Chapter 2
Introduction to Business Processes, BPM, and BPM Systems

This chapter provides a basic overview on business processes. In particular it concentrates on the actual definition and characterization of processes, and on the different languages that can be used to describe them. Some notions on the main components of BPM systems conclude this chapter.

2.1 Introduction to Business Processes

It is very common, in industrial settings, that the performed activities are repetitive and have several persons involved. In these cases, it is very useful to define a standard procedure that everyone can follow. A *business process*, essentially, is the definition of such "standard procedure".

Since the process aims at standardizing and optimizing the activities of the company, it is important to keep the process up to date and as flexible as possible, in order to meet the market requirements and the business objectives.

Business Process

There are several definitions of "business process". The most influential ones are reported in [91]. The first, presented in [76] by Hammer and Champy, states that a business process is:

> A collection of activities that takes one or more kinds of input and creates an output that is of value to the customer. A business process has a goal and is affected by events occurring in the external world or in other processes.

In another work, by Davenport [43], a business process is defined as:

> A structured, measured set of activities designed to produce a specified output for a particular customer or market. […] A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action.

A. Burattin: *Process Mining Techniques in Business Environments*, LNBIP 207,
DOI 10.1007/978-3-319-17482-2_2

In both cases, the main focus is on the "output" of the actions that must take place. The problem is that there is no mention to the originators (i.e., the executors) of such activities and how they are interoperating.

In [113], a business process is viewed as something that: (*a*) contains purposeful activities; (*b*) is carried out, collaboratively, by a group of humans and/or machines; (*c*) often crosses functional boundaries; (*d*) is invariably driven by the outside world. van der Aalst, Weijters and Medeiros, in [167], gave attention to the originators of the activities:

> By process we mean the way an organization arranges their work and resources, for instance the order in which tasks are performed and which group of people are allowed to perform specific tasks.

Ko, in his "*A Computer Scientist's Introductory Guide to Business Process Management*" [91], gave his own definition of business process:

> A series or network of value-added activities, performed by their relevant roles or collaborators, to purposefully achieve the common business goal.

A formal definition of business process is presented by Agrawal, Gunopulos and Leymann in [3]:

> A business process P is defined as a set of activities $V_P = \{V_1, \ldots, V_n\}$, a directed graph $G_P = (V_P, E_P)$, an output function o_P: $V_P \rightarrow \mathbb{N}^k$ and $\forall (u, v) \in E_P$ a boolean function $f_{(u,v)} = \mathbb{N}^k \rightarrow \{0, 1\}$.

In this case, the process is constructed in the following way: for every completed activity u, the value $o_P(u)$ is calculated and then, for every other activity v, if $f_{(u,v)}(o_P(u))$ is "true", v can be executed. Of course, such definition of business process is hard to be handled by business people, but is useful for formal modeling purposes.

More general definitions are given by standards and manuals. For example, the glossary of the BPMN manual [112] describes a process as "*any activity performed within a company or organization*". The ISO 9000 [125] presents a process as:

> A set of activities that are interrelated or that interact with one another. Processes use resources to transform inputs into outputs. Processes are interconnected because the output from one process becomes the input for another process. In effect, processes are "glued" together by means of such input output relationships.

Right now, no general consensus has been reached on a specific definition. This lack is due to the size of the field and to the different aspects that every definition aims to point out.

In the context of this work, it is not important to fix one definition: each definition highlights some aspects of the global idea of business process. The most important issues, that should be covered by a definition of business process, are:

1. there is a finite set of **activities** (or **tasks**) and their executions are partially ordered (it's important to note that not all the activities are mandatory in all the process executions);
2. each activity is executed by one or more **originators** (can be humans or machines or both);

3. the execution of every activity produces some **output** (as a general notion, with no specific requirement: it can be a document, a service or just a "flag of state" set to "executed") that can be used by the following activity.

This is not intended to be another new definition of business process, but it's just a list of the most important issues that emerge from the definitions reported above.

Representation of Business Processes

Closely related to Business Processes is Business Process Management (BPM). van der Aalst, ter Hofstede and Weske, in [161], define BPM as:

> Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information.

From this definition, it clearly emerges that two of the most important aspects of BPM are **design** and **documentation**. The importance of these two tasks is clear if one thinks about the need to communicate some specific information on the process that has been modeled. The main benefits of adopting a clear business model can summarized in the following list:

- it is possible to increase the visibility of the activities, that allows the identification of problems (e.g. bottlenecks) but also areas of potential optimization and improvement;
- grouping the activities in "department" and grouping the persons in "roles", in order to better define duties, auditing and assessment activities.

For the reasons just explained, some characteristics of a process model can be identified. The most important one is that a model should be **unambiguous**, in the sense that the process is precisely described without leaving uncertainties to the potential reader.

There are many languages that allow the modeling of systems and business processes. The most used formalisms for the specification of business processes have in common to be graph-based representations, so that nodes, typically, represent the process tasks (or, in some notations, also the states and the possible events of the process); arcs represent ordering relations between tasks (for example, an arc from node n_1 to n_2 represents a dependency in the execution so that n_2 is executed only after n_1). Two of the most important graph based languages are: Petri nets [111, 118, 146, 188] and BPMN [112][1].

[1] Another language for the definition of "processes" is, for example, the Π-calculus [115, 185]: a mathematical framework for the definition of processes whose connections vary based on the interaction. Actually, it is not used in business contexts and by non-expert users because of its complexity. With other similar languages (such as Calculus of Communicating Systems, CCS and Communicating Sequential Processes, CSP) the situation is similar: in general, mathematical approaches are suitable for the definition of interaction protocols or for the analysis of procedures (such as deadlock identification) but not for business people.

2.1.1 Petri Nets

Petri nets, proposed in 1962 in the Ph.D. thesis of Carl Adam Petri [119], constitute a graphical language for the representation of a process. In particular, a Petri net is a bipartite graph, where two types of nodes can be defined: transitions and places. Typically, transitions represent activities that can be executed, and places represent states (intermediate or final) that the process can reach. Edges, always directed, must connect a place and a transition, so an edge is not allowed to connect two places or two transitions. Each place can contain a certain number of tokens and the distribution of the tokens on the network is called "marking". In Fig. 2.1 a small Petri net is shown; circles represent places, squares represent transitions.

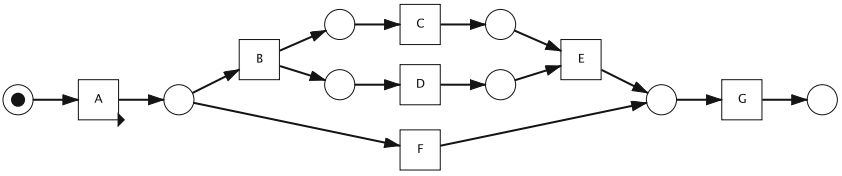


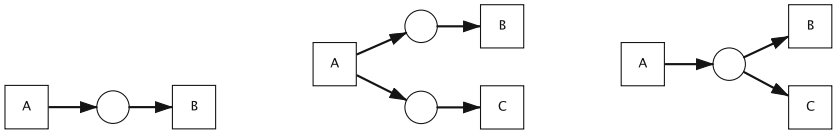
Fig. 2.1 Petri net example, where some basic patterns can be observed: the "AND-split" (activity B), the "AND-join" (activity E), the "OR-split" (activity A) and the "OR-join" (activity G).

Petri nets have been studied in depth from many points of view, such as from their clear semantic to a certain number of possible extensions (such as time, color, …). A formal definition of Petri net, as presented, for example, in [145], is the following:

Definition 2.1 (Petri net). A Petri net is a tuple (P, T, F) where: P is a finite set of places; T is a finite set of transitions, such that $P \cap T = \emptyset$, and $F \subset (P \times T) \cup (T \times P)$ is a set of directed arcs, called flow relation.

The "dynamic semantic" of a Petri net is based on the "firing rule": a transition can *fire* (i.e., be executed) if all its "input places" (places with edges entering into the transition) contain at least one token. The firing of a transition generates one token for all its "output places" (places with edges exiting from the transition). The distribution of tokens among the places of a net, at a certain time, is called "marking". With this semantic, it is possible to model many different behaviors, for example, in Fig. 2.2, three basic templates are proposed. The sequence template describes the causal dependency between two activities (in the figure example, activity B requires the execution of A); the AND template represents the concurrent branching of two or more flows (in the figure example, once A is terminated, B and C can start, in no specific order and concurrently); the XOR template defines the mutual exclusion of two or more flows (in the figure example, once A is terminated, only B or C can start). Figure 2.3 proposes the same process of Fig. 2.2 with a different marking (after the execution of activities A, B and C).

An important subclass of Petri nets is the Workflow nets (WF-net), whose most important characteristic is to have a dedicated "start" and "end":



(a) Sequence template. (b) AND template. (c) XOR template.

Fig. 2.2 Some basic workflow templates that can be modeled using Petri net notation.

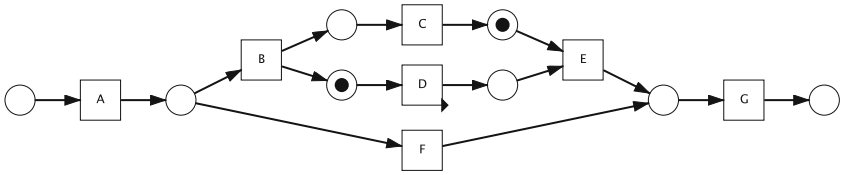


Fig. 2.3 The marked Petri net of Fig. 2.1, after the execution of activities A, B and C. The only enabled transition, at this stage, is D.

Definition 2.2 (WF-net). A WF-net is a Petri net $N = (P, T, F)$ such that:

a. P contains a place i with no incoming arcs (the starting point of the process);
b. P contains a place o with no outgoing arcs (the end point of the process);
c. if we consider $\bar{t} \notin P \cup T$, and we use it to connect o and i (so to obtain the so called "short-circuited" net: $\overline{N} = (P, T \cup \{\bar{t}\}), F \cup \{(o, \bar{t}), (\bar{t}, i)\}$), the new net is strongly connected (i.e. there is a direct path between any pair of nodes).

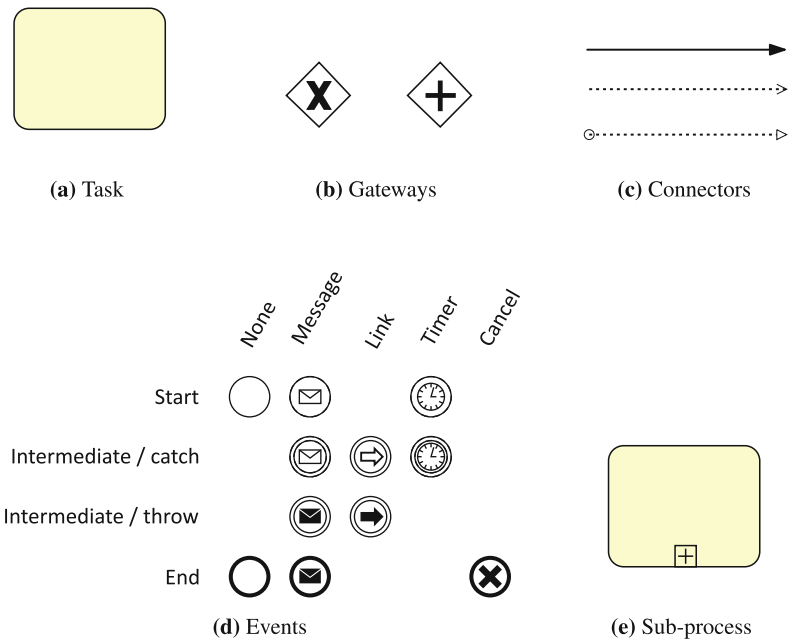
2.1.2 BPMN

BPMN (Business Process Modeling and Notation) [112] is the result of an agreement among multiple tool vendors, that agreed on the standardization of a single notation. For this reason, now it is used in many real cases and many tools adopt it daily. BPMN provides a graphical notation to describe business processes, which is, at the same time, intuitive and powerful (it is able to represent complex process structure). It is possible to map a BPMN diagram to an execution language, BPEL (Business Process Execution Language).

The main components of a BPMN diagram, presented in Fig. 2.4, are:

Events: defined as "*something that "happens" during the course of a process*"; typically they have a cause (trigger) and an impact (result). Each event is represented with a circle (containing an icon, to specify some details), as in Fig. 2.4(d). There are three types of events: *start* (single narrow border), *intermediate* (single thick border) and *end* (double narrow border).

Activities: this is the generic term that identifies the work done by a company. In the graphical representation they are identified as rounded rectangles. There are few



(a) Task **(b)** Gateways **(c)** Connectors

(d) Events **(e)** Sub-process

Fig. 2.4 Example of some basic components, used to model a business process using BPMN notation.

types of activity like tasks (a single unit of work, Fig. 2.4(a)) and subprocesses (used to hide different levels of abstraction of the work, Fig. 2.4(e)).

Gateway: structure used to control the divergences and convergences of the flow of the process (fork, merge and join). An internal marker identifies the type of gateway, like "exclusive" (Fig. 2.4(b), on the left), "event based", "inclusive", "complex" and "parallel" (Fig. 2.4(b), on the right).

Sequence and Message Flows and Associations: connectors between components of the graph. A sequence flow (Fig. 2.4(c), top) is used to indicate the order of the activities. Message flow (Fig. 2.4(c), bottom) shows the flow of the messages (as they are prepared, sent and received) between participants. Associations (Fig. 2.4(c), middle) are used to connect artifacts with other elements of the graph.

Beyond the components just described, there are also other entities that can appear in a BPMN diagram, such as artifacts (e.g. annotations, data objects) and swimlanes.

Figure 2.5 proposes a simple process fragment. It starts on Friday, executes two activities (in the figure, "Receive Issue List" and then "Review Issue List") and then checks if a condition is satisfied ("Any issues ready"); if this is the case, a discussion can take place a certain number of times ("Discussion Cycle" sub process), otherwise the process is terminated (and the "End event" is reached, marked as a circle with the bold border). There are, moreover, intermediate events (marked with the double border): the one named *A* is a "throw event" (if it is fired, the flow continues to the

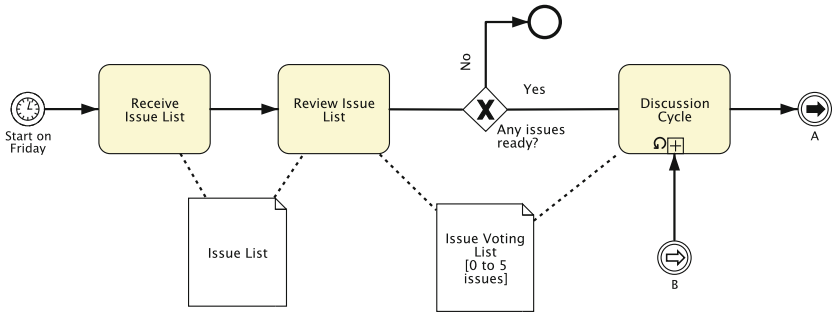


Fig. 2.5 A simple process fragment, expressed as a BPMN diagram. Compared to a Petri net (as in Fig. 2.1), it contains more information and details but it is more ambiguous.

intermediate catch event, named *A*, somewhere in the process but not represented in this figure); the *B* is a "catch event" (it waits until a throw events fires its execution).

2.1.3 YAWL

YAWL (Yet Another Workflow Language) [159] is a workflow language born from a rigorous analysis of the existing workflow patterns [160].

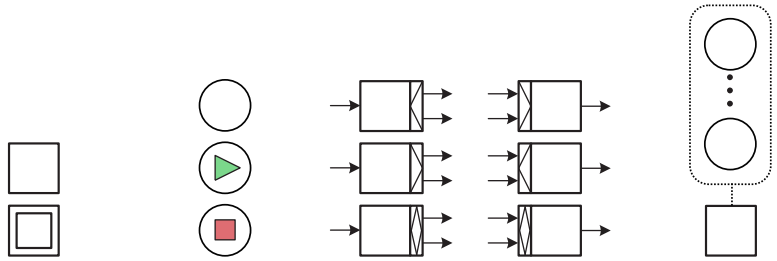
The starting point for the design of this language is the identification of the differences between many languages: out of this, authors collected a complete set of workflow patterns. This set of possible behaviors inspired authors to develop YAWL, which starts from Petri net and adds some mechanisms to allow a "*more direct and intuitive support of the workflow patterns identified*" [160]. However, as authors stated, YAWL is not a "macro" package on top of high-level Petri nets: it is possible to map a YAWL model to any other Turing complete language.

Figure 2.6 presents the main components of a YAWL process. The main components of a YAWL model are:

Task: represents an activity, as in Fig. 2.6(a). It is possible to execute multiple instances of the same task at the same time (so to have many instances of the process running in parallel). Composite tasks are used to define hierarchical structure: a composite task is a container of another YAWL model.

Conditions: a condition Fig. 2.6(b) in YAWL has the same meaning of "place" for Petri nets (i.e. the current state of the process). There are two special conditions, i.e., "start" (with a triangle inscribed) and "end" (with a square inscribed), like for WF-nets (Definition 2.2).

Splits and Joins: a task can have a particular split/join semantic. In particular, it is possible to have tasks with an AND (whose behavior is the same of the Petri



(a) Atomic and composite tasks. **(b)** Conditions (general, start, end). **(c)** Splits and join tasks (AND, XOR, OR). **(d)** Cancelation area.

Fig. 2.6 Main components of a business process modeled in YAWL.

net case, presented in Fig. 2.2(b)), XOR (same as Petri net, Fig. 2.2(c)) or OR semantic. In the last case one or more outgoing arcs are executed[2].

Cancellation Areas: all the tokens in elements within a cancellation area (the dotted area in Fig. 2.6(d)), are removed after the activation of the corresponding task (whose enabling does not depend on the tokens on the cancellation area).

2.1.4 Declare

Imperative process modeling languages such as BPMN, Petri nets, etc., are very useful in environments that are stable and where the decision procedures can be predefined. Participants can be guided according to such process models. However, they are less appropriate for environments that are more variable and that require more flexibility. Consider, for instance, a doctor in a hospital dealing with a variety of patients that need to be handled in a flexible manner. Nevertheless, some general regulations and guidelines can be followed. In such cases, *declarative* process models are more effective than the imperative ones [122, 156, 187]. Instead of explicitly specifying all possible sequences of activities in a process, declarative models implicitly define the allowed behavior of the process with constraints, i.e., rules that must be followed during execution. In comparison to imperative approaches, which produce "closed" models (what is not explicitly specified is forbidden), declarative languages are "open" (everything that is not forbidden is allowed). In this way, models offer flexibility and at the same time remain compact.

While in imperative languages designers tend to forget incorporating some possible scenarios (e.g., related to exception handling), in declarative languages, designers tend to forget certain constraints. This leads to underspecification rather than overspecification, i.e., people are expected to act responsibly and are free to select scenarios that may seem out-of-the-ordinary at first sight.

[2] In the case of OR-join, the semantic is a bit more complex: the system needs only one input token, however if more then one token is coming, the OR-join synchronizes (i.e. waits) them.

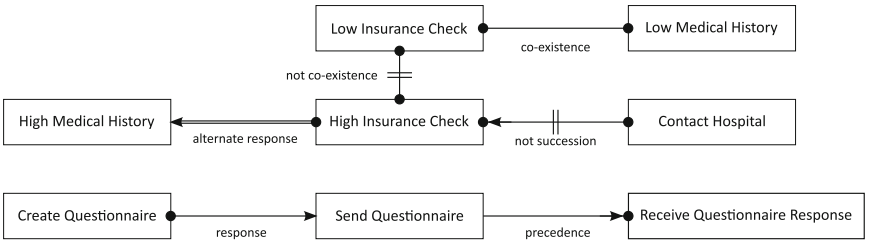


Fig. 2.7 Declare model consisting of six constraints and eight activities.

Figure 2.7 shows a simple Declare model [120, 121] with some illustrative constraints for an insurance claim process. The model includes eight activities (depicted as rectangles, e.g., *Create Questionnaire*) and six constraints (shown as connectors between the activities, e.g., *not co-existence*). The *not co-existence* constraint indicates that *Low Insurance Check* and *High Insurance Check* can never coexist in the same trace. On the other hand, the *co-existence* constraint indicates that if *Low Insurance Check* and *Low Medical History* occur in a trace, they always co-exist. If *High Medical History* is executed, *High Insurance Check* is eventually executed without other occurrences of *High Medical History* in between. This is specified by the *alternate response* constraint. Moreover, the *not succession* constraint means that *Contact Hospital* cannot be followed by *High Insurance Check*. The *precedence* constraint indicates that, if *Receive Questionnaire Response* is executed, *Send Questionnaire* must be executed before (but if *Send Questionnaire* is executed this is not necessarily followed by *Receive Questionnaire Response*). Finally, if *Create Questionnaire* is executed, it is eventually followed by *Send Questionnaire* as indicated by the *response* constraint.

More details on the Declare language will be provided in Subsect. 15.2.1.

2.1.5 Other Formalisms

The language for the definition of business processes, briefly presented in the previous sections, are only a very small fragment of all the available ones. In Table 2.1 some standards are proposed, with their background (either academic or industrial), the type of notation they adopt, if they are standardized somehow, and their current status.

2.2 Business Process Management Systems

It is interesting to distinguish, from a technological point of view, business process design and business process modeling: the first refers to the overall process design (and all its activities); the latter refers to the actual way of representing the process (from a "language" point of view).

In the Gartner's position document [80], a software is defined "BPM-enabled" if allows to work on three parts: integration, runtime environment and rule engine.

Table 2.1 Extraction from Table 2 of [91] where some prominent BPM standards, languages, notations and theories are classified.

Language	Background	Notation	Standardized	Current status
BPDM	Industry	Interchange	Yes	Unfinished
BPEL	Industry	Execution	Yes	Popular
BPML	Industry	Execution	Yes	Obsolete
BPMN	Industry	Graphical	Yes	Popular
BPQL	Industry	Diagnosis	Yes	Unfinished
BPRI	Industry	Diagnosis	Yes	Unfinished
ebXML BPSS	Industry	B2B	Yes	Popular
EDI	Industry	B2B	Yes	Stable
EPC	Academic	Graphical	No	Legacy
Petri nets	Academic	Theory/Graphical	N.A.	Popular
Π-Calculus	Academic	Theory/Execution	N.A.	Popular
Rosetta-Net	Industry	B2B	Yes	Popular
UBL	Industry	B2B	Yes	Stable
UML A.D.	Industry	Graphical	Yes	Popular
WSCI	Industry	Execution	Yes	Obsolete
WSCL	Industry	Execution	Yes	Obsolete
WS-CDL	Industry	Execution	Yes	Popular
WSFL	Industry	Execution	No	Obsolete
XLANG	Industry	Execution	No	Obsolete
XPDL	Industry	Execution	Yes	Stable
YAWL	Academic	Graphical/Execution	No	Stable

When all these aspects are provided, the system is called "BPMS". These aspects are provided if the system contains:

- an *orchestration engine*, that coordinates the sequencing of activities according to the designed flow and rules;
- a *business intelligence* and *analysis tools*, that analyze data produced during the executions. An example of this kind of tools is the Business Activity Monitoring (BAM) that provides real-time alerts for a proactive approach;
- a *rule engine*, that simplifies the changes to the process rules and provides more abstraction from the policies and from the decision tables, allowing more flexibility;
- a *repository* that stores process models, components, documents, business rules and all the information required for the correct execution of the process;
- tools for *simulation and optimization* of the process, that allow the designer to compare possible new process models with the current one in order to get an idea of the possible impact into the current production environment;
- an *integration* tool, that links the process model to other components in order to execute the process activities.

From a more pragmatic perspective, the infrastructure that seems to be the best candidate in achieving all the objectives indicated by BPM is the Service-Oriented Architecture (SOA) [54, 114, 117].

With the term SOA we refer to a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units constitute a larger piece of business logic; individually these can be distributed among different nodes. An example of such composition is presented in Fig. 2.8.

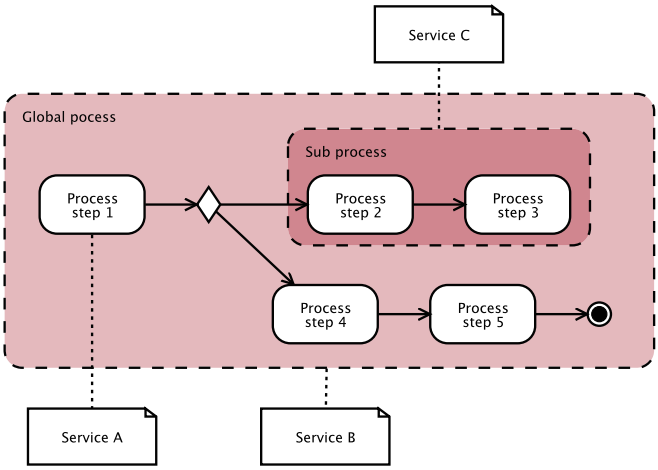


Fig. 2.8 Example of process handled by more than one service. Each service encapsulates a different amount of logic. This figure is inspired by Fig. 3.1 in [54].

In [140], a clear definition of "Business Service" is presented:

> A discrete unit of business activity, with significance to the business, initiated in response to a business event, that can't be broken down into smaller units and still be meaningful (atomic, indivisible, or elementary).

This term indicates the so-called "internal requirements" of an Information System, in opposition to the "external" ones, identified as Use Cases: *a single case in which a specific actor will use a system to obtain a particular business service from one system*. In authors' opinion, this separation simplifies the identification of the requirements and can be considered a methodological approach to the identification of the components of the system.

In the context of SOA, one of the most promising technologies is represented by Web services. In this case, a Web service is going to represent a complex process that can span even more organizations. With the Web services composition, complex systems can be built according to the given process design; however, this is still a young discipline and industries should be more involved in the standardization process.

http://www.springer.com/978-3-319-17481-5

Process Mining Techniques in Business Environments
Theoretical Aspects, Algorithms, Techniques and Open Challenges in Process Mining
Burattin, A.
2015, XII, 220 p. 101 illus., Softcover
ISBN: 978-3-319-17481-5