Software and Systems Modeling manuscript No.
(will be inserted by the editor)

Business Process Management as the "Killer App" for Petri Nets

W.M.P. van der Aalst

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands;
Business Process Management Discipline, Queensland University of Technology, Brisbane, Australia; and
International Laboratory of Process-Aware Information Systems, National Research University Higher School of Economics, Moscow, Russia.
e-mail: `w.m.p.v.d.aalst@tue.nl`

Received: date / Revised version: date

Abstract Since their inception in 1962, Petri nets have been used in a wide variety of application domains. Although Petri nets are graphical and easy to understand, they have formal semantics and allow for analysis techniques ranging from model checking and structural analysis to process mining and performance analysis. Over time Petri nets emerged as a solid foundation for Business Process Management (BPM) research. The BPM discipline develops methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes. Mainstream business process modeling notations and workflow management systems are using token-based semantics borrowed from Petri nets. Moreover, state-of-the-art BPM analysis techniques are using Petri nets as an internal representation. Users of BPM methods and tools are often not aware of this. This paper aims to unveil the seminal role of Petri nets in BPM.

Key words business process management – Petri nets – process modeling – process mining

1 Introduction

Concurrent to the development of Petri-net theory, there has been a remarkable shift from "data-aware" information systems to "process-aware" information systems [4,6]. To support business processes an enterprise information system needs to be aware of these processes and their organizational context. Whereas conventional information systems evolved around a centralized database system, today's systems are distributed and process-oriented. The growing importance of *Business Process Management* (BPM) illustrates these developments [2,5,9]. BPM includes methods, techniques, and tools to support the design, enactment, management, and analysis of such operational business processes. BPM can be considered as an extension of classical Workflow Management (WFM) systems and approaches.

Most of the contemporary BPM notations and systems use token-based semantics adopted from Petri nets [8]. Petri nets were proposed by Carl Adam Petri (1926-2010) in 1962. This was the first formalism able to model concurrency. Concurrency is very important for BPM as in business processes many things may happen in parallel. Thousands of cases may be handled at the same time and even within a case there may be various activities enabled or running concurrently. Therefore, BPM notations, techniques, and systems should support concurrency natively.

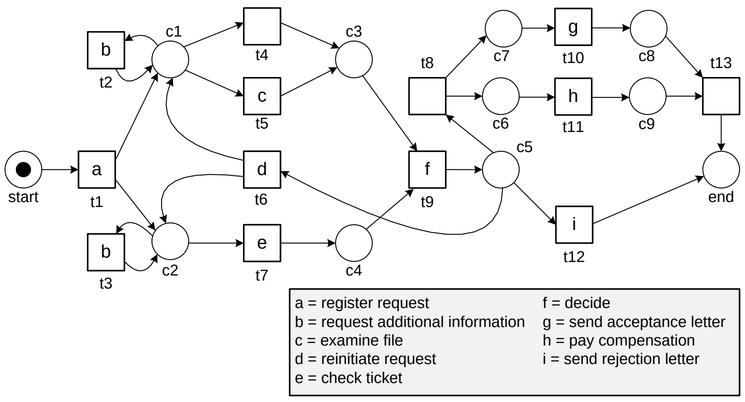

Fig. 1 A sound WorkFlow net (WF-net) modeling the life-cycle of a request for compensation. A transition may carry a label referring to some activity. Transitions without a label are "silent".

Figure 1 shows a Petri net modeling an operational process consisting of nine activities. The transitions have labels referring to these activities. For example, transition $t1$ has label a referring to the activity of registering a request for compensation. There are two transitions having a b label: both $t2$ and $t3$ represent a request for more information. Transition $t4$ has no label and corresponds to a "silent activity", i.e., an internal step not visible when it is executed. The Petri net shown in Figure 1 is a so-called *WorkFlow net* (WF-net, [3]) because there is a unique source place *start*, a unique sink place *end*, and all nodes are on a path from *start* to *end*. A WF-net models the life-cycle of *process instances*, commonly referred to as *cases*. The token in *start* refers to such a case. There can be multiple tokens referring to the same case, e.g., the tokens in places $c1$ and $c4$ after registration (a) and checking the ticket (e). However, tokens of different cases cannot get "mixed" in a WF-net. In other words, the WF-net describes the life-cycle of a case in isolation. The same assumption can be found in other notations for business process modeling (BPMN, UML activity diagrams, BPEL, EPCs, etc.). The WF-net in Figure 1 is *sound* because cases cannot get stuck before reaching the end (termination is always possible) and all parts of the process can be activated (no dead segments) [3].

Figure 1 only models the *control-flow perspective*. More sophisticated Petri nets are needed to also model the *resource* (or organization) perspective, the *data* (or information) perspective, the *interaction* (or communication) perspective, or the *time* perspective [4,6]. Elementary nets having only "black untimed dots" as tokens are not suitable for modeling these additional perspectives. Therefore, one needs to resort to *Petri nets extended with color (i.e., data), time, and hierarchy* [4,7].

As asserted in the remainder of this paper, there are at least three good reasons for using Petri nets for business process modeling, analysis, and enactment:

– *Formal semantics despite the graphical nature*: On the one hand, Petri nets are a graphical language and even simple elementary nets allow for the modeling of the basic workflow primitives [6]. On the other hand, the semantics of Petri nets (including most of the extensions) have been defined formally. Many of today's available BPM systems and notations provide ad-hoc constructs to model business processes. Especially when it comes to mixtures of concurrency and choice, semantics are not always clear and difficult to operationalize. Because of these problems it is better to use a well-established design language with formal semantics as a solid basis. Note that this does not imply that Petri nets should be used to visualize processes. Petri nets may be "hidden" by using higher-level or more colorful notations, as long as a direct mapping is possible.
– *State-based instead of event-based*: In contrast to some other process modeling techniques, the state of a case can be modeled explicitly in a Petri net. Process modeling techniques ranging from informal techniques such as dataflow diagrams to formal techniques such as process algebras are *event-based*, i.e., transitions are modeled explicitly and the states between subsequent transitions are only modeled implicitly. However, internally, processes and systems have states and these are of utmost importance for enactment and analysis. When analyzing or supporting the flow of work one should not only consider activities, but also the stages in-between activities. Typically most time passes by when cases are in-between activities and various workflow patterns (e.g., milestones and deferred choices) cannot be expressed without explicitly modeling states [6]. Therefore, states need to be "first-class citizens" for business process modeling.
– *Abundance of analysis techniques*: Petri nets are characterized by the availability of many analysis techniques. General analysis techniques ranging from model checking to simulation can be applied to Petri nets due to their concise operational semantics. Moreover, Petri-net-specific

notions such as traps, siphons, place invariants, transition invariants, coverability graphs, regions, and distributed runs [8] can be used for analysis. For example, various process mining algorithms exploit these notions when discovering a process model or when aligning modeled and observed behavior [1].

The remainder is organized as follows. First, we elaborate on the role of Petri nets in the BPM life-cycle (Section 2). Then, in Section 3, we discuss the impact of Petri nets on the BPM discipline. Finally, we philosophize about the relation between models and reality inspired by Carl Adam Petri's adagium that process models should be in accordance with the laws of physics (Section 4).

2 Playing the Token Game in Business Process Management

To explain the role of Petri nets in the BPM discipline, we start by discussing the *BPM life-cycle* [2,4] shown in Figure 2. In the *(re)design phase*, a process model is designed. This model is transformed into a running system in the *implementation/configuration phase*. If the model is already in executable form and a WFM or BPM system is already running, this phase may be very short. However, if the model is informal and needs to be hard-coded using some conventional programming language, this phase may take substantial time. After the system supports the designed processes, the *run & adjust phase* starts. In this phase, the process is enacted and adjusted when needed. In the run & adjust phase, the process is not redesigned and no new software is created; only predefined controls are used to adapt or reconfigure the process. Figure 2 shows two types of analysis: *model-based analysis* and *data-based analysis*. While the system is running, event data are collected. These data can be used to analyze running processes, e.g., discover bottlenecks, waste, and deviations. This is input for the redesign phase. During this phase process models can be used for analysis. For example, simulation is used for what-if analysis or the correctness of a new design is verified using model checking.

Petri nets may play an important, but sometimes hidden, role in all three phases shown in Figure 2. In the remainder, we detail the purpose of Petri nets when it comes to *modeling*, *analysis* and *enactment*.

2.1 Modeling

The old adagium "a picture is worth a thousand words" succinctly explains the powerful role Petri nets can play when describing or designing business processes. The simple WF-net in Figure 1 can serve as input for discussions, e.g., different process redesigns can be explored and ideas can be structured. An important feature of Petri nets is that one can play the so-called "token game", i.e., the process can be animated and different scenarios can be explored by using a simple set of rules.

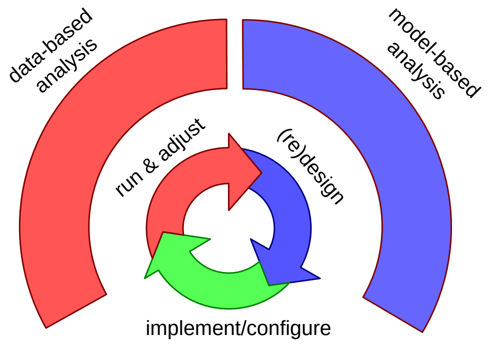

Fig. 2 The BPM life-cycle consisting of three phases: (1) *(re)design*, (2) *implement/configure* and (3) *run & adjust*.

As indicated before, Figure 1 only models the *control-flow perspective*, i.e., the ordering of activities. Often one would also like to model the *resource perspective*, i.e., the involvement of people, departments, rooms, machines, and other resources. This is sometimes referred to as the organizational perspective. For example, one may want to model relations between roles (resource classes based on functional aspects) and groups (resource classes based on organizational aspects), and clarify organizational issues such as responsibility, availability, and authorization. Resources, ranging from humans to devices, form the organizational population and are allocated to roles and groups.

The *data perspective* deals with control and production data. Control data are case attributes introduced solely for routing purposes. Production data are information objects (e.g., documents, forms, and tables) whose existence does not depend on routing only. Activities often require particular input data and produce particular output data, e.g., a person needs to fill out a form with pre-filled data. Moreover, decisions in the process may be based on such data.

The *interaction perspective* is concerned with all interrelations among different processes and cases. For example, activities for orders, orderlines, and deliveries are interrelated but cannot be straightjacketed into a single monolithic WF-net, BPMN, EPC, or UML model. Moreover, processes may need to communicate across organizational boundaries.

The *time perspective* deals with flow times, deadlines, timeouts, waiting times, service times, and response times. For example, one may model that a claim needs to be rejected if it is not processed within two weeks. One can also model that the average time needed to make a decision is 2 hours.

Whether all of these perspectives need to be modeled in detail, depends on the model's *purpose*. For example, if the model is used for simulating “what-if scenarios” it is important to model service times and the availability of resources, but it may be less relevant to model all data elements. Conversely, if the model is used for enactment there is no need to model service times (as these will emerge automatically), but it is crucial to model the input and output data of all activities.

The WF-net shown in Figure 1 is an elementary net, i.e., tokens are "black dots" that cannot be distinguished and carry no data. To adequately model all perspectives, one can use Petri nets extended with color (i.e., data), time, and hierarchy [4,7]. However, when using such extended Petri nets, one may still want to analyze the process based on abstractions corresponding to elementary nets.

2.2 Analysis

When using informal models, it is impossible to use them for process analysis. Fortunately, for Petri nets a broad range of analysis techniques is available. Figure 3 classifies these techniques using two dimensions. First of all, one can analyze a process using just a hand-made model or one can use actual event data (referred to as data-based analysis in Figure 2). Secondly, one can focus on the functional properties or also incorporate non-functional properties.

	model-based analysis	analysis based on data and model
functional properties (e.g. deadlocks)	*verification, model checking, soundness checking, etc.*	*process mining (e.g., process discovery and conformance checking)*
non-functional properties (e.g. performance)	*simulation, Markovian analysis, optimization, etc.*	*process mining (e.g., model extension and prediction)*

Fig. 3 A basic characterization of process-based analysis techniques.

Traditionally, the bulk of Petri net research focused on *model-based analysis*. Moreover, the largest proportion of model-based analysis techniques is limited to *functional properties*. Generic techniques such as model checking can be used to check whether a Petri net has particular properties, e.g., free of deadlocks. Petri-net-specific notions such as traps, siphons, place invariants, transition invariants, and coverability graphs are often used to verify desired functional properties, e.g., liveness or safety properties [8]. Consider for example the notion of *soundness* defined for WF-nets [3]. A WF-net is sound if and only if the following three requirements are satisfied: (1) *option to complete*: for each case it is always still possible to reach the state which just marks place *end*, (2) *proper completion*: if place *end* is marked all other places are empty (for a given case), and (3) *no dead transitions*: it should be possible to execute an arbitrary activity by following the appropriate route through the WF-net. The WF-net in Figure 1 is sound and as a result cases cannot get stuck before reaching the end (termination is always possible) and all parts of the process can be activated (no dead segments). Obviously, soundness is important in the context of BPM. Fortunately, there exist nice theorems connecting soundness to classical Petri-net properties. For example, a WF-net is sound if and only if the corresponding short-circuited Petri net is live and bounded. Hence, proven techniques and tools can be used to verify soundness. Soundness is just one of many properties one may want to investigate. Questions like "Can the same resource execute activities c and f for a request involving a transatlantic flight?" can only be checked using more sophisticated techniques.

Model-based analysis may also focus on *non-functional properties* such as flow times, response times, costs, risks, utilization, and availability. Such properties are of the utmost importance for BPM. For particular classes of Petri nets one can use Markovian analysis, e.g., stochastic Petri nets with negative exponential delay distributions can be translated into Markov chains that can be analyzed to determine flow times, likelihoods, etc. For more sophisticated process models and questions, one often needs to resort to simulation. Therefore, there are many BPM tools that allow for some form of simulation [4].

In recent years, more and more researchers started to investigate the right-hand side of Figure 3. The interest in *data-based analysis* is fueled by the increasing availability of event data and the interest of organizations in fact-based analysis (evidence-based BPM). The term *process mining* is used to refer to techniques that extract knowledge from event logs [1]. Process mining techniques form a family of *a-posteriori* analysis techniques exploiting the information recorded in audit trails, transaction logs, databases, etc. Process mining includes (automated) process discovery (i.e., extracting process models from an event log as shown in Figure 4(c-d)), conformance checking (i.e., monitoring deviations by comparing model and log), social network/organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations.

The growing importance of process mining for anyone interested in process analysis can be illustrated as follows. Consider a typical 1 TB hard disk purchased in 2010. The disk can store 10^{12} bytes (i.e., one Terabyte). According to IDC, the entire "Digital Universe" was 1.2 Zettabyte (1.2×10^{21} bytes) at that time.[1] Hence, the 1 TB disk needs to grow $2^{30.16} = \frac{1.2 \times 10^{21}}{10^{12}}$ times. Based on the average growth rate of hard disks over the last decades and an extrapolation of Moore's law, we assume that hard disks indeed double every 1.56 years. This implies that in $30.16 \times 1.56 = 47.05$ years a standard hard disk may contain the whole "Digital Universe" of 2010. This

[1] Estimate taken from IDC's annual report, "The Digital Universe Decade: Are You Ready?", May 2010.

includes the entire internet, all computer files, transaction logs, movies, photos, music, books, databases, etc. This simple calculation exemplifies the incredible growth of event data in the next decennia. Business processes will generate more and more event data that can be used for analysis. Detailed transaction data and sensor data (cf. RFID tags) will enable new process mining applications replacing traditional analysis based on handmade models [1].

2.3 Enactment

Petri nets have executable semantics, i.e., they can be used to generate behavior. The core of any WFM/BPM system is the so-called “workflow engine”. Such an engine is basically playing the “token game” while interacting with its environment. Most engines use token-based semantics. After completing an activity for a particular case, the corresponding state (marking in Petri net terms) is updated and the newly enabled activities are offered to the environment.

3 Influence of Petri Nets on Languages and Systems

There seems to be a never ending stream of new process modeling notations. Some of these notations are foundational and have been around for decades (e.g., Petri nets). Other notations are vendor specific, incremental, or are only popular for a short while. It seems that ongoing discussions on the various competing notations often conceal more foundational issues.

Notations range from languages aiming to provide a formal basis (e.g., finite state machines, Petri nets, and process algebras) to vendor specific notations (e.g., the different workflow languages used by BPM vendors). Industry standards such as BPEL (Business Process Execution Language) and BPMN (Business Process Modeling Notation) are typically only partially adopted; vendors support just subsets of these standards and users tend to use only a tiny fraction of the concepts offered [6]. Obviously, there is little consensus on the modeling language to be used. This resulted in the “Tower of Babel of process languages”: a plethora of similar but subtly different languages inhibiting effective and unified process support and analysis.

Despite the “BPM Tower of Babel”, Petri nets played an important role in de development of the field. Almost all business process modeling languages and BPM/WFM systems use token-based semantics inspired by the Petri-net “token game”. Although Petri nets are often hidden, there are also examples of BPM/WFM systems and tools showing Petri nets directly to the user. *COSA*, one of the leading WFM tools in the 90-ties, is completely based on Petri nets: the COSA modeler, COSA engine, and COSA simulator all use Petri nets. *Baan*, the main ERP (Enterprise Resource Planning)

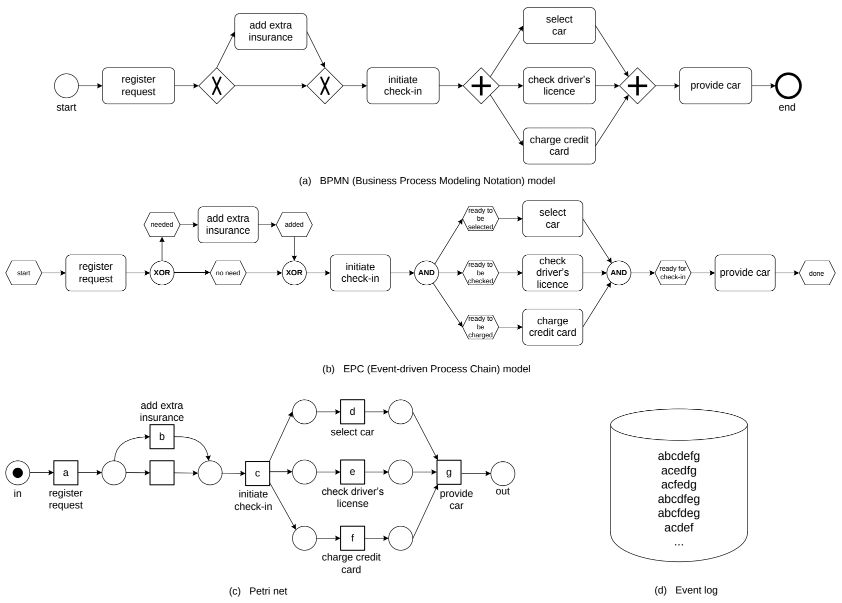

Fig. 4 Three types of models describing the same process: (a) BPMN, (b) EPC, and (c) Petri net. The event log (d) shows possible traces of this model using the short activity names provided by the Petri net.

competitor of SAP in the mid 90-ties, was famous for its Dynamic Enterprise Modeler (DEM). Baan’s DEM used Petri nets to model processes and was used to align and implement the Baan ERP system in the organizational architecture of the end-user company. COSA and DEM influenced many later BPM/WFM/ERP systems. See for example today’s Business Operations Platform (BOP) of Cordys and Oracle’s BPM suite.

Another remarkable example is the *Protos* modeler developed by Pallas Athena. This modeler uses Petri nets as a modeling notation. In 2010 more than 250 out of 441 Dutch municipalities were actively using Protos as a modeling tool. Protos also supports simulation and is internally using the ExSpecT simulation tool. ExSpecT was originally developed at Eindhoven University of Technology (TU/e) as a prototyping and simulation tool. Today, Protos is part of Perceptive’s *BPMone* platform, a BPM suite to discover, design, execute and improve business processes.

Of course there are also many open-source/academic BPM systems and tools prominently using Petri nets. Some examples are: *YAWL* (WFM system), *WoPeD* and *Yasper* (business process modeling and simulation), and *ProM* (process mining). Moreover, when going back in history, one can find many examples of Petri-net based tools for process automation, e.g., *Officetalk* at Xerox PARC in the late 70-ties, *SCOOP* (System for Computerizing of Office Processes) developed by Michael Zisman (late 1970s), *Income Workflow* by Promatis in the 90-ties, etc.

In spite of the many examples of interesting BPM systems and tools exposing their users to Petri nets, the actual impact of Petri nets on BPM is concealed behind the colorful notions typically used in industry. Figure 4 shows the same process using three different notations. The *Business Process Modeling Notation* (BPMN) uses activities, events, and gateways to model the control-flow. In Figure 4(a) two types of gateways are used: exclusive gateways are used to model XOR-splits and joins and parallel gateways are used to model AND-splits and joins. BPMN also supports other types of gateways corresponding to inclusive OR-splits and joins, deferred choices, etc. [5,6,9]. *Event-driven Process Chains* (EPCs) use functions, events, and connectors to model the control-flow (cf. Figure 4(b)). Connectors in EPCs are similar to gateways in BPMN. There are OR, XOR, and AND connectors. Events in EPCs are similar to places in Petri nets, e.g., just like places and transitions, events and functions need to alternate along any path in an EPC. However, events cannot have multiple successor nodes, thus making it impossible to model deferred choices [6]. *UML Activity Diagrams* (UML ADs) – not shown in Figure 4 – are similar to BPMN when it comes to the basic control-flow constructs.

BPMN, EPCs, UML ADs, and many other business process modeling notations have in common that they all use token-based semantics. Therefore, there are many techniques and tools to convert Petri nets to BPMN, BPEL, EPCs and UML ADs, and vice versa. As a result, the core concepts of Petri nets are often used indirectly, e.g., to enable analysis, to enact models, and to clarify semantics.

4 The True Fabric of Business Processes

Two maxims put forward by Carl Adam Petri are "Concurrency should be incorporated as a starting point rather than an afterthought (locality of actions)" and "A modeling technique should obey the laws of physics". Petri nets were the first model to adequately capture concurrency. Of course concurrency plays an important role in business processes, e.g., there may be many resources (people, machines, etc.) working concurrently and at any point in time there may be many running process instances. Petri was interested in the relationship between process modeling and physics (e.g., the finite and invariant velocity of light and Heisenberg's uncertainty principle).

In the context of BPM one should also pay attention to the relation between process models and the actual characteristics of business processes. Business processes tend to be highly concurrent and non-monolithic. Therefore, sequential models are inadequate [4]. Moreover, one cannot restrict attention to a single process instance in isolation (as is the case in BPMN, EPCs, etc.). For example, there may be complex many-to-many relationships between orders, order lines, and deliveries. One order may consist of multiple order lines, there may be multiple deliveries related to the same order, and a delivery may refer to order lines of different orders. Traditional modeling approaches have problems dealing with such complex dependencies whereas practical experiences with process mining show that interactions between artifacts are essential for process analysis [1].

The empirical nature of process mining helps managers, consultants, and process analysts to better understand the "fabric of real business processes" and, thus, also see the limitations of conventional process modeling languages [1]. The challenge is to link elegant succinct formal models like Petri nets to behavior actually observed in reality.

References

1. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer-Verlag, Berlin, 2011.
2. W.M.P. van der Aalst. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, 2012 (in print).
3. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
4. W.M.P. van der Aalst and C. Stahl. *Modeling Business Processes: A Petri Net Oriented Approach.* MIT press, Cambridge, MA, 2011.
5. M. Dumas, M. La Rosa, J. Mendling, and H. Reijers. *Fundamentals of Business Process Management.* Springer-Verlag, Berlin, 2013.
6. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment.* Springer-Verlag, Berlin, 2010.
7. K. Jensen and L.M. Kristensen. *Coloured Petri Nets.* Springer-Verlag, Berlin, 2009.
8. W. Reisig. *Petri Nets: Modeling Techniques, Analysis, Methods, Case Studies.* Springer-Verlag, Berlin, 2013.
9. M. Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer-Verlag, Berlin, 2007.