

# Coping with natural parallelism in business process models

Václav Řepa

Prague University of Economics and Business, W. Churchill sq. 4, Prague 3, Czech Republic

## Abstract

Business system can be understood as a system of mutually cooperating parallel business processes. The parallelism of business processes is natural, meaning it comes from the essence of the Real World. Therefore, it has to be respected in the business system modeling methodologies. There are various kinds of expressing parallelism in the business process models that differ in quality with respect to the basic principles of algorithms, general logic, and the principles of process-driven management. In the paper, we show the way how to express the natural parallelism of business processes with full respect to the above mentioned principles using the concept of the "process state". We explain this concept in the context of the MMABP methodology and show its essential relation to the general principles of various relevant fields. In the simple real example, we also show the practical consequences of this way of modeling business processes for the conceptual design of a process-driven organization.

## Keywords

Business process model · Algorithm · MMABP · Cooperation of processes · Jackson System Development

## 1. Introduction

The business system can be understood as a system of mutually cooperating parallel business processes. The parallelism of business processes is natural, meaning it comes from the essence of the Real World. Therefore, it has to be respected in the business system modeling methodologies. There are various kinds of expressing parallelism in the business process models that differ in quality with respect to the basic principles of algorithms, general logic, and the principles of process-driven management.

Proper working with business processes especially in the conceptual stage requires a perfect understanding of their nature. The typical mistake is underestimating of some aspects of its multidimensional nature. IT people often understand business processes as just a technical issue and consequently, overlook an essential reason for this way of managing an enterprise. On the other hand, managers often neglect the technical consequences of managing business processes and regard them as just a task for technicians that has nothing to do with managerial decisions. But a cruel truth is that no one of these approaches can result in the final success since there is no proper managerial decision without technical aspects, and there is no good solution to the technical problem in the business process without understanding its managerial consequences. Apparently, in the process of modeling the business processes both managerial and technical dimensions must be respected at once, which is an essential requirement for a business system modeling methodology (see [2] for instance). By the business process system we mean the system of mutually collaborating business processes. To understand the "business essence" of the collaboration of processes in terms of ideas of process-driven management [5] one primarily has to differentiate between two basic functional types of processes: *key* ones versus *support* ones. As customer needs are constantly changing, the processes in the

<sup>1</sup>BIR 2022 Workshops and Doctoral Consortium, 21st International Conference on Perspectives in Business Informatics Research (BIR 2022), September 20-23, 2022, Rostock, Germany


EMAIL: repa@vse.cz

ORCID: 0000-0001-9113-3447



© 2020 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

organization should change as well. This means that any process in the organization should be linked to the customer's needs as directly as possible. Thus, the general classification of processes in the organization distinguishes mainly between **Key processes**, i.e. those processes in the organization that are linked directly to the customer, covering the whole business cycle from the expression of the customer need to its satisfaction with the product/service, and **Support processes**, that are linked to the customer indirectly by means of key processes, which they support with particular products/services. The value of the key process is given by its direct contact with the value for the customer as it is its main goal. The values of other (support) processes are given by the services by which these processes support other processes. This way every process is ultimately connected to the value for the customer either directly (key process) or through its services for other processes. From these basic characteristics of the key and support processes follows another very important difference between them:

Key processes represent a specific enterprise's way of satisfying the customer needs while support processes represent rather a standard functionality often connected with some technology. Consequently:

- **key processes** are very dynamic, often changing, and permanently developing,
- while **support processes** are primarily static, and stable, offering standardized and multiple usable services, they are often tied with technology or even fully automated.

So the main effort in the process of creating the concept of the system of processes must be **establishing the equilibrium of needed dynamics of key processes** on one hand and the **necessary stability of the system ensured with its maximally standardized support processes** on the other hand.

The above-mentioned importance of the logical contents of processes invokes the consequential need for high clarity of process models. Needed activities must be ordered in the logical algorithmic units, each representing a single logical process either key or supporting, respecting also the ontological substance and relationships of objects, which they handle. Every logical unit must fulfill the requirements for being an algorithm: uniqueness, preciseness, finiteness, openness, and generality. The logical distribution of activities in processes is given primarily by the purpose and ontological meaning of processes.

On the other hand, in the physical world and the real-time, many instances of processes are running in parallel. Parallelism of actions is a natural feature of the real world. The logical sequences of activities from different processes mutually entwine in the given moment. Moreover, logical processes may require mutual interactions of activities from the processes, whose instances run in parallel. Besides a conceptual logic of processes, also all these facts have to be respected at the same time.

Concluding from the previous two paragraphs, one can see that for a successful implementation of the business system in terms of the process-driven management principles, we have to respect at the same time both the logical structure and relationships of logical processes and the natural parallelism of their physical instances. Thus, the essential question is: **How to cope with the natural parallelism of business processes with contemporaneous respect to their essential logic?**

The presented solution is based on the work of M.A.Jackson, particularly on the *program inversion technique*. In the following two sections, we briefly characterize the background methodology MMABP and the main ideas of M.A.Jackson, particularly, the program inversion technique. Then, the section with the real-world example of mutually collaborating parallel processes follows with the explanation of the problem of natural parallelism and the way of coping with it by means of process states. In the last sub-section, we discuss essential circumstances and the way of evolution of the models to demonstrate the use of the **process inversion idea**.

## 2. Methodology background

**MMABP** (Methodology for Modeling and Analysis of Business Processes) is a methodology for modeling business systems. MMABP is based on the idea of two basic dimensions of the business system: **Ontological dimension** represents the business system as a system of mutually related

objects in the standard modeling language **UML** [13, 4], The ontological model determines the contents of the business system and basic rules that all business activities in the system have to respect. **Intentional dimension** represents the business system as a system of mutually related business processes. The business processes model expresses the business goals and the ways of achieving them. Mutual relationships between business processes mean their collaboration. Both dimensions of the business system are closely related to the other. In **Intentional (behavioral) dimension**, MMABP uses the standard business process modeling language **BPMN** [1] for the model of the run (i.e. temporal aspects) of the process. For the description of the process system MMABP uses the de-facto standard **Process Map**, widely popular type of diagram originally based on so-called *process diagram* from the Eriksson/Penker methodology [3]. The basic classification of processes (key versus support) serves as a starting point of structuring the process system to the particular processes that represents an essential relation to the "business essence" of the system. Other aspects of the process system that have to be taken into account represent rather "technical" viewpoints. The most important of them is **natural parallelism**, which is the main topic of interest in this paper. A detailed and more comprehensive description of MMABP can be found in [9, 11].

The common denominator of both the global view of the whole process system (Process Map) and the detailed models of particular business processes (BPMN language) is the **cooperation of business processes**. At the same time, it is also the common denominator of both basic dimensions of the business system represented by business objects (ontological dimension) and business processes (intentional dimension). Cooperation of processes is in MMABP implemented by means of *process steps* and *process states*.

*Process step* means such a part of the process, which does not need to be interrupted by the cooperation with other processes. In other words, it represents an internal issue of the process. Every process step ends with the process state, which can be either internal or end state.

Internal *Process state* means such a point in the process structure, where nothing can be done until the input to the process occurs, i.e. point of waiting for the input. There must be an internal process state between each two neighboring process steps. In this way, MMABP models the needed cooperation of processes. Process state represents the waiting for the event that means the message from the cooperating process, usually reporting that the required service has been performed and with which result.

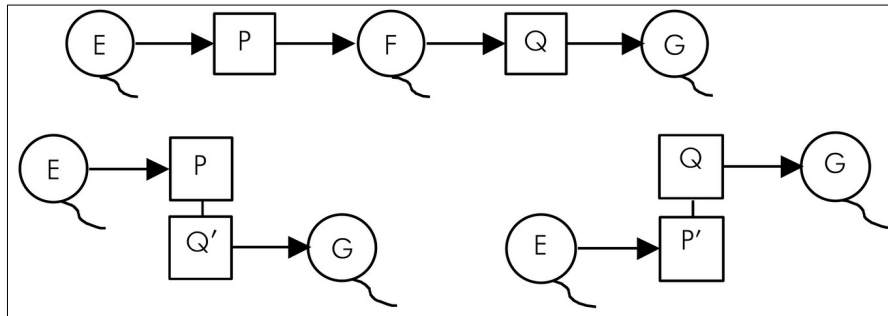
The concept of process state is present just in some process modeling standards (like IDEF [10]), partially present in some others (like ARIS [12]), some standards do not support it. Widely accepted process modeling standard BPMN ([1]) does not recognize this concept at all. The way, in which we express the process state in BPMN language is described in the sub-section Implementation environment.

In the following two sections, we describe the way, in which the **natural parallelism of business processes** can be handled with primary respect to the **logical transparency of models**. The transparency of process models is very important, especially from the methodology point of view. It is because modeling the business system is a complex and multidimensional task that requires full respect for all its dimensions. The consequences of any unnecessary complication in such a complex model are thus multiplied by its multidimensionality. Particularly, the relationships among the ontological and behavioral models require as much as possible clearness and unambiguity of all involved models to maximize the chance to address all essential issues that are mostly very abstract and may cause a lot of misunderstanding [6].

### 3. Process inversion idea

The approach to handling the natural parallelism of business processes, presented in this paper, is based on the work of M.A.Jackson. In Jackson Structured Programming (JSP [7]) the author introduces his approach to program development based on the work with data structures and a special technique for the solution of the problem of so-called "structure clash": program inversion. "The JSP technique for dealing with a structure clash is to decompose the original program into two or more programs communicating by intermediate data structures. A boundary clash, for example, requires a decomposition into two programs communicating by an intermediate sequential stream." [7]. "The

underlying idea of program inversion is that reading and writing sequential files on tape is only a specialized version of a more general form of communication. In the general form, programs communicate by producing and consuming sequential streams of records, each stream being either unbuffered or buffered according to any of several possible regimes. The choice of the buffering regime is, to a large extent, independent of the design of the communicating programs. But it is not independent of their scheduling." [8]. Later, in Jackson System Development JSD [8], the author generalizes the inversion technique as the main principle for the development of the system of programs running in parallel.



**Figure 1:** Program inversion technique [8]

Figure 1 shows how the inversion technique works. The problem of the boundary clash between the structures of streams  $E$  and  $G$  can be always solved by a division of the processing to two programs  $P$  and  $Q$ . (see the upper part of Figure 1)  $P$  creates an intermediate stream  $F$  consisting of such parts of stream  $E$  that are compatible with the structure of the stream  $G$ .  $Q$  then can simply process the stream  $F$  and produce the stream  $G$ . The lower part of Figure 1 shows two basic options of inversion. The scheme on the left side shows the inversion of  $Q$  with respect to the stream  $F$ . The former routine  $Q$  exists there as a sub-routine  $Q'$  repeatedly called in the process of processing the stream  $E$ . The former routine  $Q$  is interrupted by the processing of  $E$ . Repeating interruption requires storing the information about the state of an interrupted process in the data structure called "state vector" that contains the identification of the state and other important data (attributes) related to the process state. The scheme on the right side shows the inversion of  $P$  with respect to the stream  $F$ . The former routine  $P$  exists there as a sub-routine  $P'$  repeatedly called in the process of processing the stream  $G$ . From the operational perspective, both options are equivalent so, we can prefer the one, which is better from another perspective, for instance, from the perspective of algorithmic logic. Sequential streaming is a natural feature of business processes. It is a consequence of the flow of time as a natural dimension of the Real World. The definition of the process determines the behavior of its actors valid for all possible instances of the process. In the reality, the process exists only in the form of instances, each of them anchored in a particular time slot. All instances are thus ordered in a sequential stream and the operation of the process over multiple instances is actually the processing of the sequential stream of starting events. Since the run of the process instance consumes some time, the parallel existence of multiple instances is natural. So, the typical situation in the real-world business processes exactly meets Jackson's problem of structure clash and the use of the **program inversion** is there fully relevant. In the next two sections, we show a particular example of handling the problem of parallel process instances with the use of the "**process inversion**" technique in the real system of processes and then we discuss this approach in a more general context.

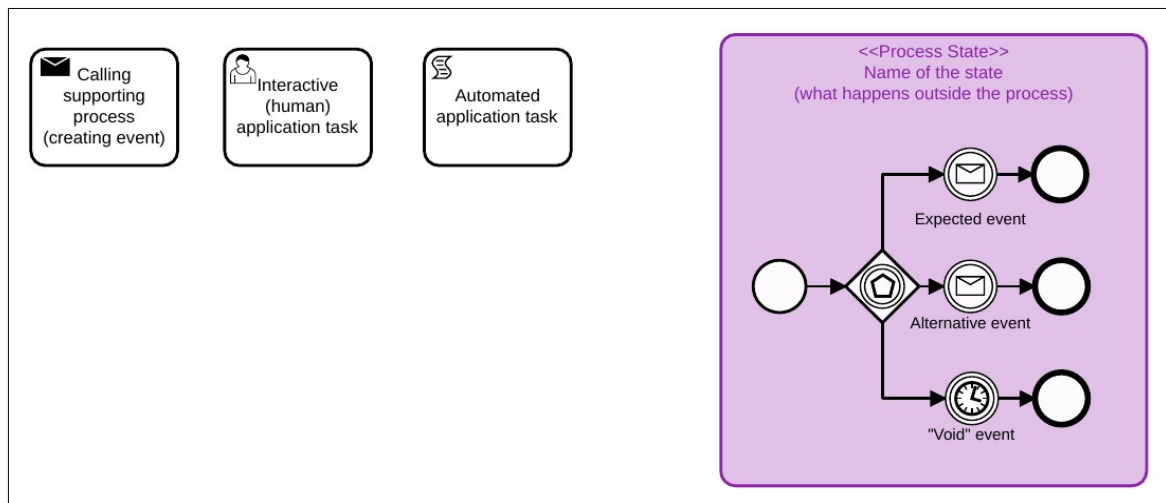
## 4. Practical example

In this section, we demonstrate the use of the "process inversion" technique in the example of a real system of processes. At first, we shortly describe the implementation environment since it significantly influences the way of description of processes, which might needlessly complicate the understanding of their contents. Then we describe the contents of the example models, and finally, we show the use of the process inversion technique for handling parallel process instances.

## 4.1. Implementation environment

To make the example understandable, we have at first to explain the way, in which we need to use the particular process implementation environment to fulfill the principles of MMABP methodology. MMABP uses for modeling the process BPMN language as a de-facto standard in the field. Since BPMN is not sufficiently in accordance with all MMABP principles, MMABP uses the so-called minimal version of BPMN reduced only to the basic elements and extended with the implementation of the *process state* that is not present in BPMN. We present our example in the workflow engine CAMUNDA, our favorite process implementation environment. CAMUNDA is absolutely loyal to the BPMN v 2.0 and it does not admit the use of the constructs not present in the standard. Moreover, it forces the user to use the constructs defined in the standard even if it does not make any special sense. All these facts force us to extend our minimal version of BPMN just to be able to implement our processes in CAMUNDA engine. Figure 2 shows the basic BPMN v 2.0 types of activity that we have to use. For the communication of processes, we need to be able to create the event, which calls for the needed reaction of another process. This is implemented in CAMUNDA as a "call activity" signed with an envelope. In this type of activity, CAMUNDA allows us to call its object correlate for creating the event. We also need to distinguish between the activities performed by a human actor (see the Interactive application task) and the activities performed automatically by the system (see the Automated application task). In the automated task (so-called script task) CAMUNDA allows us to call the script while in the interactive task we can use the inbuilt forms generator.

The right side of Figure 2 shows the way, in which we implement the process state in CAMUNDA. The process state is a point of the communication of the process with its environment (another process or actor). Technically, it represents waiting of the process for one of the possible events. Since BPMN does not recognize the concept of process state, we use for its representation the BPMN element "Event-based gateway", the only element, which allows a correct connection of the internal process flow with the events as representatives of the actions of collaborating processes. To prevent possible potential misunderstanding, we model the process state as a standalone expanded sub-process consisting only of the gateway and expected events. Process state has its original name and stereotype <<Process State>>.

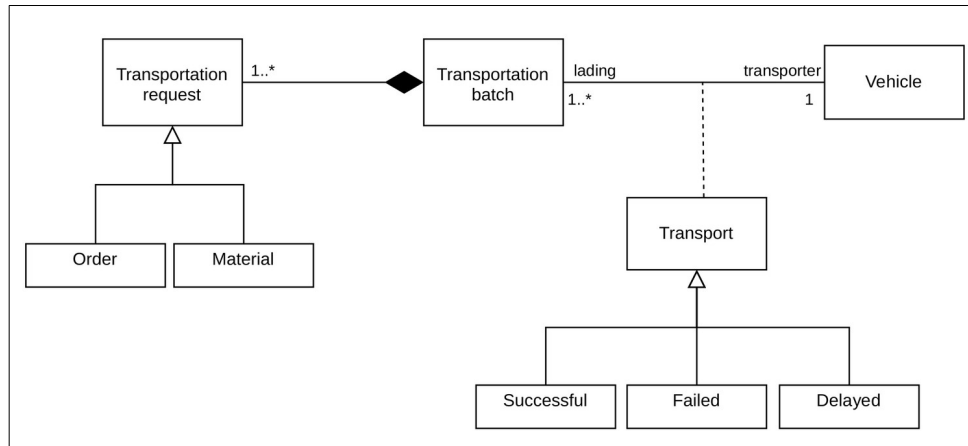


**Figure 2:** Accommodation of MMABP to the CAMUNDA BPMN v 2.0

For particular examples of the use of *process state* see the process models in the following subsections.

## 4.2. Example: transportation management

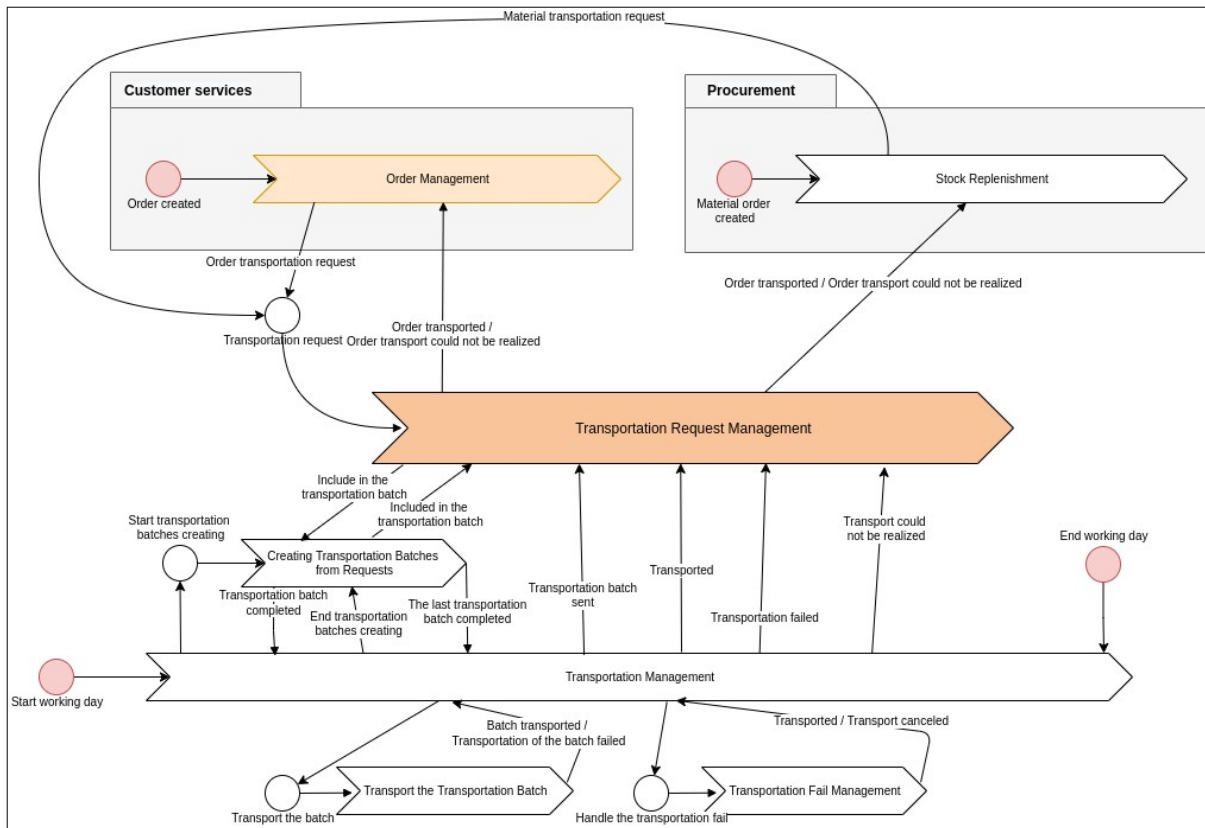
As an example of a real system of mutually cooperating business processes, we use a functional fragment of the enterprise, focused on transportation. The fragment of the conceptual model in Figure 3 shows a rough overview of the domain ontology.



**Figure 3:** Fragment of the Transportation Conceptual model

*Transportation request* is the request for the transport of either the *Order* to the company's customer or *Material* from the supplier to the company. Several *Transportation requests* are aggregated in the *Transportation batch*. One or more *Transportation batches* should be transported by a single *Vehicle*. This relationship is called *Transport* and it may be either *Successful*, *Failed* or *Delayed*.

*Process Map* in Figure 4 shows the relevant processes in this functional field and the ways of their mutual cooperation. The key process of the transportation functional area is the process *Transportation Request Management*. It serves the "customer" processes from other functional areas of the enterprise that have requests for transport. Particularly, there are processes *Order Management* from the *Customer Services* functional area and *Stock Replenishment* from the *Procurement* functional area. *Order Management* process requires transporting the final product of the customer's order to the enterprise's customer, and *Stock Replenishment* process requires transporting an ordered stock from the supplier to the enterprise. The task for the transportation processes is to ensure transportation is optimal so that they assemble the optimized transportation batches from the requests for the transportation of the orders from the company and the ordered material to the company on the way back. *Transportation Request Management* process receives the transportation request and asks the process *Creating transportation batches from requests* for its inclusion in the transportation batch. Then, it watches out the whole process of the transportation of the given request from its inclusion in the transportation batch over its sending after the completion of the batch, and finally running to the destination expecting all possible situations. Process *Creating Transportation Batches from Requests* permanently assembles the transportation batches from the transportation requests and optimizes them. Process *Transportation Management* is a main operational process of the transportation field responsible for the organization of the whole working day. It is started at the beginning of the working day and ended at the end of the working day. At first, this process starts the work in the transportation functional area by starting the process *Creating Transportation Batches from Requests* and then it handles particular created batches sending them to the transport (using the service of the supporting process *Transport the Transportation Batch*) and solving casual transportation fail (using the service of the supporting process *Transportation Fail Management*). It continuously reports every result of the batch transport to the corresponding key processes *Transportation Request Management*.



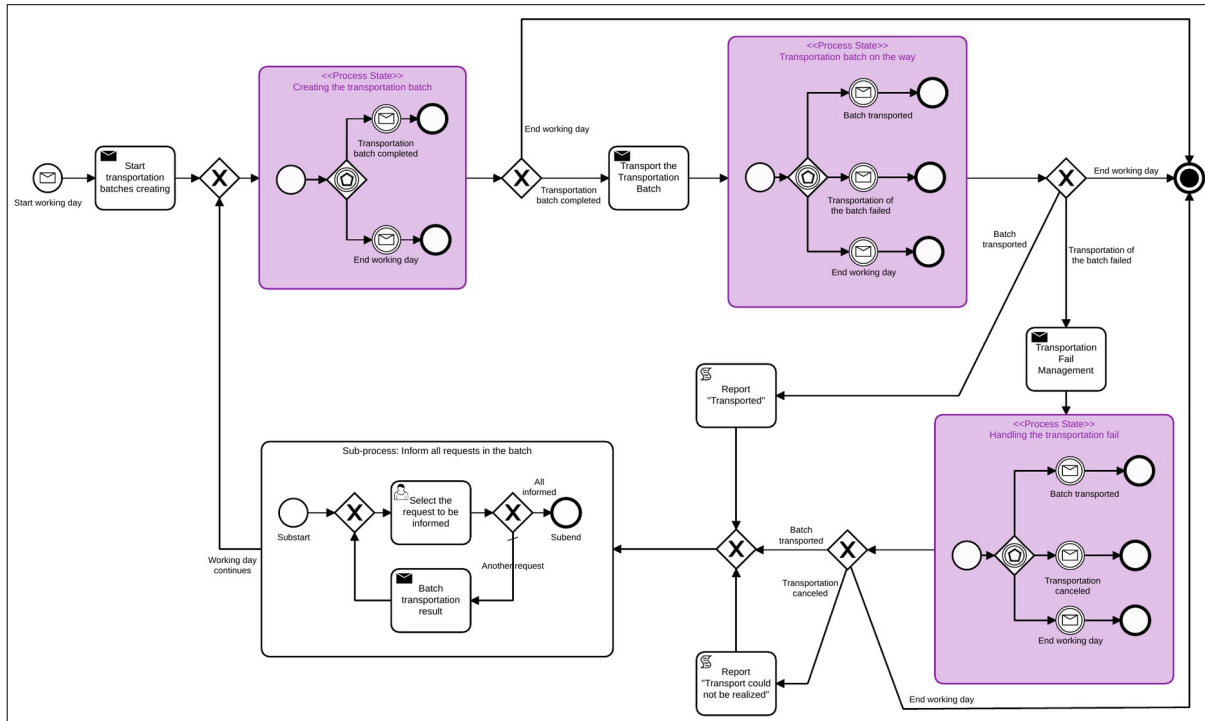
**Figure 4:** Process Map of the functional area Transportation

### 4.3. Handling the natural parallelism of cooperating processes by process inversion

In this sub-section, we discuss the way, in which the natural parallelism of the business processes can be implemented without shadowing the essence of particular processes. It is based on Jackson's technique of "*program inversion*" that allows dividing the specification of the natural structure of particular processes from their implementation as mutually cooperating procedures running in parallel. Our example shows how parallelism in business processes is natural. Every process definition represents a number of possible process instances, some of them running simultaneously. The number of simultaneous instances may be pretty high. In this example of the transportation company, there may be dozens or even hundreds of transportation requests at the same time. The company processes every request by a standalone instance of the key process *Transportation Request Management*. This process watches the whole transportation process of the request from the inclusion into the transportation batch over its physical transport up to the final successful or unsuccessful delivery. In practice, there may be dozens or hundreds of simultaneous instances of this process for a relatively long period; the transportation may last several days.

Such parallel processes are not necessarily a big problem for the organization since they are mutually independent. They do not need to mutually collaborate. The problem appears once we take into account the ontological substance of the transportation business outlined in the conceptual model in Figure 3. The *Transportation Management* process that is primarily responsible for the realization of the transport has to collect the transportation batches from the transportation requests has to cope with the fact that the transportation batch is a set of transportation requests, together with the fact that one physical transport as a unit of the possible result (not the batch nor request but the transport may fail or be delayed) consists of possibly more batches. This task contains two essentially different and relatively independent sub-tasks: the creation of the transportation batch from requests, and the creation of the vehicle load from transportation batches. Therefore, these two sub-tasks are

"outsourced" from the *Transportation Management* process to standalone supporting processes *Creating Transportation Batches from Requests* and *Transport the Transportation Batch*. Each of them is specialized in one sub-task and its structure is relatively simple and understandable. This outsourcing also somewhat simplifies the structure of the supported process *Transportation Management*.



**Figure 5:** Transportation Management process, a sequential version

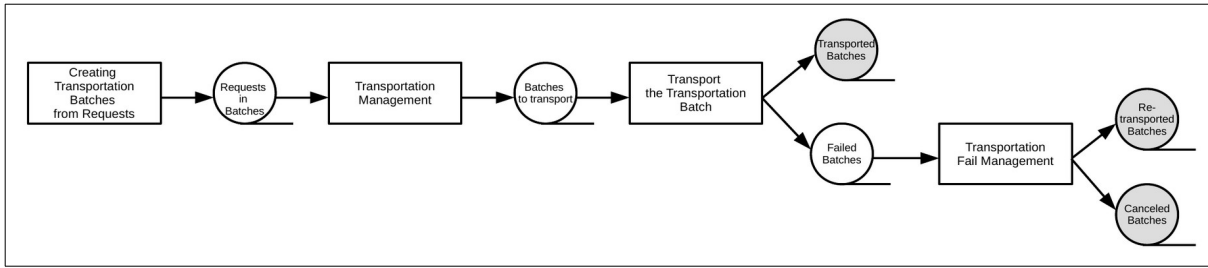
Outsourcing the part of the process structure to the supporting process always requires calling the supporting service and consequential waiting for the result in the process state. In the sequential version of the *Transportation Management* process (see Figure 5), the process at first starts the creation of the transportation batch from requests and repeatedly waits for the created batches. For every created batch, it orders the transport and waits for its result. After the corresponding actions according to the result of the transportation, the process informs all requests from the batch(es) and returns to the beginning to call the creation of the next transportation batch until the end of the working day.

Such an approach cannot be used in real-world business for two basic reasons:

- The first one is that it delays the creation of the next transportation batch until the transportation of the previous batch is finished. Respecting the number of simultaneously existing requests and the time of the whole transportation, such a delay is not acceptable even if this problem may not be regarded as critical in the case of a small number and frequency of requests.
- The second reason is absolutely critical. Such a process contains a certain deadlock coming from the fact that multi-batch lading of the vehicle can never be finished since the process cannot create a new batch before the delivery of the previous one. In such a situation, the process infinitely waits.

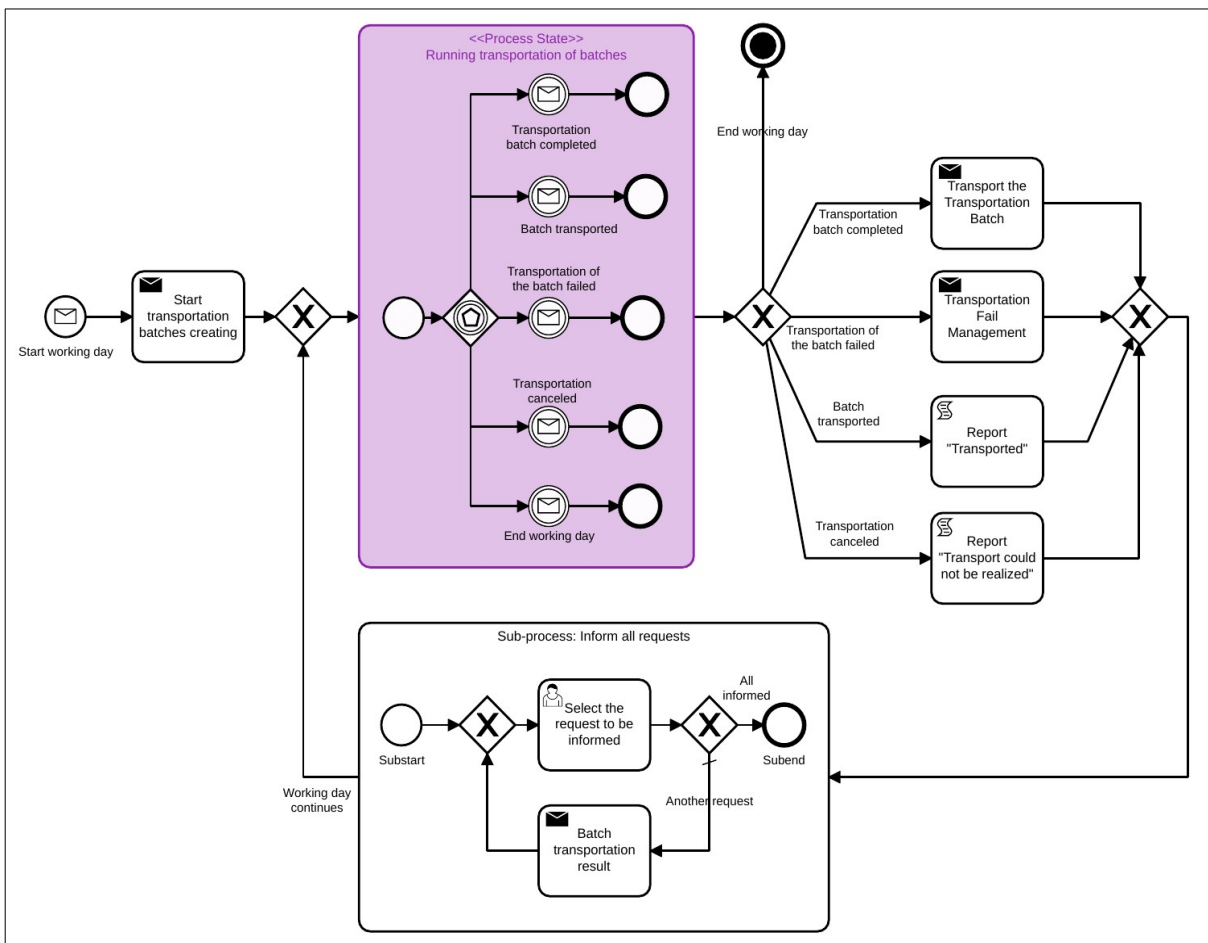
Obviously, transportation batches have to be created relatively independently of the physical transportation but coordinated with it. The creation of the batch must not be delayed by waiting for the transport and no one transport can be delayed by the other simultaneously running transports.





**Figure 6:** Sequential version of the Transportation Management, Jackson's point of view

The solution is freely based on Jackson's idea of *program inversion*. Of course, we do not use the program inversion technique directly since the process-driven processing is not the same as the traditional batch processing. We just use the idea of the difference between sub-routines and co-routines and following consequences excellently defined by Jackson. To apply the idea of Jackson's inversion technique let us look at the sequential processing in Jackson's traditional view in Figure 6. Created batches are processed by the *Transportation Management* routine that creates the file *Batches to transport*, which is subsequently processed by the *Transport the Transportation Batch* routine that creates two files: *Transported Batches* and *Failed Batches*. The second file is subsequently processed by the *Transportation Fail Management* routine to the two resulting files *Re-transported Batches* and *Canceled Batches*. To transform the sequential processing to the parallel one, we then can invert existing logical sub-routines making them the co-routines, and remove the sequential files. The result in the BPMN notation can be seen in Figure 7.



**Figure 7:** Transportation Management process after inversion

The supporting process *Creating Transportation Batches from Requests* has been inverted to the process *Transportation Management* with respect to the event flow *Requests in Batches*. The supporting process *Transportation Fail Management* has been inverted to the process *Transport the Transportation Batch* with respect to the event flow *Failed Batches*. Finally, the process *Transport the Transportation Batch* including already inverted process *Transportation Fail Management* has been inverted to the process *Transportation Management* with respect to the event flow *Batches to Transport*. In this version of the process, all events are expected in a single common state *Running transportation of batches*. Therefore, the instances of all three supporting processes can run independently in parallel. Every particular instance of the supporting processes has to be identified since it represents the particular sub-routine (i.e. the transportation batch processed). In this way, the so-called state vector (in Jackson's terminology) is implemented. In CAMUNDA, the identifier of the process instance is called "*business key*", which is in our implementation actually the identifier of the processed transportation batch. Independent creation of the transportation batches is coordinated with the *Transportation Management* process only via the event *Transportation batch completed*. Independent performance of the physical transports is coordinated with the *Transportation Management* process via events *Batch transported*, *Transportation of the batch failed*, and *Transportation canceled*. All instances of the supporting processes *Transport the Transportation Batch* and *Transportation Fail Management* can run simultaneously and are handled by a single instance of the *Transportation Management* process together with the transportation batches repeatedly created by a single instance of the process *Create Transportation Batches from Requests* from the individual transportation requests that continuously come from the key process *Transportation Request Management*. The inversion only changed the structure of the process *Transportation Management*. Other (supporting) processes remain the same as in the sequential version, their structures are simple and understandable, and each of them can focus on its subject of interest.

## 5. Conclusions

Process inversion, presented and discussed in the previous sections allows handling the natural real-world parallelism of business processes with keeping the structure of processes transparent with respect to the ontological essence of the given real-world domain. This fact is very important from the analytical point of view. The relationship between the system of business processes and the ontological essence of the business domain is a critical aspect of the conception of the business process system, and it is also one of the basic principles of MMABP methodology. The ontological essence of the business domain determines possible business processes since the Real World ontology determines possible intentions and the ways of achieving them. The work of M.A.Jackson is still a commonly underestimated pool of valuable knowledge that can be used especially in the field of business process modeling and management. In MMABP, Jackson's work plays an important role not only as we present it in this paper. The concept of so-called structural consistency of models, which is the main MMABP tool for achieving their general quality, is based on the generalization of Jackson's idea of the correspondence of structures, which is the root of his whole work. Nevertheless, there still exist the ideas in his work that can be exploited in the field of business process modeling and management, like the *Backtracking technique*, for instance.

## 6. Acknowledgements

This paper was supported by the Prague University of Economics and Business, Faculty of Informatics and Statistics.

## 7. References

- [1] Business Process Model and Notation (BPMN), 2011. OMG Doc. No.: formal/2011-01-03, Standard document URL: <http://www.omg.org/spec/BPMN/2.0>.
- [2] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-33143-5>.
- [3] Eriksson, H. E., and Penker, M. *Business Modeling with UML: Business Patterns at Work*. New York: John Wiley and Sons, 2000.
- [4] Guizzardi, G.: *Ontological foundations for structural conceptual models*. University of Twente. Centre for telematics and information technology, Telematics instituut, Enschede (2005).
- [5] Hammer, M., Champy, J.: *Reengineering the corporation: a manifesto for business revolution*. Brealey, London (1993).
- [6] Hammer, M.: The Process Audit. *Harvard business review*. 85, 111-9, 122 (2007).
- [7] Jackson, M.A.: *Principles of program design*. Academic Press, London (1983).
- [8] Jackson, M.A.: *System development*. Prentice/Hall, Englewood Cliffs, N.J (1983).
- [9] Repa, V., Svatos, O.: *Adaptive and Resilient Business Architecture for the Digital Age*. In: Zimmermann, A., Schmidt, R., and Jain, L.C. (eds.) *Architecting the Digital Transformation*. pp. 199-221. Springer International Publishing, Cham (2021).
- [10] Mayer, R.J., Menzel, C.P., Painter, M.K., deWitte, P.S., Blinn, T., Perakath, B. (1997): *iDEF3 Process Description Capture Method Report*, Knowledge Based Systems, inc.
- [11] Repa, V. *Essential Challenges in Business Systems Modeling*. In *Information Systems: Research, Development, Applications, Education - 10th SIGSAND/PLAIS EuroSymposium 2017*, Gdansk, Poland, September 22, 2017, Proceedings, 99-110, 2017.
- [12] Scheer, A.-W. (1992) "Architecture of integrated Information Systems - Foundations of Enterprise-modeling", Berlin.
- [13] *Unified Modelling Language Infrastructure Specification, version 2.4.1*, Object Management Group (OMG) (<http://www.omg.org>).