Event-driven business process modeling and a quick guide to application modernization

Pavel Hruby, Christian Vibe Scheller

REA Technology, http://reatechnology.com

Abstract

This paper presents an event-driven business process model (EDBPM), which is easy to understand by non-technical experts and at the same time is precise enough to be used as a specification for software design. In contrast to traditional process modeling approaches that emphasize the sequential flow of activities, event-driven process modeling focuses on events and responses to the events. The paper describes how the event-driven model is constructed, and then, at a high level, illustrates how it can be transformed into a software design.

Keywords

Event-driven, business process modeling, problem domain, problem space

1. Introduction

Business processes are typically described by various diagramming techniques, such as BPMN [5], VDML [6], and flowcharts [9]. An apparent problem with these descriptions is that the non-technical experts must understand the notation and its precisely defined semantics. When presenting a business process to people who do not fully understand what arrows, boxes, and other symbols on a diagram exactly mean, they will get a false impression that the process correctly describes what they think, even if it does not. They easily miss subtle details, cannot assess all the consequences to which the process leads, and their feedback to the process modeler is limited because they do not know how to express their thoughts on a diagram.

A solution to this problem could be an event-driven business process model: identify all external events relevant to the organization, and describe in plain English how various applications and system components respond to these events. The events are grouped to form business processes, and usually, there are several business processes for each economic resource controlled by the organization.

This paper is structured as follows: The first three sections deal with the problem domain (also called problem space in domain-driven design [11]), and outline how to create an as-is and to-be event-driven business process model. Sections 5 and 6 deal with the solution domain – how to create an information architecture and integrations between applications so that the software application fulfills the requirements of the business process model.

Proceedings of the 17th International Workshop on Value Modelling and Business Ontologies (VMBO 2024), February 26–27, 2024, 'sHertogenbosch, The Netherlands

phruby@acm.org (P. Hruby); cvs@reatechnology.com (C. V. Scheller)

0000-0002-8278-283X (P. Hruby)

2. Motivation

Figure 1 illustrates a simple onboarding process for newly hired employees. After candidates sign the contract, their data is collected and onboarding workflow is started, which encompasses creating a user account and email address, assigning the software licenses, and other tasks using the ITSM tool.

This BPMN diagram was created at the workshop with non-technical subject matter experts, after which all participants agreed that this diagram represents the current onboarding process. However, after several days, when this diagram was discussed again, many workshop participants came up with their version of the onboarding process, and they all believed that their specific version was illustrated in this diagram.

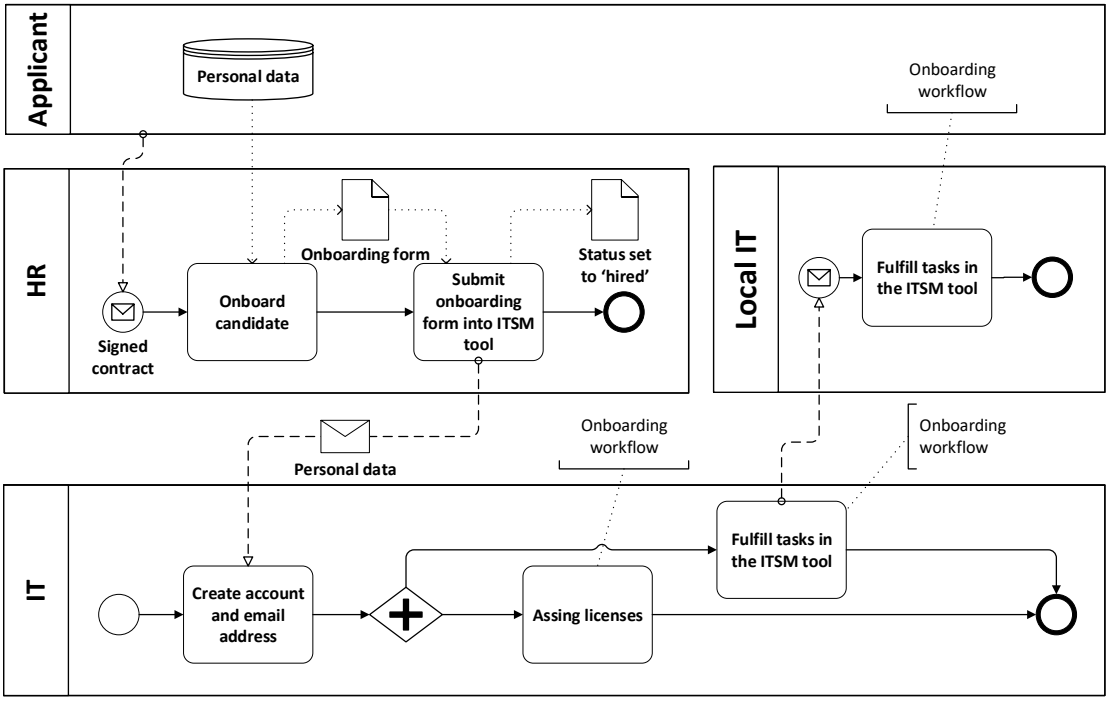

Figure 1: People say they understand this drawing, but they often don't.

In BMPN it is too easy to create a syntactically correct model, although it is semantically incorrect. A similar problem exists with other methods, too. Qingyu Wang evaluated five business process modeling methods EPC, IDEF0, IDEF3, UML activity diagrams, and EDPM, and concluded that "most of them are either ontologically incomplete or unclear" [10].

If this diagram was presented to the developers, it would lead to an IT solution that does not meet the requirements and expectations of the subject matter experts, because the subject matter experts and developers interpret it differently. The strange thing is that everyone thinks they understand it perfectly.

A solution to this problem could be as follows. Instead of creating diagrams focusing on sequences of activities, the focus is on understanding how organizational units and software applications respond to the events that occur outside of the organization. The responses are

described in plain English, eliminating the problem of interpreting the diagramming notation. The English description also includes the exceptions and compensating activities. We will show that this description is understandable to the subject matter experts and at the same time precise enough for software architects to design the system.

3. Event-Driven Business Process Model

Creating the event-driven business process encompasses several steps briefly described below. By event, we mean an external event that occurs in the real world. This is the main difference between the event-driven business process model and Domain-Driven Design [11] and event-driven software architectures, which focus on the events generated by applications because their main goal is not business process description but software design and implementation.

The modeling process starts by identifying all economic resources that users of an enterprise would like to plan, monitor, and control, such as labor, hardware, software, material, services, money, etc. A typical modernization project has only some economic resources in its scope.

For each economic resource, we identify all events that occur in the real world and affect the resource, i.e. change its state. For each event, we describe how various software applications and organizational units respond to this event, see Table 1.

Table 1 shows two processes of economic resource labor; the joiner and leaver process. Note, that this is a simplified example to illustrate the idea; the real-world complexity is discussed later in this paper.

Table 1
Event-driven business process model for the labor economic resource

Process	Event	HR System	User account management	ITSM tool	Local IT support
Joiner Process	Position opened	**register**	ignore	ignore	ignore
	Contract signed	**register**	-create a UserID -create an email address	-create user -run onboarding workflow	reserve equipment
	One week before start date	ignore	**activate account**	ignore	ignore
	Start date	ignore	ignore	ignore	**issue equipment**
Leaver process	Letter of termination received	**register**	ignore	run leaver workflow	expect equipment return
	End date	**revoke access rights**	deactivate account	deactivate account	receive equipment

The core part of Table 1 is the second column labeled Event, containing all events changing the state of the economic resource labor, thus representing the labor lifecycle. The Joiner and Leaver processes in the first column are convenient groups of events. Processes are useful for communication, but conceptually they are secondary modeling artifacts compared to the events.

The remaining columns represent the applications and organizational units that may respond to the events. The *HR (Human Resource)* system, such as Workday and SAP Success Factors, stores and manages employee data and supports the administrative activities of human resource personnel. The *User Account Management* application is, for example, a bundle of Microsoft Active Directory and Microsoft Exchange; these applications are tightly coupled and in Table 1 are considered a single system. The *ITSM tool* (*IT Service Management tool*) is, for example, ServiceNow, often used as an automation platform. The *Local IT support* is a physical service desk, where technicians help employees with various IT issues.

The first event is a management decision of a *Position opened*, registered by the HR system, and does not affect other applications. The *Contract signed* event is registered in the HR system, and consequently, the User Account Management app creates a user ID and email address; the user is also created in the ITSM tool, which runs the onboarding workflow, where the hiring manager decides about the IT equipment for the new employee. Consequently, the Local IT reserves the equipment. *One week before the start date* the User Account Management app activates the already created user account. At the *Start date* event, the Local IT issues the equipment to the new employee.

There is typically only one application that registers each external event and then notifies other applications. This response is indicated in **bold** font in Table 1. Note that allowing several applications to register the same external event could lead to race conditions during system execution, as discussed in [3], page 11.

Responses to some events can be complex. For example, the *Run Onboarding Workflow* of the ITSM tool may be an algorithm encompassing interactions and information exchange with other applications and organizational units. This complexity is out of the scope of the event-driven business process model and is handled in the following steps described later.

The result is a precise and accurate business process model, yet described in plain English, so it is easy to understand by non-technical subject matter experts. The event-driven business process model determines the software architecture supporting the described business processes, as shown in sections 5 and 6.

3.1. Is the model complete?

The main goal of the event-driven business process model is wholeness [1]. The model should describe the complete lifecycle of each economic resource, from its creation to consumption. How to verify that the model contains all events affecting the economic resource?

One way is to examine the data stored in existing applications dealing with the given economic resource. For example, the Person table in SAP Success Factors has 52 attributes, though a typical HR organization uses only about half of them. For each attribute that HR uses, there must exist at least one external event that sets this attribute, an event that reads this attribute, and optionally an event that updates this attribute.

Another way is to create an economic model of the enterprise using, for example, the REA [8] or POA [4] ontologies, and verify that all events specified by these ontologies are part of the model. For example, for each process, there should be one or more events representing a plan

or intention, corresponding to the commitment in the REA ontology, and one or more events representing the execution of the plan, corresponding to the economic event in the REA ontology. Likewise, for each economic resource, there should be a process (a collection of events) describing how the economic resource gets under the control of the enterprise and a process describing how it leaves the enterprise's control. For example, the economic resource labor has the Joiner and Leaver processes, and for the economic resource material, there can be the Purchase and Disposal processes or the Production and Consumption processes.

A precise representation of the lifecycle is a state diagram, which can be created for each economic resource. The state diagram will identify not only the events affecting the resource but also the actions performed upon state change.

4. Describe the future state and roadmap

In this step, the subject matter experts identify the pain points and responses to real-world events that need attention, and decide, which of them should be fully or partially automated, or can run manually because they are very rare.

This is also an opportunity to rethink the business processes from scratch, such as whether a response to a real-world event can be moved from one to another system.

Table 2 illustrates the future state event-driven process model. Two events were added: *No-show on the start date*, and *Leave confirmed*. Notifications were added to some responses. By notification here we mean that some info should be propagated from one application to another. This can in practice be implemented as application-generated events; this is an implementation decision belonging to the solution domain. The most important decision in this step is which processes will be automated, indicated by the * symbol in Table 2. For example, on the *Start date* event, HR responds manually by the action *change status to active*, and this status change of the worker is automatically propagated to other systems, therefore, the action *change status to active* is automatic in User Account Management and ITSM tool.

An example of decisions taken in this step is whether the run onboarding workflow should be performed by the ITSM tool or the HR system. Likewise, can the creation of a User ID and email address be automated by the User Account Management system from the data already existing in the HR system? We also need to determine the exceptions and issue handling at the business process level. What happens if the onboarding workflow gets stalled and does not finish at a date before the start date, for example, should the hiring manager be notified? The unhappy scenarios should be described in the table with happy path responses.

Table 2
Event-driven business process model - future state. The * symbols indicate automated responses to the events.

Process	Event	HR System	User account management	ITSM tool	Local IT support
Joiner Process	Position opened	**register**	ignore	ignore	ignore
	Contract signed	**register**	*create a UserID* *create an email address*	*create user* *run onboarding workflow* *on failure:*	reserve equipment

				create incident *notify initiator* *on success:* *notify hiring manager*	
	One week before start date	ignore	***activate account***	*check workflow completion* *notify HR*	ignore
	No show on the start date	*run no-show workflow*	*deactivate account*	*run leaver workflow *	ignore
	Start date	change status to 'active'	*change status to 'active'*	*change status to 'active'*	**issue equipment**
Leaver process	Letter of termination received	**register**	ignore	ignore	ignore
	Leave confirmed	**register**	ignore	*start leaver request*	expect equipment return
	End date	run leave controls	*deactivate account*	***notify supervisor*** ***notify HR***	receive equipment

The result is an updated model created in the previous step, determining the future state and roadmap, aligned with user needs and the strategic vision.

4.1. Business rules

Specifying business rules is currently not part of the event-driven business process model. However, specifying the decisions and logic triggering notifications is an important part of the problem domain. The OMG standard Decision Model and Notation (DMN) [7], could be a useful complement to the process model, for specifying the business rules.

5. Determine the information architecture and application algorithms

As noted in section 3, there is typically a single application that registers an external event and notifies other applications and organizational units about that event, i.e. the information about the real-world events must be propagated through the application ecosystem. Table 1 and Table 2 purposely do not specify how this info is shared, because it is part of the solution domain.

The decision on how to propagate the information about the event through the system belongs to the solution domain. These considerations are illustrated in Figure 2 and Figure 3. Note that there are many other possibilities to design the information architecture, for example, using an event broker that encapsulates the communication mechanism between the applications.

In Figure 2 the HR System registers the *Contract Signed* event and propagates this information to the User account Management, ITSM Tool, and the Local Support. In other

words, the HR System is the orchestrator in this interaction. This design will require integrations between the HR System and User Account Management, and between the HR System and the ITSM tool, and the HR system will need to implement the notification mechanism to Local IT Support.

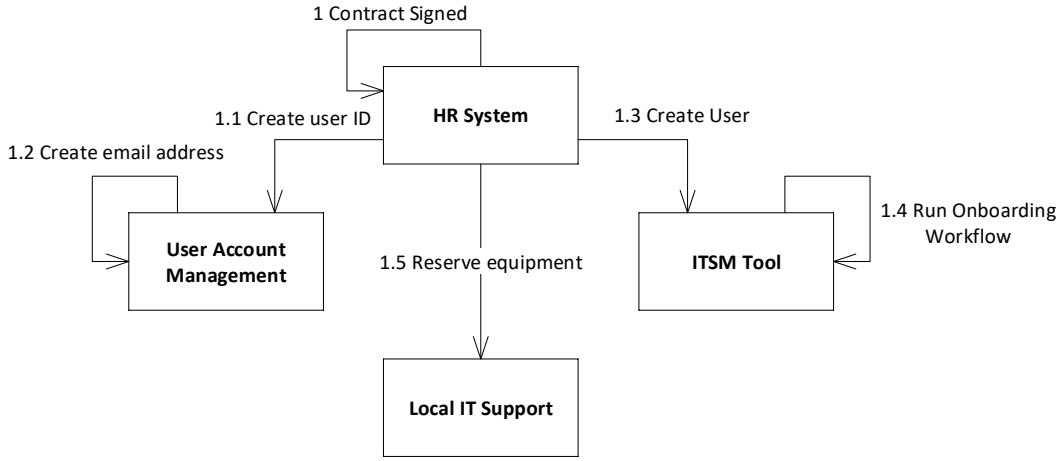

Figure 2: Response to the Contract Signed event – HR System orchestrates the flow

Figure 3 shows an alternative solution, where the HR System registers the *Contract Signed* event and propagates it to the ITSM Tool. The ITSM Tool propagates it to User Account Management and Local IT Support. In other words, the ITSM Tool is the orchestrator in this interaction. This design does not require integration between the HR System and User Account Management, though it requires integrations between the ITSM Tool and User Account Management, and between the HR System and the ITSM Tool. The ITSM Tool must implement the notification mechanism to Local IT Support.

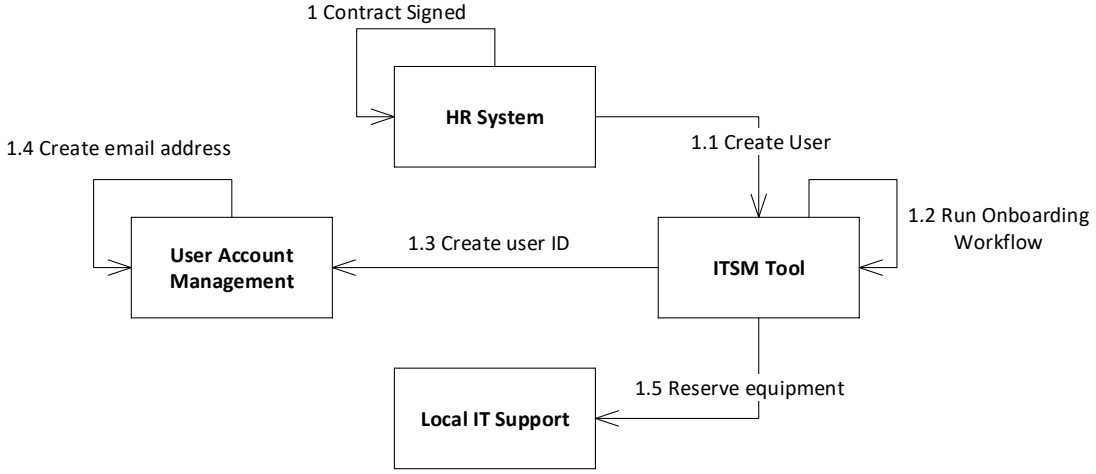

Figure 3: Response to the Contract Signed event – ITSM Tool orchestrates the flow

Which solution to choose? As with any other architectural decision, information architecture is a series of trade-offs.

One aspect to consider is Conway's law. Conway's law observes that "Organizations, which design systems are constrained to produce designs which are copies of the communication structures of these organizations" [2]. It appears that Conway's law also works in the reverse order. If the organization has a dominant department, then this department's preferred application is a candidate for orchestrating the information propagation, not because of technical reasons, but rather due to organizational dynamics. For example, if a service management department is more dominant than the HR department, then the ITSM tool would be a better choice to orchestrate the flows rather than the HR system.

At this step, we also design the application algorithms (fully or partially automated), such as running the onboarding workflow. It requires a decision of the hiring manager to select appropriate IT equipment for the employee, such as laptop model, application licenses, access to production systems which will trigger additional workflow tasks, etc. Part of the design is error handling, such as what to do if some tasks are not completed on time.

6. Implement the integrations and application algorithms

This step requires decisions about the actual data transfer mechanisms between applications, such as REST, gRPC, file transfer using batch jobs, etc. Existing applications usually set constraints on what transfer mechanisms are available.

To respond to the events and to run the application algorithms, each application and organizational unit typically needs additional data. This determines the additional data flows between the applications. For optimizing these data connections in a digital solution, we can use several patterns for distributed computing, such as orchestration, usually suitable for smaller systems, event broker typically used with microservices, or data mesh determining data ownership.

Figure 4 illustrates two variants of the joiner process, corresponding to the information flows in Figure 2 and Figure 3. At this phase, all data must be specified, e.g. the content of employee data flow is specified including the data types at both providing and receiving ends, and appropriate conversions whenever required. As the HR system has to store the account info generated by the User Account Management system, such as user ID and email address, the required integrations have been added.

As with any distributed computing system, the data are eventually consistent. That is, different applications may, for some time, contain different versions of the data. It cannot be completely avoided due to the CAP theorem, but fortunately, there are several ways to deal with this problem. For example, we can design the information architecture in section 5 such there is only a single way to propagate the truth; there are also other solutions, so the eventual consistency does not become a business issue. We should also consider possible race conditions, commutativity, idempotence, conflicting updates, single-sign-on, consolidated logging, etc. outlined in [3].

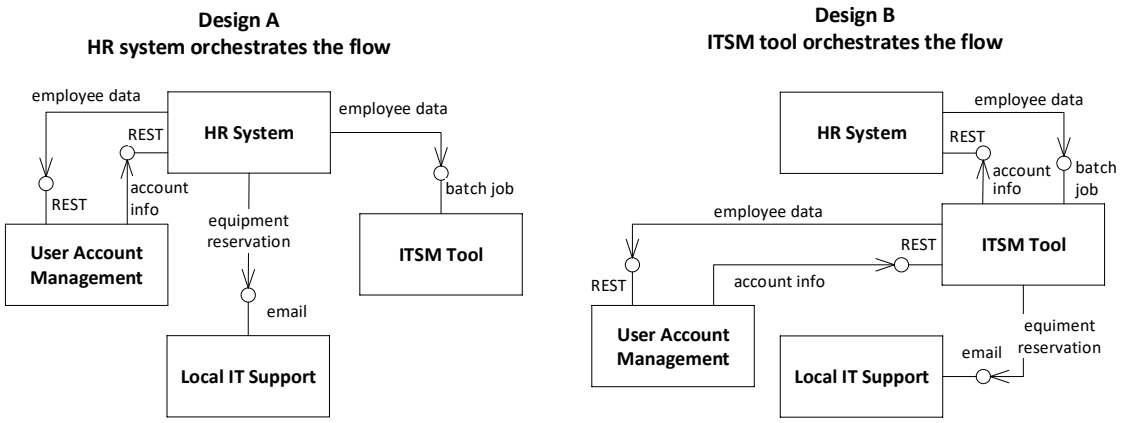

Figure 4: Two variants of the joiner process integrations

7. Future work

We first of all welcome feedback, experience, and suggestions on future improvements to the method. Based on our current experience with this method, the following areas should be explored in more detail:

- Business rules. Business rules are not specified in the event table, though they belong to the problem domain. A possible approach would be using the OMG standard for Decision Model and Notation, DMM [7]. DMM provides a mechanism to model decision-making associated with tasks and is designed to work together with business modeling methods such as BPMN.
- Event ordering. While well-designed software applications must accept events occurring in any order, in business process modeling there is often a natural ordering of the events or assumptions that certain events can occur only if other events have already occurred. For example, the event First Day at Work can occur only if the event Contract Signed has already occurred. These logical dependencies between events are currently not depicted in the model. We would like to further investigate a suitable way to represent the event ordering. A possibility to specify event ordering precisely is creating a state diagram for each economic resource.
- Ontological completeness. As the event-driven business process modeling has simple, yet clear semantics, mapping its semantics to a higher-level ontology can evaluate the expressiveness and ontological clarity of this method.

8. Discussion and conclusion

The event-driven business process model is a novel approach that provides the following benefits:

- Completeness. The event-driven business process model is a top-down approach, that encompasses all economic resources that the organization wants to plan, monitor, and

control, ensuring nothing important has been forgotten. It could otherwise easily happen when more complex information models are created only using the expertise and intuition of the modeler.

- Understandability and clear communication. Although it is a precise model, it is expressed in plain English and is easily understandable to non-technical subject matter experts. Anecdotal evidence, also discussed in the motivation section, is when a process has been presented as a BPMN diagram to the HR personnel, their response was "looks good". However, presenting the same process to the same audience as an event-driven business process model started a discussion and constructive feedback – some application responses were not accurately described, some responses needed additional clarification and some pain points of the current implementation were identified.
- Consistency. The event-driven business process model makes it visible how each event is affecting various applications of the enterprise.
- Business process automation. The event-driven business process model enables the subject matter experts to make a qualified decision on whether a response to an event should be automated.
- Clear distinction between the problem and solution domain. Keeping this distinction is extremely important in software design, in communicating with vendors of software solutions, etc. The event-driven business process model specifies the actions of applications and organizational units but does not specify how the information is shared between them. This is part of the solution domain, i.e. software architecture and design.

We can conclude by emphasizing that an event-driven business process model is a declarative representation of a business process. Traditional process modeling approaches focus on sequential flows of activities. This is useful for explaining the happy path, but if all variants and exceptions should also be included, the description becomes overly complex. The event-driven business process model describes a process invariant, often much simpler, which makes it easier for software architects to design solutions implementing the process.

Acknowledgements

Thanks to the participants of the Value Modeling and Business Ontologies (VMBO) 2024 workshop and the anonymous reviewers for their valuable comments and suggestions. Also, thanks to insights of Fritz Henglein and Boris Düdder of the University of Copenhagen about Conway's law applicable in the reverse order, i.e. an IT system, that does not follow the organizational communication structure, will encounter resistance. Last but not least, we thank Graham Gal of Isenberg School of Management, University of Massachusetts Amherst, who during discussions at VMBO 2023 recommended using the application data model to discover additional events to make the event-driven business model complete.

References

[1] Christopher Alexander The Nature of Order: An Essay on the Art of Building and the Nature of the Universe, Center for Environmental Structure, 2002.

[2] Melvin Conway, ”How Do Committees Invent?” Datamation: 14 (5): 28–31, 1968. http://www.melconway.com/Home/pdf/committees.pdf
[3] Pavel Hruby, Christian Vibe Scheller, ”Microservice Architecture Patterns for Enterprise Applications Supporting Business Agility”, PLoP '22: Proceedings of the 29th Conference on Pattern Languages of Programs, October 2022, Article No.: 21, Pages 1–21, URL: https://dl.acm.org/doi/abs/10.5555/3631672.3631698
[4] Christian Vibe Scheller, Pavel Hruby, ”Business Process and Value Delivery Modeling Using Possession, Ownership, and Availability (POA) in Enterprises and Business Networks”. Journal of Information Systems, summer 2016, Vol. 30, No. 2, pp. 5-47. URL: https://www.researchgate.net/publication/287454859_Business_Process_and_Value_Delivery_Modeling_Using_Possession_Ownership_and_Availability_POA_in_Enterprises_and_Business_Networks
[5] OMG, Business Process Model and Notation (BPMN), 2011, URL: https://www.bpmn.org/
[6] OMG, Value Delivery Metamodel (VDML), 2018, URL: https://www.omg.org/spec/VDML/
[7] OMG, Decision Model and Notation (DMN), 2023, URL: https://www.omg.org/dmn/
[8] Guido Geerts, Wiliam E. McCarthy, The Ontological Foundations of REA Enterprise Information Systems. Annual Meeting of the American Accounting Association, Philadelphia, PA, 2000, URL: https://msu.edu/user/mccarth4/Alabama.doc
[9] ISO 5807, Information processing, Documentation symbols and conventions for data, program and system flowcharts, program network charts, and system resources charts, 1985, URL: https://www.iso.org/standard/11955.html
[10] Qingyu Wang: A Proposal for a Process Modeling Methodology, MSc Thesis, The University of British Columbia, Faculty of Commerce and Business Administration, 2002 URL: https://open.library.ubc.ca/soa/cIRcle/collections/ubctheses/831/items/1.0090921
[11] Eric Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Pearson International, 2003