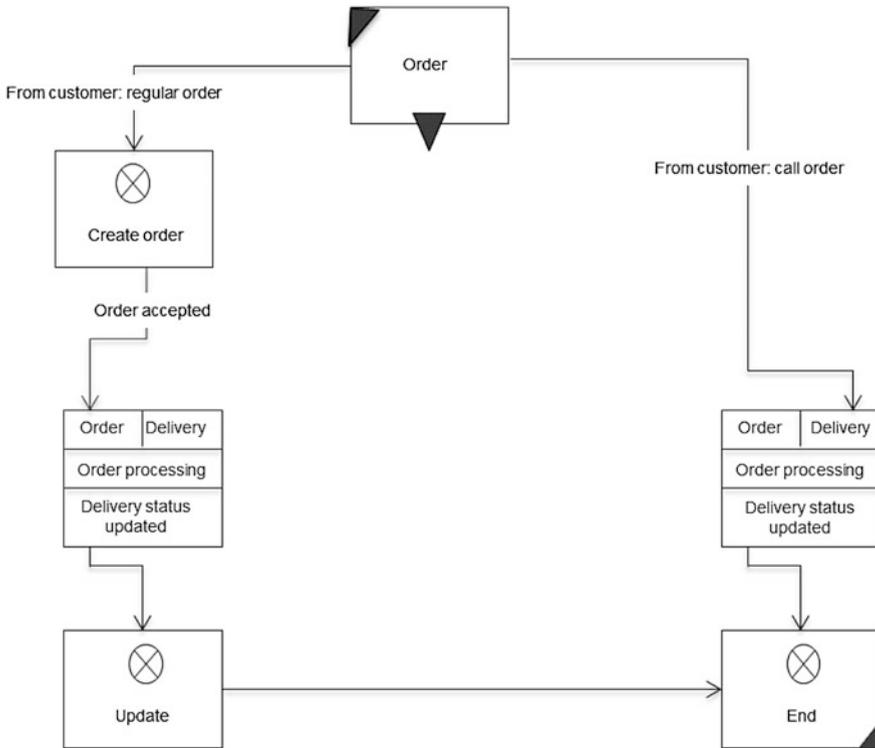


Fig. 5.40 Subject behavior for order processing with macro integration

Similar subject behavior can be combined into macros. When being specified, the environment is initially hidden, since it is not known at the time of modeling.

5.7.3 Behavior Macro Classes

The behavior macros presented in Sect. 5.7.2 enable multiple use of the description of similar sequences of behavior within a subject. There are also situations in which identical behavior sequences are required in several subjects. In order to avoid redundant modeling of this behavior, we introduce so-called behavior macro classes. These are descriptions of behavior that can be included multiple times in different subjects.

When defining a behavior macro class, the subjects involved in communication are not known. We use formal subject names to handle this. They represent subjects as part of internal macro communication. When embedded in a subject, the formal names for the other send and receive operations are replaced by the names of the subjects with which the calling subject communicates corresponding to the macro.

An example of the use of a behavior macro class in the course of modeling is a generic approval process. This runs the same way, regardless of what specific case (business trip request, vacation request, etc.) needs to be handled. In Fig. 5.41, the behavior macro class for the approval process is shown. The formal subject name “approver”, which at runtime contains the concrete subject that should review the request, is set in angle brackets to mark it accordingly.

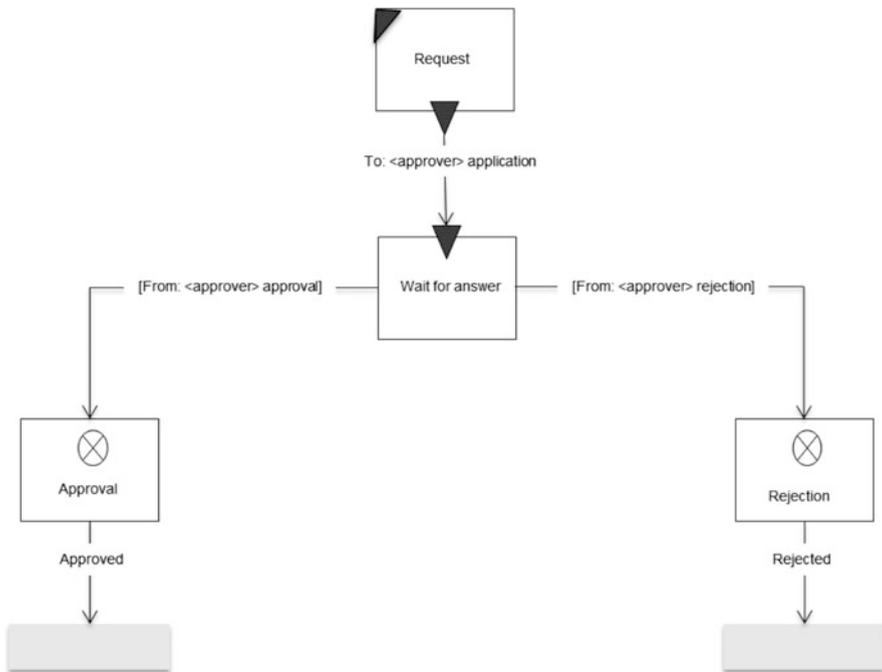


Fig. 5.41 Behavior macro class “request for approval”

Figure 5.42 exemplifies for “request for approval” how a macro of a behavior macro class can be integrated into a subject behavior. The formal name of the subject “<approver>” is replaced at runtime by the subject name “manager”.

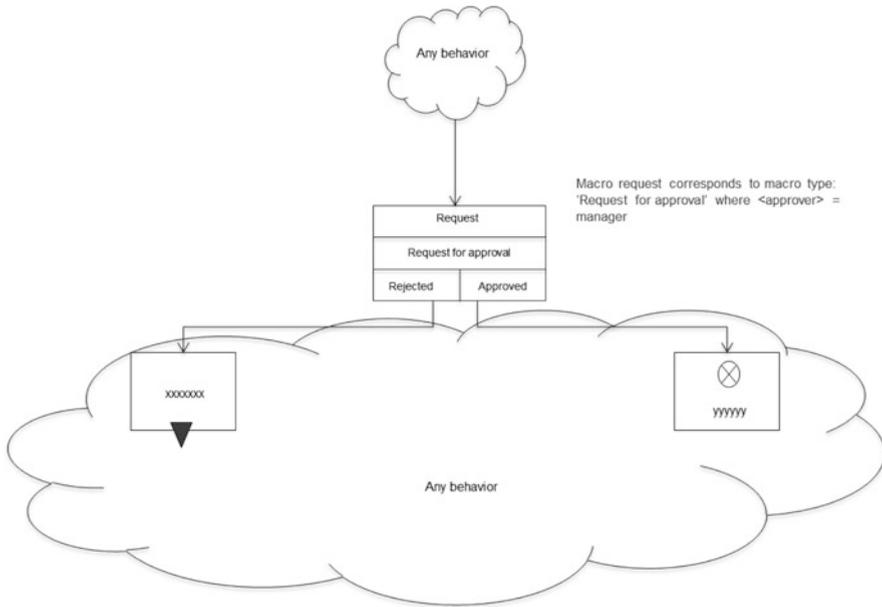


Fig. 5.42 Using a behavior macro class

Behavior macro classes improve the management of processes. For example, if the approval process needs to be fundamentally changed, it is sufficient to adapt the definition of the macro class. Consequently, all processes using this macro class have the revised behavior. However, it has to be ensured that the communication partners of a subject with a modified macro class are compatible to this modified behavior.

Macro classes generalize subject behavior and establish behavior conventions in this way.

5.7.4 Subject Classes

In processes, there are sometimes subjects, which have the same behavior. To avoid redundant description of these subjects, subject classes can be defined.

A subject class is an abstract subject that is assigned a specific subject name at runtime.

As with behavior macro classes, at the time of modeling the subjects involved are not known, since these depend on the respective process. Therefore, also in this case, a formal subject name is used for sending and receiving operations.

As an example, we can use again the approval process. A subject can act in many different contexts as approving instance (“approver”). Examples include business trip or vacation requests, buying a PC, etc. The behavior often follows the pattern shown in Fig. 5.43, which is therefore modeled as a subject class “approver”. Instead of the process-specific subject identifier, the formal name “applicant” set in angle brackets is used for the send and receive states in the subject class.

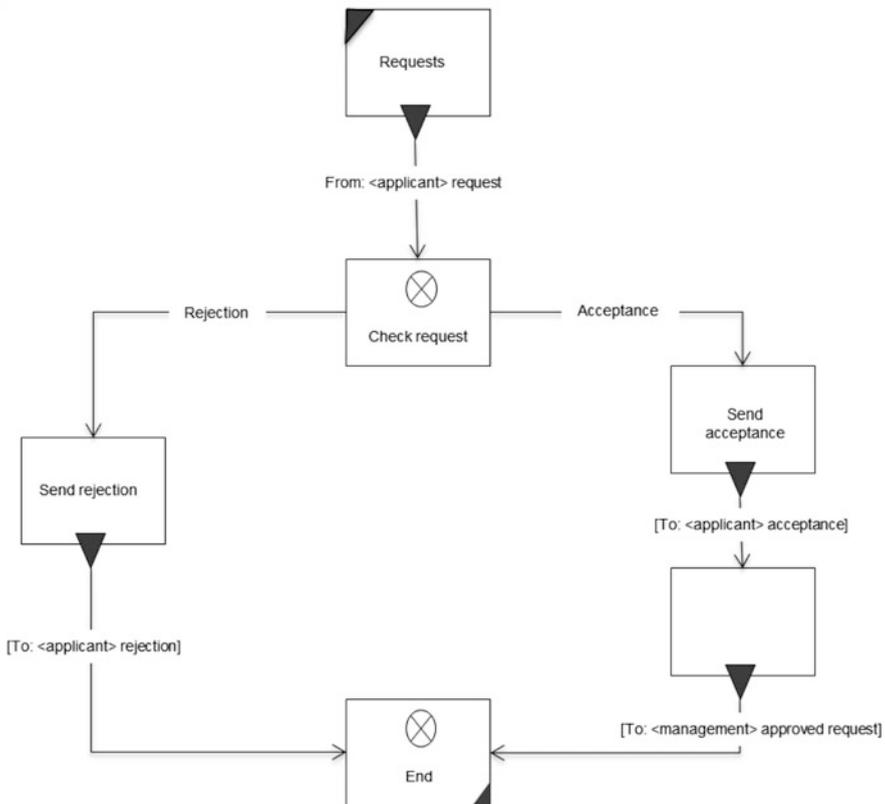


Fig. 5.43 Sample subject class “approver”

Figure 5.44 shows how subject classes can be used in processes. The defined subject class “approver” is used in both the process “business trip application” and in the process “PC purchase”. In the process “business trip application”, it represents the subject “manager” and in the process “PC purchase” the subject “controller”. The formal name of the subject “applicant” is replaced in the case of the business trip application by the subject “employee” and in the case of the PC

purchase by the subject “customer”, respectively. The subject name of “management” is replaced by “manager” in the process “business trip application”, and by “accounting” in the process “PC purchase”.

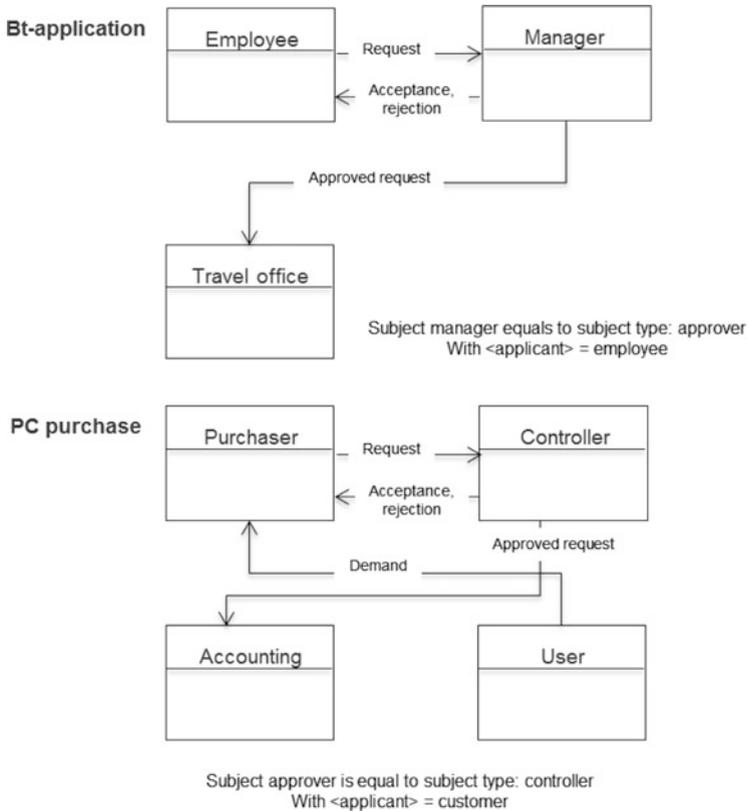


Fig. 5.44 Use of the subject class “approver”

5.7.5 Freedom of Choice

So far, the behavior of subjects has been regarded as a distinct sequence of internal functions, send and receive activities. In many cases, however, the sequence of internal execution is not important.

Certain sequences of actions can be executed overlapping. We are talking about freedom of choice when accomplishing tasks. In this case, the modeler does not specify a strict sequence of activities. Rather, a subject (or concrete entity assigned to a subject) will organize to a particular extent its own behavior at runtime.

The freedom of choice with respect to behavior is described as a set of alternative clauses, which outline a number of parallel paths. At the beginning and end of each alternative, switches are used: a switch set at the beginning means that this

alternative path is mandatory to get started, a switch set at the end means that this alternative path must be completely traversed. This leads to the following constellations:

- Beginning is set/end is set: Alternative needs to be processed to the end.
- Beginning is set/end is open: Alternative must be started but does not need to be finished.
- Beginning is open/end is set: Alternative may be processed, but if so must be completed.
- Beginning is open/end is open: Alternative may be processed but does not have to be completed.

The execution of an alternative clause is considered complete when all alternative sequences, which were begun and had to be completed, have actually been entirely processed and have reached the end operator of the alternative clause.

Transitions between the alternative paths of an alternative clause are not allowed. An alternate sequence starts in its start point and ends entirely within its end point.

Figure 5.45 shows an example for modeling alternative clauses. After receiving an order from the customer, three alternative behavioral sequences can be started, whereby the leftmost sequence, with the internal function “update order” and sending the message “deliver order” to the subject “warehouse”, must be started in any case. This is determined by the “X” in the symbol for the start of the alternative sequences (gray bar is the starting point for alternatives). This sequence must be processed through to the end of the alternative because it is also marked in the end symbol of this alternative with an “X” (gray bar as the end point of the alternative).

The other two sequences may, but do not have to be, started. However, in case the middle sequence is started, i.e., the message “order arrived” is sent to the sales department, it must be processed to the end. This is defined by an appropriate marking in the end symbol of the alternatives (“X” in the lower gray bar as the end point of the alternatives). The rightmost path can be started but does not need to be completed.

The individual actions in the alternative paths of an alternative clause may be arbitrarily executed in parallel and overlapping, or in other words: a step can be executed in an alternative sequence and then be followed by an action in any other sequence. This gives the performer of a subject the appropriate freedom of choice while executing his actions.

In the example, the order can thus first be updated, and then the message “order arrived” sent to sales. Now, either the message “deliver order” can be sent to the warehouse or one of the internal functions, “update sales status” or “collect data for statistics”, can be executed.

The left alternative must be executed completely, and the middle alternative must also have been completed, if the first action (“inform sales” in the example) is executed. It can occur that only the left alternative is processed because the middle one was never started. Alternatively, the sequence in the middle may have already reached its end point, while the left is not yet complete. In this case, the process

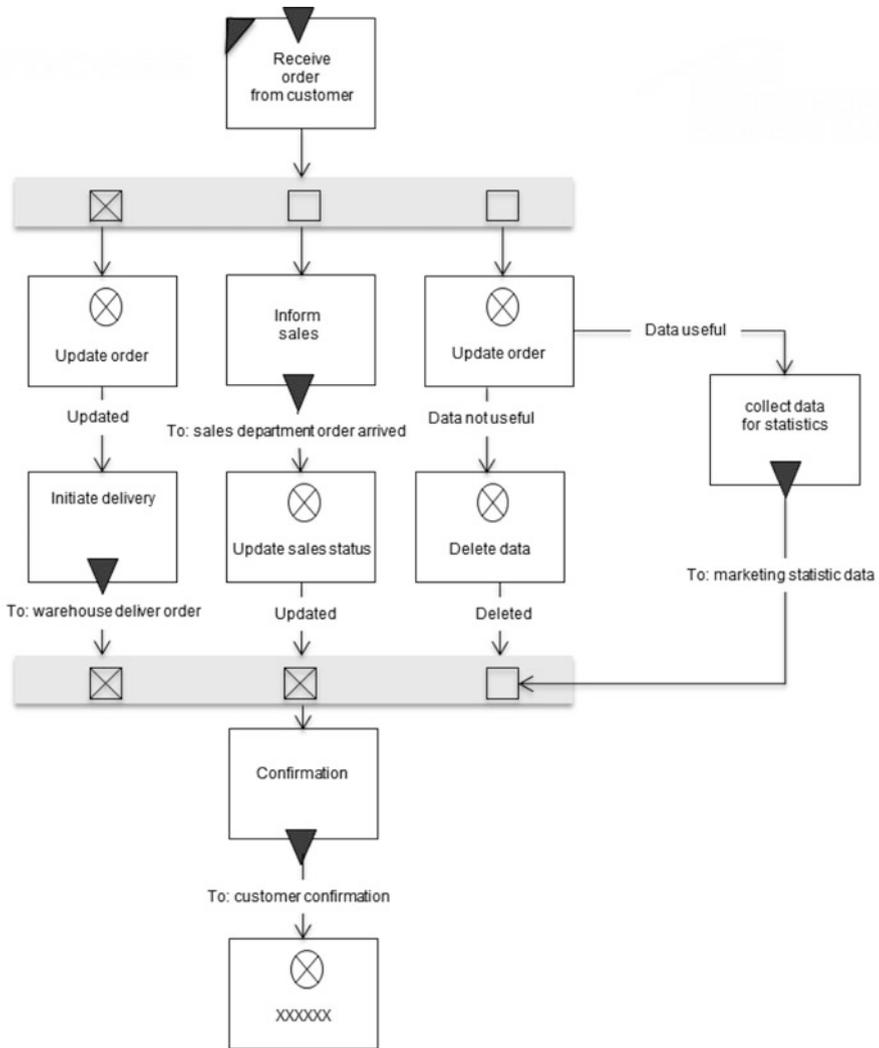


Fig. 5.45 Example of process alternatives

waits until the left one has reached its end point. Only then will the state “confirmation” be reached in the alternative clause. The right branch neither needs to be started nor to be completed. It is therefore irrelevant for the completion of the alternative construct.

The leeway for freedom of choice with regard to actions and decisions associated with work activities can be represented through modeling the various alternatives—situations can thus be modeled according to actual regularities and preferences.

5.7.6 Exception Handling

Handling of an exception (also termed message guard, message control, message monitoring, and message observer) is a behavioral description of a subject that becomes relevant when a specific, exceptional situation occurs while executing a subject behavior specification. It is activated when a corresponding message is received, and the subject is in a state in which it is able to respond to the exception handling. In such a case, the transition to exception handling has the highest priority and will be enforced.

Exception handling is characterized by the fact that it can occur in a process in many behavior states of subjects. The receipt of certain messages, e.g., to abort the process, always results in the same processing pattern. This pattern would have to be modeled for each state in which it is relevant. Exception handlings cause high modeling effort and lead to complex process models, since from each affected state a corresponding transition has to be specified. In order to prevent this situation, we introduce a concept similar to exception handling in programming languages or interrupt handling in operating systems.

To illustrate the compact description of exception handlings, we use again the service management process with the subject “service desk” introduced in Sect. 5.6.5. This subject identifies a need for a business trip in the context of processing a customer order—an employee needs to visit the customer to provide a service locally. The subject “service desk” passes on a service order to an employee. Hence, the employee issues a business trip request. In principle, the service order may be canceled at any stage during processing up to its completion. Consequently, this also applies to the business trip application and its subsequent activities.

Below, it is first shown how the behavior modeling looks without the concept of exception handling. The cancellation message must be passed on to all affected subjects to bring the process to a defined end. Figure 5.46 shows the communication structure diagram with the added cancellation messages to the involved subjects.

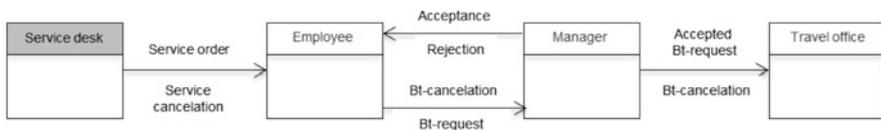


Fig. 5.46 Communication structure diagram (CSD) of the business trip application process including cancellation messages

A cancellation message can be received by the employee either while filling out the application, or while waiting for the approval or rejection message from the manager. With respect to the behavior of the subject “employee”, the state “response received from manager” must also be enriched with the possible input message containing the cancellation and the associated consequences (see Fig. 5.47). The verification of whether filing the request is followed by a cancellation, is modeled through a receive state with a timeout. In case the timeout is zero,

there is no cancellation message in the input pool and the business trip request is sent to the manager. Otherwise, the manager is informed of the cancellation and the process terminates for the subject “employee”.

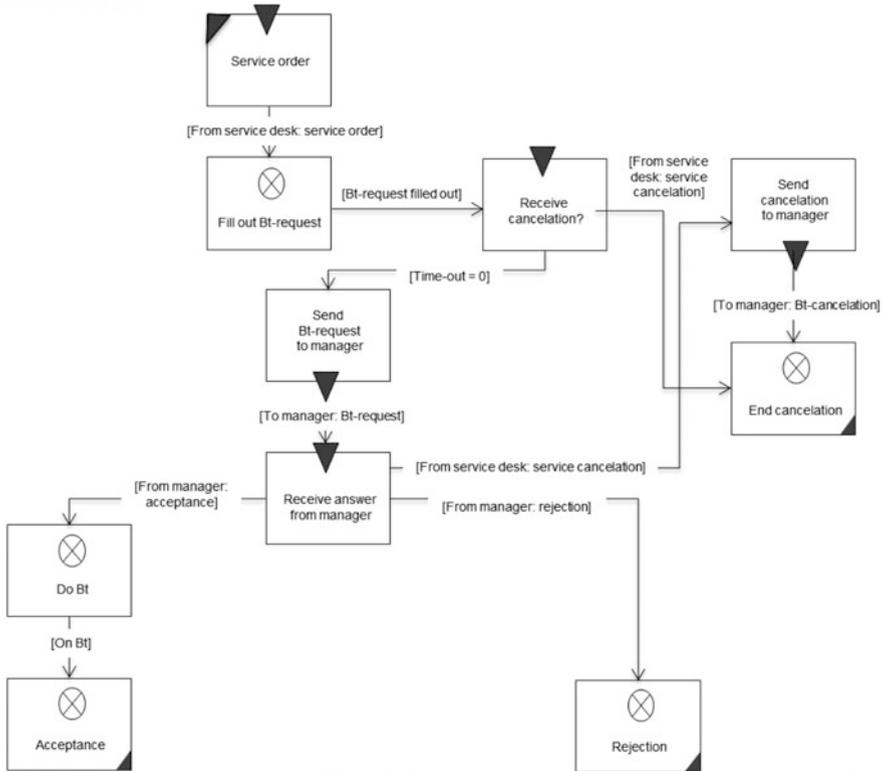


Fig. 5.47 Handling the cancelation message using existing constructs (without the concept of exception handling)

A corresponding adjustment of the behavior must be made for each subject which can receive a cancelation message, including the manager, the travel office, and the interface subject “travel agent”.

This relatively simple example already shows that taking such exception messages into account can quickly make behavior descriptions confusing to understand. The concept of exception handling, therefore, should enable supplementing exceptions to the default behavior of subjects in a structured and compact form. Figure 5.48 shows how such a concept affects the behavior of the employee.

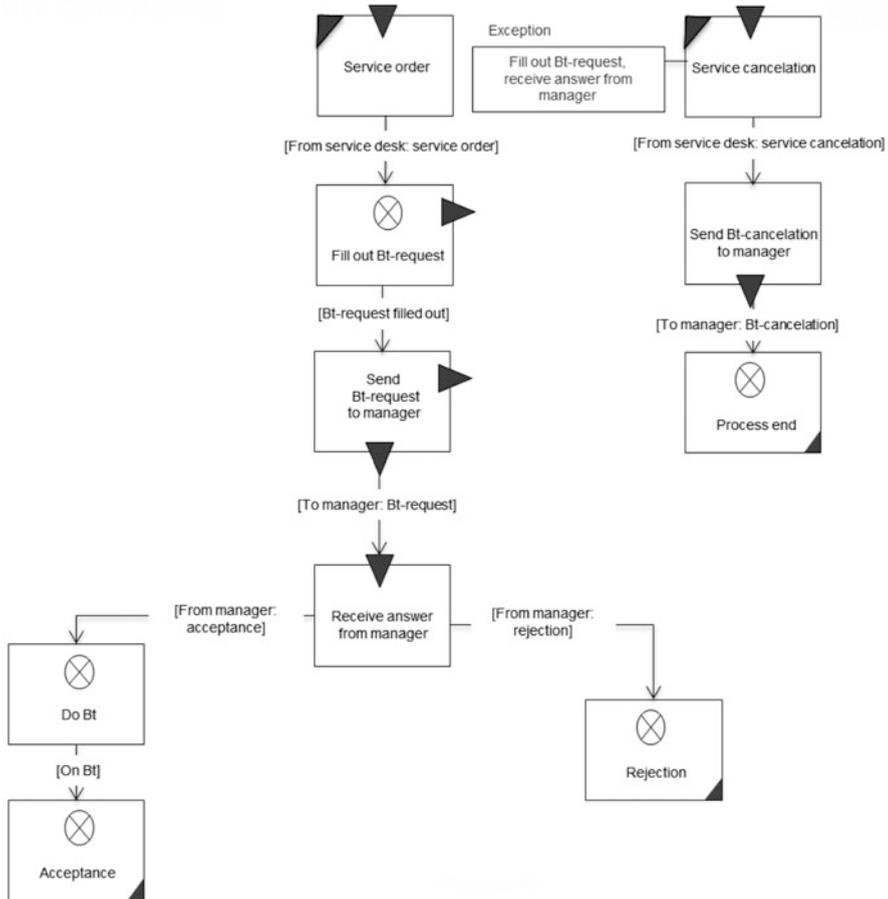


Fig. 5.48 Behavior of subject “employee” with exception handling

Instead of, as shown in Fig. 5.47, modeling receive states with a timeout zero and corresponding state transitions, the behavioral description is enriched with the exception handling “service cancellation”. Its initial state is labeled with the states from which it is branched to, once the message “service cancellation” is received. In the example, these are the states “fill out Bt-request” and “receive answer from manager”. Each of them is marked by a triangle on the right edge of the state symbol. The exception behavior leads to an exit of the subject, after the message “service cancellation” has been sent to the subject “manager”.

A subject behavior does not necessarily have to be brought to an end by an exception handling; it can also return from there to the specified default behavior. Exception handling behavior in a subject may vary, depending on from which state or what type of message (cancellation, temporary stopping of the process,

etc.) it is called. The initial state of exception handling can be a receive state or a function state.

Messages, like “service cancelation”, that lead to exception handling always have higher priority than other messages. This is how modelers express that specific messages are read in a preferred way. For instance, when the approval message from the manager is received in the input pool of the employee, and shortly thereafter the cancelation message, the latter is read first. This leads to the corresponding abort consequences.

Since now additional messages can be exchanged between subjects, it may be necessary to adjust the corresponding conditions for the input pool structure. In particular, the input pool conditions should allow storing an interrupt message in the input pool.

To meet organizational dynamics, exception handling and extensions are required. They allow taking not only discrepancies, but also new patterns of behavior, into account.

5.7.7 Behavior Extensions

When exceptions occur, currently running operations are interrupted. This can lead to inconsistencies in the processing of business objects. For example, the completion of the business trip form is interrupted once a cancelation message is received, and the business trip application is only partially completed. Such consequences are considered acceptable due to the urgency of cancelation messages. In less urgent cases, the modeler would like to extend the behavior of subjects in a similar way, however, without causing inconsistencies. This can be achieved by using a notation analogous to exception handling. Instead of denoting the corresponding diagram with “exception”, it is labeled with “extension”.

Behavior extensions enrich a subject’s behavior with behavior sequences that can be reached from several states equivocally.

For example, the employee may be able to decide on his own that the business trip is no longer required and withdraw his trip request. Figure 5.49 shows that the employee is able to cancel a business trip request in the states “send business trip request to manager” and “receive answer from manager”. If the transition “withdraw business trip request” is executed in the state “send business trip request to manager”, then the extension “F1” is activated. It leads merely to canceling of the application. Since the manager has not yet received a request, he does not need to be informed.

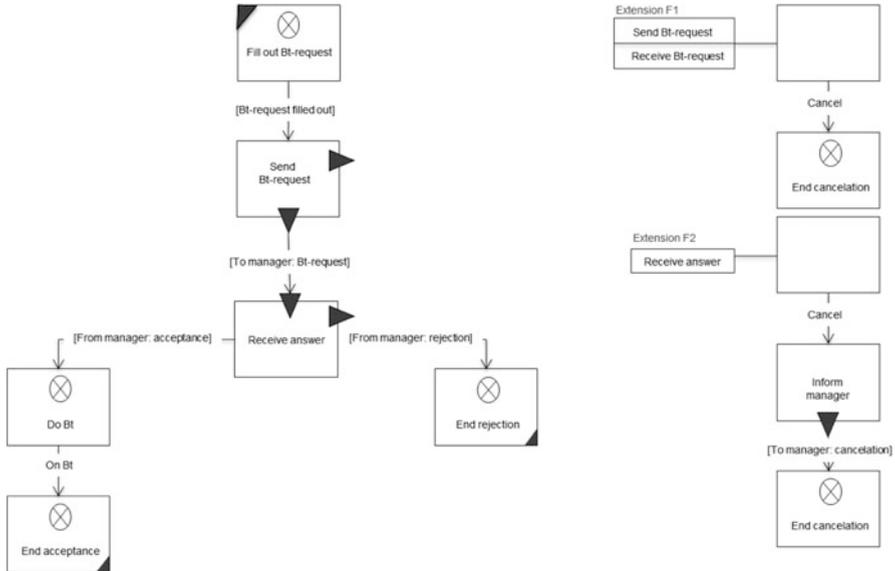


Fig. 5.49 Subject behavior of employee with behavior extensions

In case the employee decides to withdraw the business trip request in the state “receive answer from manager”, then extension “F2” is activated. Here, first the supervisor is informed, and then the business trip is canceled.

5.7.8 Additional Semantics

Often it is necessary to record further information in a process, explaining what specific considerations have influenced modeling. This is possible with the so-called additional semantics. It allows specification of reasons for the existence of subjects or conditions to be added within the behavioral description.

For example, it may be necessary for reasons of compliance to include additional subjects in a process and to introduce additional interactions between subjects, in order to satisfy certain external or internal rules. Such requirements can, e.g., result from quality management systems like ISO 9001, environmental regulations, or rules affecting Internal Control Systems (ICS), such as the Sarbanes-Oxley Act (SOX) (see Sect. 3.6.4). They usually cause higher communication overhead and thus, often more complex processes. This poses the risk that the additionally modeled subjects and states are removed in the course of a subsequent optimization because the optimizer might no longer know the reasons why certain subjects or communication patterns had been installed. Therefore, such subjects and states should be provided with appropriate references to those regulations that justify their introduction.

Figure 5.50 shows the existing business trip application process with the addition of an internal control for international business trips. This states that, effective immediately, such trips must be approved by management, to better control travel costs in difficult economic times, and to reduce them where appropriate. In the modified process, there is now a new subject “management board”. This subject will receive for approval all submitted requests for international travel. Its specification is therefore enriched with a corresponding comment, pointing out that it was introduced in the process for reasons of compliance in conjunction with the internal control system (ICS).

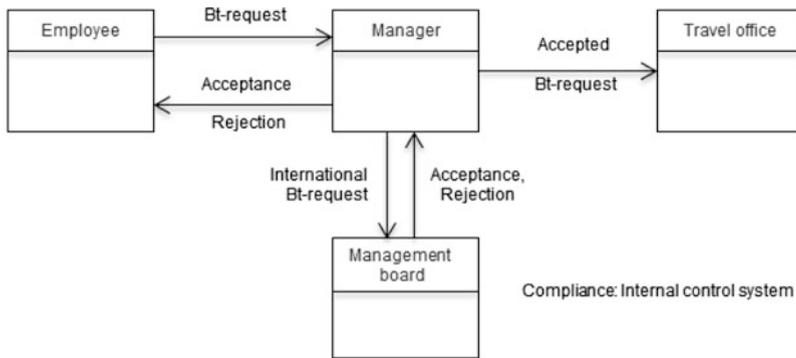


Fig. 5.50 Revised business trip application process including the management review of requests for international travel and their justification

Due to the introduction of the subject “management board” in the business trip application process, the behavior of the subject “manager” also needs to be adapted. The manager first checks whether an application has been made for international travel. If this is not the case, he will proceed as previously specified. In case of an international travel request, the trip request is forwarded for consideration to the board. This is specified by introducing the send state “request board review” and the corresponding receive state “board response”. Both states are marked with “ICS request”. Figure 5.51 shows the modified behavior of the subject “manager”.

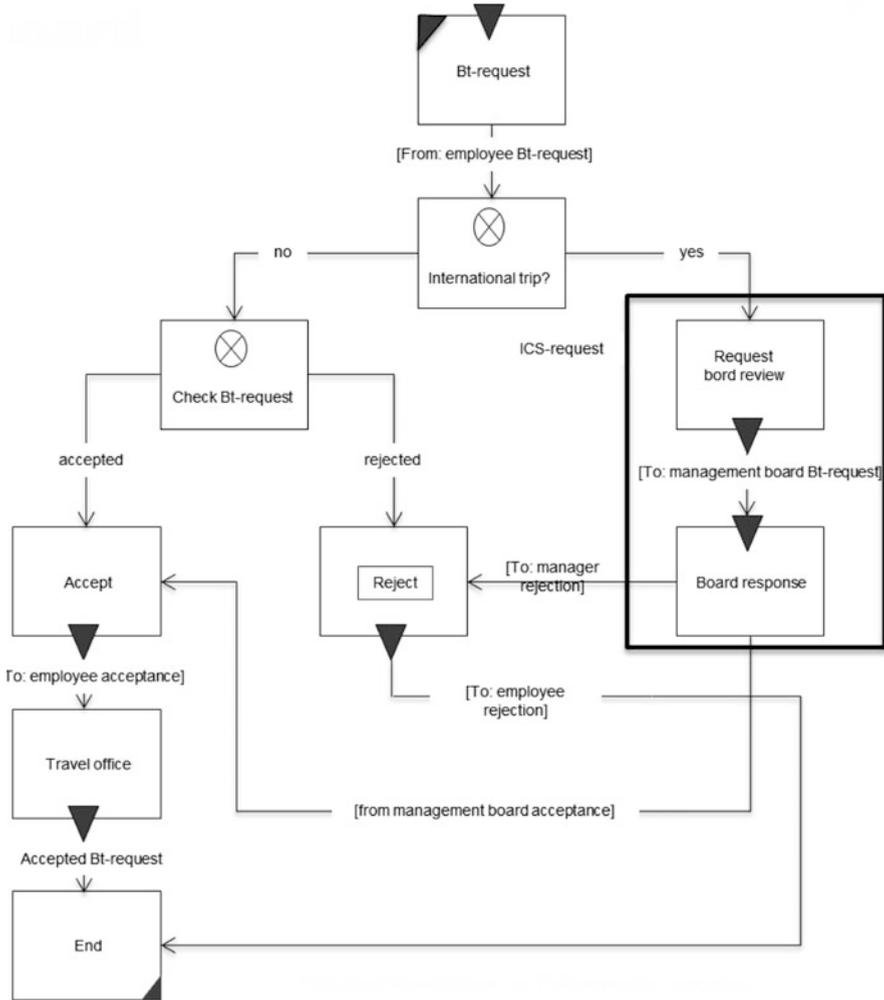


Fig. 5.51 Communication of manager with board

Although S-BPM-models are constructed in a systematic way (Who is involved? Who interacts with whom/with what? What information needs to be exchanged to perform tasks?), it is often necessary to provide additional information, on how a coherent result of the work can be achieved—this is when you should use the S-BPM feature “Additional Semantics”.

References

Fleischmann, A.: Distributed Systems – Software Design and Implementation, Heidelberg 1995.
Grässle, P.; Baumann, H.; Baumann, P.: UML 2.0 projektorientiert – Geschäftsprozessmodellierung, IT-System-Spezifikation und Systemintegration mit der UML, Bonn 2004.

Open Access. This chapter is distributed under the terms of the Creative Commons Attribution Non-commercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.