Resource Requirements in Business Process Modelling from an Operations Management Perspective

Asvin Goel
Kühne Logistics University
Hamburg, Germany
asvin.goel@the-klu.org

Min-Bin Lin
Humboldt Universität
Berlin, Germany
linmibin@hu-berlin.de

***Abstract*—Operations management entails the design and control of business operations with the goal of providing goods and services as efficiently as possible. Usually various resources are required to conduct operations and the required resources are usually limited in numbers and availability. Business process models supporting operations management should aid in effectively using available resources, identifying process inefficiencies, and enabling appropriate action if resources are unavailable or cannot operate as required. In this work we show how typical resource requirements found in operations management can be included in business process models and propose BPMN 2.0 compatible modelling patterns for describing resource requirements based on the concepts of *requests* and *releases*. We discuss various modelling requirements from an operations management perspective and discuss how request-release modelling patterns can be used for different application cases.**

***Index Terms*—business process modelling, resource requirements, BPMN 2.0**

I. Introduction and related work

Operations management is a multidisciplinary domain responsible for managing business processes which create goods and services [1]. It includes: planning, organising, coordinating, and controlling the transformation of inputs into desired outputs. Operations managers spend a large share of their time on efficiently managing expensive resources like skilled workforce or machines and it is of utmost importance that the utilisation of such scarce resources is well aligned with the requirements of business processes. Furthermore, cost cutting initiatives, such as lean management and just-in-time strategies [2, 3], have led many companies to reduce inventory levels. Low inventory levels, however, bear the risk of stockouts, if required parts and components are not available when needed. As stockouts can be a major cause of delays and can have a strong impact on the performance of processes, operations managers must carefully balance decisions concerning the availability of staff, machines, parts, and components with the efficiency of the processes.

Business process model and notation 2.0 (BPMN 2.0) can explicitly illustrate the interactions among the operational activities, and has become a de facto standard for process modelling in various fields [4]. Giving graphical notation to business processes facilitates the capability to communicate a wide range of information and to cooperate with various types of users in a well-defined manner. Operations managers can comprehensibly understand workflow logics and conduct process improvement by using BPMN 2.0. However, the resource perspective of BPMN 2.0 is limited. With so-called swimlanes, a process modeller can indicate which resource is responsible for the execution of particular activities. However, BPMN 2.0 does neither provide information on whether the needed resource is actually available at the time when the activity is to be conducted, nor does it give any indication which particular resource shall be assigned to the activity if multiple suitable resources are available. Furthermore, swimlanes cannot be used to model the simultaneous requirement of multiple resources and are not suitable for passive resources like raw materials, parts, and components.

In order to support limited availabilities of resources required by processes, [5] tackles resource allocations using answer set programming. A *Resource Assignment Language (RAL)* allowing to model which human resource can perform which activity is proposed in [6]. In the visual modelling language for human resources proposed in [7], requirements demanding that the same resource is used for different tasks can be modelled. A language allowing to support the simultaneous allocation of multiple human resources is presented by [8].

With the *Business Process Simulation Specification (BPSim)* [9] it is possible to specify simulation-relevant data such as resource roles, quantities, and availabilities. However, the additional data is not included within the BPMN model. A critical discussion of BPSim is given by [10]. To consider more complex resource requirements that cannot be simulated with BPSim, [11] propose to extend BPMN 2.0 introducing the concept of a *shared task*. Here, *shared tasks* are used to indicate tasks that share the same resource with a limited capacity. Tasks belonging to different process instances can occupy some of

2378-1971/22/$31.00 ©2022 IEEE
DOI 10.1109/CBI54897.2022.10047

the capacity of the resource and the token flow of different process instances is no longer independent. [12] propose an execution semantics for so-called *batch activities* allowing to dynamically generate batches across multiple process instances.

In discrete-event system simulation, resource requirements are explicitly considered and useful modelling patterns exist since the 1960s. In the simulation language GPSS, the *seize-delay-release* modelling pattern is used allowing to allocate resource to activities. Between the *seize* and *release* no other task can be conducted by the resource. A *delay* is specified to ensure that the time between the *seize* and *release* of the resource is sufficiently large [13]. To tackle the lack of similar considerations of resource requirements in Business Process Modeling, [14] introduces the concept of *resource-constrained activities* and proposes to extend BPMN by adding visual markers to sequence flows in order to indicate that the subsequent activity requires a resource. Complex resource requirements demanding that the same resource is allocated to multiple activities are not considered.

A comprehensive survey on the literature related to resource allocations in business processes has been conducted by [15]. One of their findings is that the variability of many approaches is limited and does not facilitate an easy adaptation for different use cases.

Despite the various approaches to consider resource requirements in business process modelling and BPMN 2.0, so far most approaches have focussed on a particular feature and use case and it is unclear how the various approaches can be combined for heterogeneous use cases in operations management. A query on the EBSCO research database revealed only seven results when searching for articles published in the years 2011 to 2021 which contain the term *BPMN* in the text (TX All Text) and which are published in journals containing the term *operations* or *operational* in the title (SO Publication Name). This indicates that BPMN 2.0 has not found much recognition in the operations management and operations research communities. The authors believe that the limited support of resource requirements in BPMN can be seen as one of the main reasons for this negligence.

This paper provides an overview over various requirements on modelling resource requirements from an operations management perspective. Most importantly we propose modelling patterns using dedicated BPMN 2.0 subprocesses for representing resource requirements and we propose a visual representation to ease modelling with this pattern. By satisfying the observed requirements, the proposed modelling patterns help to bridge the gap between operations management and BPMN 2.0 and facilitate the development of information systems for simulating, automating, and optimising the execution of business processes.

II. Requirements

Based on the authors' experience over two decades of teaching and conducting research and projects in the field of logistics and supply chain management, this section provides several requirements on modelling resource requirements commonly found in operations management.

R1 **Availability**. The availability of resources is usually limited. Some resources are only available during certain time intervals and in certain quantities. Furthermore, resources may be consumed by processes and resources may be temporarily reserved for the execution of specific tasks. Operations managers require an explicit representation illustrating from when to when resource are needed. This helps operations managers in identifying potential bottlenecks as well as in acquiring additional resources to prevent bottlenecks.

R2 **Consistency**. Multiple tasks may have to be conducted by the same resources. For example, the loading and unloading task for a transportation process must be conducted with the same vehicle. Operations managers must be able to specify such requirements.

R3 **Heterogeneity**. In most operations heterogeneous resources are available with different characteristics and capabilities. For an efficient process execution it is necessary that a resource request can be made based on the required characteristics and capabilities of resources. Therefore, operations managers must be able to specify requirements on resources in multiple ways, e.g., by providing an identifying name, by providing the name of suitable roles, or by explicitly describing the required capabilities of resources.

R4 **Pooling**. Multiple resources may be needed simultaneously to execute a task. For example, in a production process, a semi-automatic machine and a human operator may be simultaneously needed to execute a task. Operations managers must be able to specify such requests for pools of resources. Furthermore, it must be possible to add and remove resources from the pool when needed.

R5 **Collaboration**. It may be necessary to conduct complex interactions with resources during process execution. Operations managers must be able to specify such interaction with the requested resources by modelling a suitable collaboration.

R6 **Reactivity**. Processes must be able to react on unpredictable events or errors that may occur within the scope of a required resource. For example, a process waiting for a resource to become available should have the possibility to react when being informed that the requested resource encounters a problem. The process should have the possibility to cancel the request and proceed with some appropriate countermeasure if necessary. Operations managers must be able to specify such event-specific execution paths.

R7 Flexibility Resources can be allocated in different ways, e.g., by manual decisions, by decision rules, or by optimisation algorithms. Furthermore, multiple requests may be consolidated before allocating the resulting batch to a resource for efficient execution. The process model should not impose limitations on the allocation logic used and on whether requests shall be consolidated before allocating a resource.

While some of the requirements, in particular, requirements R1 to R4 have already been considered in prior work, the authors are not aware of any approach fulfilling all of the above-mentioned requirements. In particular, requirements R5 to R7 have received very little attention, if any. From an operations management perspective these requirements are of utmost importance because the need to interact with resources and the need to react on events arising within the scope of the resource are essential to many operations as shown in the application examples in a later section. Furthermore, the processes and respective resource requirements should be modelled in such a way that the models remain valid, even if decision processes for resource allocations change or new technologies like 3D-printers, for example, allow an efficient process execution for individual process instances, although, traditionally, batching of multiple production orders was necessary to ensure an efficient production.

After presenting modelling patterns fulfilling these requirements in the next section, Section 4 provides some application examples showing how the requirements are fulfilled by the proposed modelling patterns.

III. Modelling resource requirements with *Requests* and *Releases*

Based on the *seize-delay-release* modelling pattern we propose to fulfill the requirements stated above by using dedicated activities for requesting and releasing resources. The behaviour of these activities is represented by the subprocesses shown in Figures 1 and 2. A request activity is used to indicate that one or more resources are needed for the process to continue. The release activity is used to indicate that one or more resources are no longer needed for the process instance and can be allocated to other process instances.

The request activity starts with sending a message to the resources required[1]. Each of these resources responds with a ready message when the resource is available for being used by the requesting process. In many cases, the requested resources are not available promptly and the requesting process has to wait until all resources are ready to be used. When all requested resources have indicated that they are ready, the request informs all involved resources via a start message. From this moment on, the requesting process knows which of the resources have been assigned and can collaborate with each of them.

[1]Note that we use implicit start and end events for a compact representation.

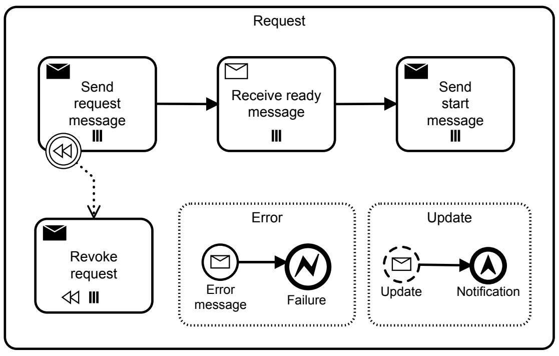

Fig. 1. Request activity

While waiting for the requested resources to become ready, the request activity can react on status updates provided by the resources. This is achieved using the "Update" event-subprocess. A non-interrupting boundary event can be attached to the request activity to react on such notifications. For example, the resources may provide information about the estimated time when the resources will be available. In such cases the requesting process may initiate adequate measures to prepare accordingly.

Furthermore, a resource can inform the process that it has to reject the request, e.g., if it is unable to fulfil the request. This is achieved using the "Error" event-subprocess. The process can react on an error message by terminating the request and initiating some alternative actions.

Lastly, the requesting process can also interrupt the request, e.g., when waiting too long for the resources to become ready. This can be achieved using an interrupting event attached to the boundary of the request activity. Note that a process engine has to ensure that the request is revoked whenever the request activity fails or is interrupted.

Whenever the request is rejected or cancelled, a message is sent to all requested resources to inform them that they are no longer needed.

After a request is completed, all requested resources are available to the process instance. The interaction with of the process instance with the resources can be modelled as a collaboration.

When the resources are no longer needed, the process can release the resources using the release activity shown in Figure 2. The release activity starts with sending a message to all specified resources and indicates that the process instance is at a state where the resource can be released. When the resources are ready to be released, they respond accordingly, and thereafter, the release activity sends a message that the release of the resources may now be completed. In many cases, resources will immediately respond to a release message. However, in some cases it may take some time until the resource is ready to complete

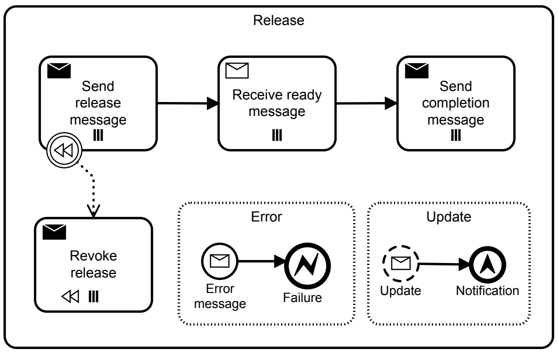

Fig. 2. Release activity

the release. For example, if a request for a vehicle requires that the vehicle transports goods from one place to another, loading and unloading activities may be modelled within the process requesting the transport. The transport activity, however, is part of the process of the allocated vehicle and details may differ depending on the type of the vehicle. The process instance which had initiated the request may send the release message directly after the goods are loaded at the origin. The release can only be completed when the transport activity is completed and the vehicle has arrived at the destination. Thus, the vehicle will only respond with the done message after completion of the transport activity.

During the time that a process instance may have to wait for the release to be completed, the process instance can react on status updates and errors similarly to the request activity. Additionally, the process may revoke the release activity without waiting for a successful completion to proceed. In this case, all resources receive a message that the release activity has been revoked and can react accordingly.

It is possible to use the request and release activities shown in Figures 1 and 2 for modelling of processes in which resources are required. However, modelling the details of these activities would be cumbersome and impractical because many resource requirements may have to be considered. We therefore propose a graphical extension for BPMN 2.0 that simplifies the modelling of request and release activities. In the graphical notation *Requests* and *Releases* are depicted by three connected rectangles representing a horizontal stack or queue. A *Request* is drawn with normal line width, in analogy to a catching event waiting for the requested resource(s). A *Release* is drawn with a thick line, in analogy to a throwing event returning the requested resource(s). Figure 3 shows an example using the proposed graphical notation that can be used to model resource-constrained business processes.

In the figure, the process starts with requesting a resource using a request activity. When the requested resource is ready, the process can proceed with a subprocess

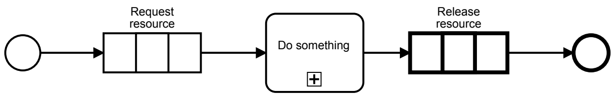

Fig. 3. Graphical representation of requests and releases

collaborating with the resource. After completion of this subprocess, the process releases the resource by a release activity.

If multiple resources with different characteristics are available, the requested resources can be specified by their distinctive names or the description of their required roles or capabilities.

The boundary events shown in Figures 1 and 2 can be attached to the request and release activities like for normal activities. This allows the modeller, e.g., to use an interrupting timer event to abort the request if too much time has passed before the resource is provided. Similarly, an interrupting error event can be applied to consider the possibility that the request is rejected by a resource. Any interrupting boundary event results in a cancellation of the request and triggers the compensation shown in Figure 1 to revoke the request. Moreover, non-interrupting boundary events can also be used, e.g., to initiate certain tasks if a requested resource is not allocated within a certain time limit, but the process shall continue waiting for the allocation of the requested resource.

When a notification from the resource is received, a non-interrupting boundary event can be used to react on this notification. For example, an interrupting escalation event attached to the boundary of the request activity can be used to interrupt the request when a notification is received from the resource that it cannot serve the request within a reasonable amount of time.

In some resource-constrained business processes, the requested resources are consumables that can only be used once. The consumption of resources can easily be modelled by using a request for the consumable resource without a subsequent release.

IV. Application examples

This section showcases application examples in which the requests and release activities presented above can be used to model resource requirements typically found in operations management. The modelling pattern has also been applied on a variety of other use cases related to production, transportation, and inventory management.

A. Batch Transport

Figure 4 shows a manufacturing process requiring a transport of a semi-finished good from one production stage to another. After the semi-finished good is produced in the first production stage, a transport to the second production stage is requested. Directly afterwards, the transport resource is released and the process waits for the resource to confirm the release. After that, the production process continues with production at the second production stage.

From the perspective of the manufacturing process, it is not important how the semi-finished good is transported from one stage to another. It is only important to ensure that a suitable resource is available to conduct the transport. For an efficient transport, however, it may be beneficial to consolidate multiple transport requests for a batch transport. In the example shown in the figure, the batch transport process collects multiple transport requests until a batch is ready. Then all requests are informed that the batch is ready and, when all requests have responded with the start message, the batch transport is conducted. After completion of the batch transport, the release messages are received [2] for each item in the batch and each of the release activities is informed that the transport is done and the release can be completed. The transport process ends after all release activities have responded with the completion message.

B. Healthcare

Figure 5 shows a healthcare process for a patient requiring an emergency surgery at a hospital. The process starts with a request for a surgeon, nurse, assistant, and an operating theatre. Unlike in the production request illustrated before, it is important that all of these resources are allocated simultaneously. This allows an allocation engine or a human planner to identify that all requested resources are needed simultaneously. Thus, resource allocation plans can be determined accordingly and unproductive times of scarce and expensive resources can be avoided. When all of the requested resources are ready, the surgery can start. The details of the surgery can be modelled as a collaboration diagram between the different resources. To ensure that only such surgeons, nurses, and assistants are allocated to the request are capable of collaborating in the surgery, the request can specify the required qualifications. After the surgery is completed, the surgeon, nurse, and operating theatre are released and the assistant moves the patient to a hospital room before being released as well.

If the required resources are not immediately available, it is possible that the patient deceases. In this case, the requested resources are no longer required, however, a physician must issue the death certificate. In this case any available physician can issue the death certificate, including the surgeon who might have been allocated to the initial request. However, the surgeon may also be assigned to another process instance where he or she may be needed more urgently. The urgency of the tasks can be provided as additional parameters of the model.

V. Discussion

The examples provided in the previous section show how the different requirements identified earlier can be fulfilled with the proposed modelling patterns for *requests* and *releases*. By using the proposed graphical representation, the effort for the modeller is significantly reduced and process execution engines can identify that the modelling pattern is used.

A benefit of this explicit way of modelling resource requirements is that it highlights those parts of a process that are dependent on the availability of suitable resources. The visual representation makes it easy for operations managers to identify when which resources are needed. It helps identifying potential causes of delays and inefficiencies, e.g., if resources are not available when needed or if resources are being held unnecessarily without being needed. Also it gives operations managers the power to adjust the resource requirements according to the specific business needs.

Figures 6 and 7 show two alternative processes requesting one type of resource to conduct two subsequent activities. The process shown in Figure 6 comprises a dedicated request and release for the two activities. The process shown in Figure 7 requests a resource, conducts the two activities, and eventually releases the resource. Despite their similarity, both processes can have a fundamentally different performance. The process shown in Figure 6 allows that the resource can conduct activities for other process instances between conducting the first and second activity of a particular instance. This can be particularly useful in case additional urgent requests become known after the resource is initially allocated. For the process shown in Figure 7, on the other hand, we can be sure that for all process instances the resource conducts the second activity immediately after the first. Thus, requests from other process instances have to wait until the second activity is completed. However, the makespan, which is the time between the start of the first activity and the end of the second activity, is minimised for all process instances. Depending on the particular business needs, an operations manager may want to explicitly decide for one or the other way of defining resource requirements in business processes.

In the *Batch Transport* example presented above, we modelled the interaction with the transport resource assuming that the process requiring transportation knows which particular resource it requests and sends a request message directly to this resource. We did so mainly for simplicity, and want to note that in most real-life cases, we would use indirect requests and releases via a broker of resources. We can use a resource manager as a broker and send all resource request and release messages via this resource manager [16]. The resource manager will allocate suitable resources to the requests and manage the information flow between requests and releases on one side and the resources on the other side. Thus, the *Manufacturing* process shown in Figure 4 would not be in direct collaboration with the *Batch Transport* process. Instead, a resource manager would check whether

[2]Note, that we assume that messages are persistent.

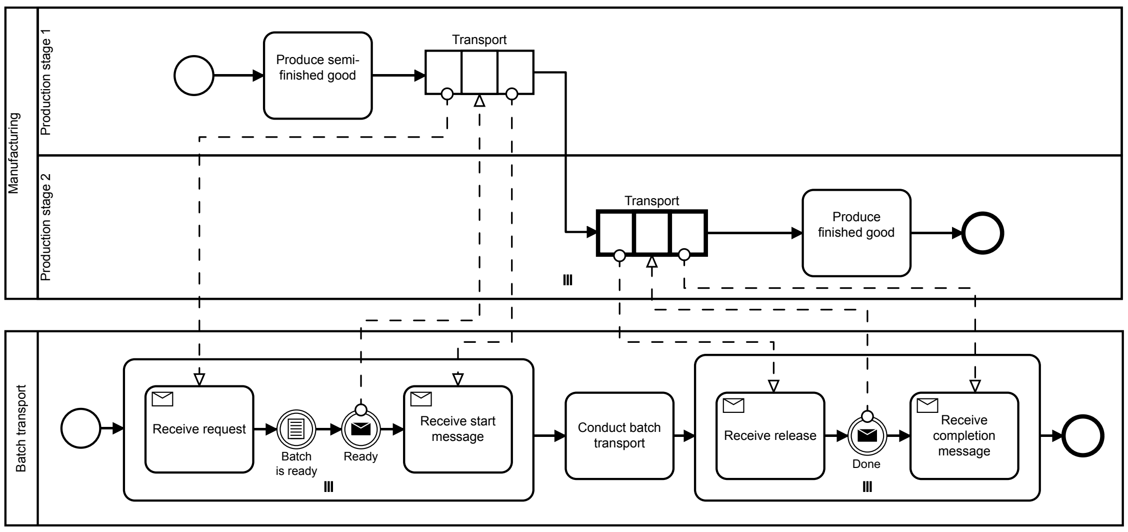

Fig. 4. Batch transport

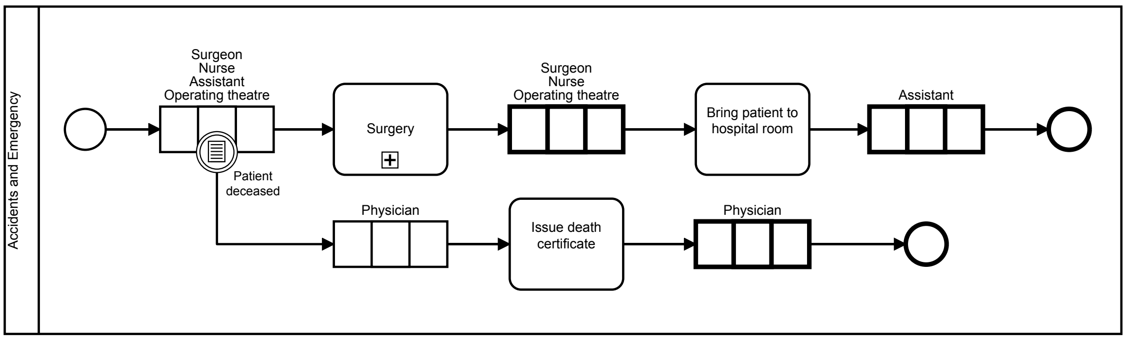

Fig. 5. Healthcare process

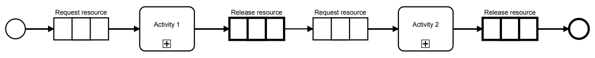

Fig. 6. Two subsequent activities with dedicated resource requests

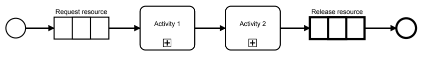

Fig. 7. Two subsequent activities with a single resource request

a suitable resource is available to conduct the transport. This could be the resource providing the *Batch Transport* process or any other suitable resource, e.g., a resource with smaller capacity that can only provide dedicated transport of individual requests. From the perspective of the *Manufacturing* process, any suitable resource would be equally good, however, in some cases an allocation to one resource would be more cost efficient, and in other cases, an allocation to another resource would be more cost efficient. The resource manager, which could be a human or an algorithm, can base the allocation decision on different criteria. The simplest allocation mechanism would operate according to a first-come first-serve policy. That is, the first request is allocated to the first suitable resource. More sophisticated allocation mechanism would be priority-based decision rules or rules based on suitable performance metrics. Furthermore, powerful operations research techniques, in particular, from the area of *resource-constrained scheduling* [17] can be used to improve the overall operational performance.

Our modelling patterns simplifies the translation of business process models into mathematical optimisation problems that can be used to determine highly efficient execution schedules. In fact, providing the basis for the application of operations research techniques for improving the performance of business process models is one of the core motivations for this paper. Besides this, our modelling pattern also bridges the gap between standard BPMN 2.0 and typical discrete-event system simulation models in which resource requirements play a central role and request-release patterns are well accepted modelling elements.

VI. XML-Extension and implementation

We implemented the proposed notation for requests and releases using the BPMN 2.0 extension mechanism. In particular, request and release activities are modelled as BPMN subprocesses with a custom attribute indicating the type as shown below.

```
<bpmn2:subProcess id="some_id" resources:type="Request">
```

and

```
<bpmn2:subProcess id="some_id" resources:type="Release">
```

In our implementation we assume that at run time resources are dynamically allocated to requests via a resource manager. In order to allow the resource manager to allocate suitable resources, additional data must be provided. Here, we assume that such data can be made available via status attributes that can be arbitrarily defined and modified throughout process execution. These status attributes can be defined on the process level as follows.

```
<bpmn2:process id="some_process">
  <extensionElements>
    <execution:status>
      <execution:attribute id="some_id"
      ↪ name="some_attribute" type="xs:decimal"
      ↪ value="3.14" />
      <execution:attribute id="other_id"
      ↪ name="other_attribute" type="xs:integer" value="42"
      ↪ />
    </execution:status>
  </extensionElements>
</bpmn2:process>
```

A `<execution:status>` block can contain any number of `<execution:attribute>` elements. Each `<execution:attribute>` must specify a unique `name` and the data type, e.g., `type="xs:integer"` or `type="xs:decimal"`. Optionally, a value can be given by adding an attribute `value` that is to be used across all process instances or by providing the value on an instance level via a separate input.

We assume that each request activity can make multiple requests for different resource allocations. Each request provides a vector of values x_j describing the requirements of the job j that the resource must conduct. It is assumed that for each resource r a function $f_r(x_j)$ is available that returns true if and only if the resource r is capable of conducting the job. The data is provided as follows.

```
<bpmn2:subProcess id="some_id" resources:type="Request">
 <bpmn2:extensionElements>
  <execution:allocations>
   <execution:request id="some_request_id">
    <execution:job>
     <execution:content id="a_content_id" key="a_key"
      ↪ attribute="some_attribute_name" />
     <execution:content id="another_content_id"
      ↪ key="another_key"
      ↪ attribute="another_attribute_name" />
    </execution:job>
   </execution:request>
  </execution:allocations>
 </bpmn2:extensionElements>
</bpmn2:subProcess>
```

For each request activity, a set of resource allocation requests is stored in an element `<execution:allocations>`. The data for each requested resource is stored in an element `<execution:request>` with a unique identifier. Each request contains an element `<execution:job>`, listing the characteristics of the job to be conducted as key-value pairs. The description of the characteristics is as general as possible to allow to provide identifiers of particular resources, the name of a role that a resource must provide, or general requirements, e.g. the width, height, length, and weight of an item to be shipped. Thus, different resource allocation patterns as proposed by [18] can be supported.

The release activities are defined as follows.

```
<bpmn2:subProcess id="some_id" resources:type="Release" >
 <bpmn2:extensionElements>
  <execution:allocations>
   <execution:release request="a_request_id">
   <execution:release request="another_request_id">
  </execution:allocations>
 </bpmn2:extensionElements>
</bpmn2:subProcess>
```

Each release activity provides information on which resources are to be released by custom extension elements. The list of resource allocations to be released is stored in an element `<execution:allocations>`, which contains elements `<execution:release>` that specify the identifier of the request to be released in the attribute `request`.

A prototype of a modeller allowing to model such resource requirements is available online at https://bpmn.telematique.eu. This prototype already includes a model to describe capabilities of resources and data concerning the status of a resource in order to assess the value of a resource allocation. A description of this resource model, however, is out of scope of this paper which focuses on the modelling of resource requirements.

VII. Final Remarks

This paper shows how typical resource requirements from operations management can be included in business process models in order to help operations managers to effectively balance the tradeoffs between efficient resource utilisations and process performance. We propose modelling patterns of *requests* and *releases* and propose a graphical representation as an extension to BPMN 2.0. With the dedicated activities for requesting and releasing resources, our extension allows operations managers to explicitly model resource requirements in business process models. Several application examples are provided illustrating how the proposed modelling pattern can be used.

The proposed modelling pattern allows operations managers to easily identify process delays caused if no suitable resource is available at the time requested (**R1 Availability**). Where necessary, operations managers can use this insight to identify possibilities of improving process performance by redesigning the temporal dependencies on resources or by providing additional resources.

The proposed modelling pattern allows to ensure that the same resource is available for multiple activities modelled between the request and release of a resource (**R2 Consistency**).

To allow a high level of flexibility we did not make any assumptions on the behaviour or nature of resources, except that resources must be capable of collaborating with request and release activities. Different resources with different characteristics can be arbitrarily modelled and allocated to requests at run time (**R3 Heterogeneity**).

Processes requiring that multiple resource are simultaneously available can be considered by requesting multiple resource within the same request activity (**R4 Pooling**).

As our modelling patterns assume that resources are provided through dedicated process models, complex collaborations between process instances requiring one or multiple resources and those providing resources can be easily modelled (**R5 Collaboration**).

By allowing to use catching events attached to the boundary of requests and releases, it is possible to react on unforeseen events or delays. This allows, for example, to interrupt a request if no resource can be allocated within a reasonable amount of time or if a resource that is allocated experiences an error (**R6 Reactivity**).

The modelling pattern proposed provides full flexibility with regard to the mechanisms used to allocate resources. Resource allocations can be made manually, using simple decision rules or heuristics, or even powerful optimisation methods (**R7 Flexibility**).

At the time of writing this paper, we are extending the modeller and XML-schema in such a way that additional data required for process execution can be included. Also, we are developing a process execution engine capable of automatically allocating suitable resources to requests at run time. Furthermore, an approach for automatically translating such resource-aware business process models into a mixed integer programming formulation stating a novel resource-constrained scheduling problem is currently developed and, in the future, we want to develop algorithms for optimising resource allocations based on the BPMN 2.0 extensions presented in this paper.

References

1. W. Stevenson, *Operations management.* McGraw-Hill Higher Education, 2014.
2. S. Bhasin, *Lean management beyond manufacturing* Springer, 2015.
3. Y. Monden, *Toyota production system: an integrated approach to just-in-time.* Productivity Press, 2011.
4. Object Management Group, "Business Process Model and Notation (BPMN) 2.0.2," http://www.omg.org/spec/BPMN/2.0.2/PDF (last accessed May 6, 2022), 2013.
5. G. Havur, C. Cabanillas, J. Mendling, and A. Polleres, "Resource allocation with dependencies in business process management systems," in *International Conference on Business Process Management.* Springer, 2016, pp. 3–19.
6. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Ral: A high-level user-oriented resource assignment language for business processes," in *Business Process Management Workshops*, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Springer, 2012, pp. 50–61.
7. C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortés, "RALph: a graphical notation for resource assignments in business processes," in *International Conference on Advanced Information Systems Engineering.* Springer, 2015, pp. 53–68.
8. C. Cabanillas, M. Resinas, J. Mendling, and A. Ruiz-Cortés, "Automated team selection and compliance checking in business processes," in *Proceedings of the 2015 International Conference on Software and System Process*, ser. ICSSP 2015. Association for Computing Machinery, 2015, p. 42–51.
9. Workflow Management Coalition, "BPSim - Business Process Simulation Specification," Document Number WFMC-BPSWG-2012-1, 2013.
10. R. Laue and C. Mueller, "The business process simulation standard (BPSIM): Chances and limits," in *Proceedings 30th European Conference on Modelling and Simulation*, T. Claus, F. Herrmann, M. Manitz, and O. Rose, Eds., 2016.
11. B. S. S. Onggo, N. Proudlove, S. D'Ambrogio, A. Calabrese, S. Bisogno, and N. Levialdi Ghiron, "A BPMN extension to support discrete-event simulation for healthcare applications: an explicit representation of queues, attributes and data-driven decision points," *Journal of the Operational Research Society*, vol. 69, no. 5, pp. 788–802, 2018.
12. L. Pufahl and M. Weske, "Batch activities in process modeling and execution," in *International Conference on Service-Oriented Computing.* Springer, 2013, pp. 283–297.
13. G. . Gordon, "A general purpose systems simulation program," in *AFIPS '61: Proceedings of the Eastern Joint Computer Conference.* Association for Computing Machinery, 1961.
14. G. Wagner, "Business process modeling and simulation with DPMN: Resource-constrained activities," in *Proceedings of the 2020 Winter Simulation Conference*, K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, Eds., 2020.
15. L. Pufahl, S. Ihde, F. Stiehle, M. Weske, and I. Weber, "Automatic resource allocation in business processes: A systematic literature survey," *arXiv preprint arXiv:2107.07264*, 2021.
16. S. Ihde, L. Pufahl, M. B. Lin, A. Goel, and M. Weske, "Optimized resource allocations in business process models," in *Business Process Management Forum*, T. Hildebrandt, B. F. van Dongen, M. Röglinger, and J. Mendling, Eds. Springer, 2019, pp. 55–71.
17. S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European journal of operational research*, vol. 207, no. 1, pp. 1–14, 2010.
18. N. Russell, A. H. M. Ter Hofstede, W. M. P. van der Aalst, and D. Edmond, "Workflow resource patterns," BETA Working Paper Series, WP 127. Eindhoven University of Technology, 2004.