




A framework for modeling and executing task-specific resource allocations in business processes

Sven Ihde¹  · Luise Pufahl² · Maximilian Völker¹ · Asvin Goel³ · Mathias Weske¹

Received: 8 September 2021 / Accepted: 11 May 2022 / Published online: 17 June 2022
© The Author(s) 2022

Abstract

As resources are valuable assets, organizations have to decide which resources to allocate to business process tasks in a way that the process is executed not only effectively but also efficiently. Traditional role-based resource allocation leads to effective process executions, since each task is performed by a resource that has the required skills and competencies to do so. However, the resulting allocations are typically not as efficient as they could be, since optimization techniques have yet to find their way in traditional business process management scenarios. On the other hand, operations research provides a rich set of analytical methods for supporting problem-specific decisions on resource allocation. This paper provides a novel framework for creating transparency on existing tasks and resources, supporting individualized allocations for each activity in a process, and the possibility to integrate problem-specific analytical methods of the operations research domain. To validate the framework, the paper reports on the design and prototypical implementation of a software architecture, which extends a traditional process engine with a dedicated resource management component. This component allows us to define specific resource allocation problems at design time, and it also facilitates optimized resource allocation at run time. The framework is evaluated using a real-world parcel delivery process. The evaluation shows that the quality of the allocation results increase significantly with a technique from operations research in contrast to the traditional applied rule-based approach.

Keywords Process Execution · Business Process Management · Resource Allocation · Resource Management · Activity-oriented Optimization

Mathematics Subject Classification 68U35 · 90C29

✉ Sven Ihde
sven.ihde@hpi.de

¹ Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

² Software & Business Engineering, Technische Universität Berlin, Berlin, Germany

³ Kühne Logistics University, Hamburg, Germany

1 Introduction

Organizations run a great variety of business processes for the successful delivery of products or services to their customers, e.g., order handling, sales processes, or employment processes. For the execution of business processes, organizations need a rich set of resources consisting of human resources, machines, vehicles, materials, etc. [5, 15]. Resources have a high influence on the success and the efficiency of a single process execution, also called *process instance* [15]. The resources of an organization are valuable assets, often cost-intensive, and limited [2]. Thus, tasks (i.e., work needed to be executed for a certain instance) need to be allocated to an appropriate resource at the right time. Otherwise, it might lead to delays in the process execution, low quality, high costs, or low utilization of resources [4]. A *resource allocation* maps the demand of the running process instances with the availability of resources under a certain business goal [20] (e.g., minimizing the execution time).

Whereas the discipline of business process management (BPM) provides methods, techniques, and applications to manage organizational processes prosperously from the design, over the implementation, until the monitoring and analysis [9], the discipline of operations research (OR) provides a rich set of analytical methods and techniques for a problem-specific resource allocation [24]. Current BPM systems (BPMSs) used to support the execution of business processes (based on a given process model) [9, 35] still mainly work with rule-based resource allocation mechanisms (e.g., the FIFO [First-In-First-Out] mechanism) [5, 28]—traditional approaches, which need less computational power, but may result in suboptimal solutions [12]. With the progress of technology and availability of higher computing power, the possibility exists to integrate more sophisticated analytical techniques into BPMSs to support realistic business scenarios. This integration of sophisticated resource allocation approaches in business processes was also tackled by other recent research works, e.g. [2, 13, 16, 18] with different optimization goals (e.g., minimizing costs) and solution techniques (e.g., linear programming). However, no generalized approach exists to integrate resource allocation techniques flexible in a multitude of business processes in an organization based on the demands of a certain process activity.

In this research work, we provide a framework and software design to support the integration of optimized resource allocation in BPMSs. The framework aims at:

- Creating transparency on existing tasks and resources, and their characteristics in an organization,
- Supporting a resource allocation that can be individualized for each activity, catering for the goals and environment of said activities, and
- Integrating problem-specific analytical methods and techniques for specific business process activities, which can be flexibly adapted at design time.

Second to the framework, we provide a software architecture realizing the framework. It integrates a BPMS with a so-called *resource manager*—a new component as sketched in our previous work [17]—that enhances existing business process execution environments to a resource-aware execution. The resource manager serves to describe available resources and resource types, and allows the definition of resource allocation problems, for which problem-specific analytical techniques can be integrated. The

software architecture is prototypically implemented and tested in a real-world setting. In contrast to our previous work in [17], we provide in this research work a general formal framework for resource allocation in business processes, a more detailed presentation of the software architecture and its prototype, and a case study from logistics. With the case study, we want to show the feasibility to integrate different allocation techniques for a process activity in the resource manager and exemplify that a sophisticated technique from OR can lead to qualitatively higher results than the rule-based techniques, which are traditionally applied in existing BPMSs.

After this introduction, existing works on resource allocation in business processes are discussed in Sect. 2, followed by motivating examples in Sect. 3. In Sect. 4, relevant concepts of business processes, resources, and the problem of allocating resources to process instances are introduced. After presenting the framework for resource allocation in business processes in Sect. 5, the software architecture for sophisticated resource allocations in a BPMS and a corresponding prototype are described in Sect. 6. A case study evaluating its effect and implications is presented in Sect. 7, and limitations and future work are discussed in Sect. 8.

2 Related work

The resource perspective is in addition to the control- and data-flow perspective in process models essential for the successful execution of business processes [9]. Cabanillas [5] distinguishes among three key operations of resource management in business processes: (1) *resource allocation configuration*¹ (i.e., definition of resource requirements for process activities at design time), (2) *resource allocation* (i.e., designation of concrete resources to a specific task during run time), and (3) *resource analysis* (i.e., run-time and post-execution analysis of the process executions with focus on resources).

A great variety of approaches have been developed to automatically support the resource allocation in business processes as presented in the structured literature review in [27]. They differ in their resource allocation capabilities, goals, whether they support global or only local optimizations, and their solution technique. In the following, we present a selective set of approaches to illustrate the different capabilities of existing approaches summarized in Table 1. For the complete set of approaches, we refer the interested reader to related literature reviews on this topic [2, 27].

Resource allocation approaches for business processes often support 1-to-1 allocations, where one task is allocated to exactly one resource, such as [1, 6, 8, 11, 14, 22, 31]. Some works also provide approaches for 1-to-many allocations, e.g. [10, 21, 36], where the capacity of a resource is greater one, and many-to-1 allocations [7], where a team or a group of resources solves a task. A minority of approaches even support many-to-many allocations where a set of tasks can be allocated to a set of resources, for instance supported by [34].

¹ Originally called *resource assignment* by Cabanillas [5]. In operations management, resource assignment is the mapping of resources to tasks. Thus, in this research work, we use the term *resource allocation configuration*.

Table 1 Related studies and their categorizations ordered by publication year

Reference	Year	Capability	Goal	Type	Technique
Van Hee et al. [34]	2001	many-to-many	Minimize cycle time	local	Linear programming
Doerner et al. [8]	2006	1-to-1	Minimize process cost	global	Heuristic
Ha et al. [11]	2006	1-to-1	Balance workload	global	Rule
Rhee et al. [29]	2008	many-to-1	Balance workload	global	Heuristic
Xu et al. [36]	2009	1-to-many	Minimize process cost	global	Rule
Huang et al. [14]	2010	1-to-1	Any	local	ML
Liu et al. [22]	2012	1-to-1	Find best-fitting resource	global	Trained rule
Caballias et al. [6]	2013	1-to-1	Find best-fitting resource	local	Heuristic
Kumar et al. [21]	2013	1-to-many	Minimize cycle time	global	Heuristic
Pflug et al. [25]	2016	many-to-1	Maximize throughput	global	ML
Schoenig et al. [31]	2016	1-to-1	Minimize cycle time	global	Rule
Xu et al. [37]	2016	many-to-1	Balance workload	global	ML
Yaghoubi et al. [38]	2017	many-to-1	Balance workload	global	Rule
Arias et al. [1]	2018	1-to-1	Find best-fitting resource	global	Linear programming
Djedovic et al. [7]	2018	1-to-many	Minimize process cost	global	ML
Duran et al. [10]	2019	1-to-many	Any	global	Heuristic

Resource allocation is an optimization problem where a certain goal is targeted and constraints are considered. Many suggested approaches for resource allocation in business processes follow a specific goal. On the one hand, process-oriented goals are supported, such as finding the best-fitting resource for a task [1, 6, 21, 22], minimizing the cycle time [31, 34], or minimizing the process costs [7, 8, 36]. On the other hand, resource-oriented goals are targeted, such as balancing the workload [11, 29, 37, 38]. Two of the selected approaches provide the option that the optimization goal can be individually defined by a user when applying and implementing the approach: Huang et al. [15] maximize the allocation reward, and the calculation of the reward can be specified, whereas Duran et al. [10] describe a multi-objective optimization problem that needs to be defined.

The execution of several business processes in an organization results in several tasks which need to be allocated to resources. Each task has a different relevance for the organization, which needs to be considered in the resource allocation. Several approaches support a *global* optimization for allocating resources to business cases, also considering different importance of tasks (see column *Type* in Table 1). Some approaches support *local* optimization, where the best resource is selected for a specific task without considering other available tasks, with the risk of arriving at a suboptimal global solution. As solution techniques for the resource allocation problem in business processes, rules, heuristics, machine learning (ML), and linear programming have been applied. In general, the advantage of rule-based approaches is that solutions can be delivered within a short time frame in the average case, but the solution quality is in comparison to the other techniques not so high [12]. The other extreme is linear programming, where a solution might not be found in the requested time, but the quality of the solution is high. Heuristics are individual problem-solution techniques which aim for a solution with a good quality that can be found in a short time frame.

Based on this review, we can observe that existing approaches usually tend to support one specific optimization goal and apply one specific allocation technique to all process activities. However, it might be useful to apply specific allocation techniques for specific tasks to reach an overall good performance. OR provides a rich set of analytical techniques, which are often problem-specific and especially useful when they are applied to specific process activities. In contrast to existing work, we want to provide an activity-oriented resource allocation approach reusing sophisticated analytical techniques from OR, which is integrated in a BPMS and can be directly defined and executed there. Thereby, we want to allow that the process designer can flexible select the optimization goal, the allocation capability and type, and the solution technique.

3 Motivation

This section is used to motivate the need for sophisticated resource allocation in business processes on the example of (1) an incident resolution, and (2) a parcel delivery process. The following examples are simplified to explain only the necessary core concepts of resource allocation in business processes to motivate our approach.



Fig. 1 BPMN diagram of the incident resolution process

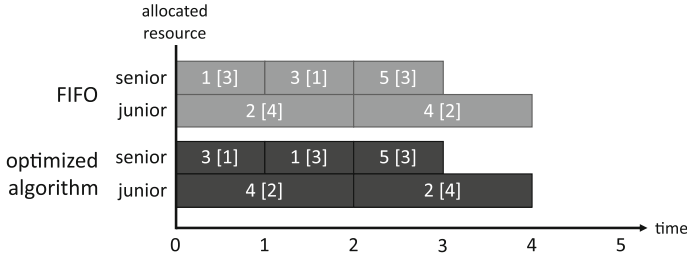


Fig. 2 Difference between resource allocation strategies

3.1 Incident resolution

In Fig. 1, a simplified incident resolution process is given as BPMN process diagram. After an incident is received it will be classified due to its severity into the classification levels 1 to 4. An incident with classification 1 is to be treated with the highest priority, meaning it is supposed to be done within a time unit of 1, whereas an incident with classification 4 can be finished within four time units. Thus, the efficiency measure is a quality measure that is increased, each time a deadline is not met. Therefore, an optimized resource allocation will minimize this value, so that every task is done within their deadlines. During the execution of this process, a human resource needs to be allocated to resolve an incident. For simplicity, we assume that the process has only two resources available - a senior (solving an incident in one time unit) and a junior service expert (needs two time units). Traditionally, a simple scheduling strategy (e.g., FIFO), like for the “Classify incident” task, is often used in BPMSs. An example of such an allocation is shown in the upper part of the Fig. 2. At time unit 0, five process instances (Incident ID 1 to 5) are created and classified and have to be resolved in the next step—meaning the task “Resolve incident” is enabled for all instances. In the notation, the number in the brackets describes the classification of the incident, which is the deadline until the task should be finished.

This example shows that a resource allocation for all incidents can be realized. However, this simple allocation entirely ignores the classification—the priority—of the incidents, which leads to inefficient process executions as the due dates were not fulfilled. Therefore, a more sophisticated allocation technique should be applied in this specific scenario that optimizes the allocation by considering the urgency of a task. The resulting allocation is depicted in the lower part of Fig. 2. The overall execution time is the same, but the second allocation is more efficient, as all deadlines could be reached.

Based on this example, we can observe that depending on the individual task, the use of a different resource allocation logic is recommended. Moreover, because of the



Fig. 3 Process model of parcel deliveries with promised delivery time windows

limited availability of resources, we need to dynamically, based on the current tasks and resource, plan the most efficient resource allocation. To achieve that, we need to be able to define different resource allocation problems, the process and task information, as well as resources needed.

3.2 Parcel delivery

We additionally consider the case of resource allocation with shared resources, meaning we have a 1-to-many resource allocation between resources and tasks. In Fig. 3, a common use case in the field of logistics of a parcel delivery is shown as BPMN process diagram. The process starts when the information is received that the parcel has arrived in a pick-up place (e.g., discounter, fitness studio, etc.). From this place, the parcel should be delivered during a preferred time frame to the recipient by a local carrier. Thus, first of all, the recipient's preferences are collected. Then, the parcel is allocated to a vehicle of a local carrier, for which the tour should be optimized, meaning the efficiency measure is the length of the tour (= to the process execution time). The carrier is then delivering the parcel according to the receiver's preferences.

If we had all information about vehicles and parcels, the resulting problem would be a vehicle routing problem [33], which is inherently NP-hard. Moreover, in many cases we do not have all the necessary information, as parcels arrive sequentially over time. If requests for parcel deliveries arrive dynamically, we need to decide on an allocation of a parcel to a vehicle based on the limited information available at the time of making decisions. Traditionally, this is done manually or by BPMS's looking at one parcel at a time and finding the best possible vehicle based on a rule-based approach. The tours resulting from such a sequential decision-making approach may be inefficient. In the OR field, research has shown that using more sophisticated algorithms can improve the result [3]. In contrast to a sequential decision-making approach that only considers one task at a time, the algorithms would look at multiple tasks (parcels) concurrently. With the framework presented in this paper, we facilitate an integration of sophisticated algorithms in the automated execution of business processes in order to increase the efficiency of the processes.

4 Foundation

In this section, relevant existing concepts for resource allocation in the environment of business processes are formally defined. First, basic concepts of an organization, such as business process, task, resource, and resource allocations are defined. Then,

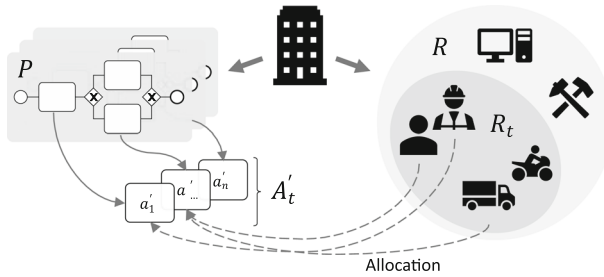


Fig. 4 Organizations having business processes and their resources connected by the tasks resulting from process executions, which need to be allocated to resources

these definitions are used to define a novel definition of a stateful orchestration that combines these aspects in a global model.

Organizations are running a set of business processes for serving their clients. Often these business processes are formalized in the form of business process models [35] P as shown in Fig. 4. A business process model is defined as follows:

Definition 1 (*Business Process Model*) A business process model is a graphical representation of a process consisting of a set of nodes (A, G, E, D). These nodes are either activities A , gateways G , events E , or data objects D . These nodes are connected to each other via control flow edges or data flow edges. A business process model acts as a blueprint for a set of business process instances. A process instance represents one concrete execution of the process model. \square

An activity requires time and resources to be performed. Similarly to the process instances, activity instances are created during run time [35]. In this work, we use the term *task*² (i.e., activity instance) defined as follows:

Definition 2 (*Task*) A task a' describes exactly one concrete execution of a predefined activity $a \in A$. Each task of an activity a adheres to the same life cycle during run time. It exists only during run time and is terminated when it reaches the final state of the life cycle. During run time, a task has attributes and can access the process instance attributes and related data objects, which is done via the function $attr(a')$. Additionally, we define the set of tasks of all activities of all process instances as \mathcal{A}' . \square

A task of a process instance has to be allocated to at least one resource for its execution, such that the business goal can be reached. Its life cycle during execution is given as a state transition diagram in Fig. 5 (based on the life cycle given in [35] and extended to the aspect of resource allocation).

As soon as a process instance is started, its tasks are initialized. During the process execution, a task in state *init* is either enabled (for example by a control flow) and then in state *ready*, or skipped (e.g., optional path due to process decisions) and in state

² Please note that this definition of a *task* is different to the BPMN standard [23], in which it is defined as single unit of work in a process diagram, whereas we define a *task* as an activity instance of a process instance.

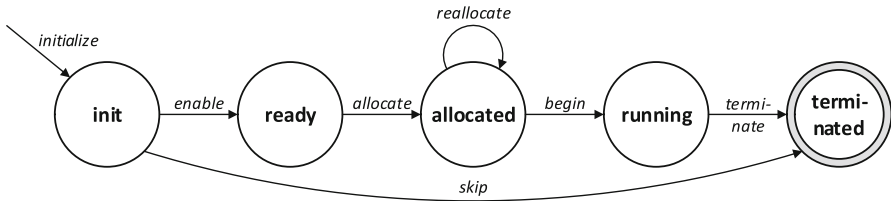


Fig. 5 The life cycle of a task that is executed in an organization

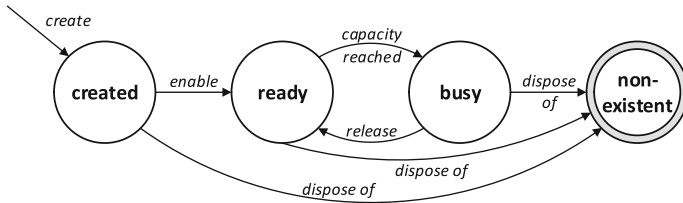


Fig. 6 The life cycle of a resource in an organization

terminated. If a task is ready, it has to wait until resources are allocated, before it can begin its execution. During the allocation state, it can be reallocated. At some point, the resources begin with executing the task (running state), if the task is finished, it is terminated.

Next, we define a resource in the context of resource allocation:

Definition 3 (Resource) Let R be the set of resources in an organization. Then $r \in R$ is a resource (e.g., human, vehicle, software, etc.), that is able to execute tasks. A resource has a set of attributes, which change during run time. Additionally, a resource has a current life cycle state, which is changed depending on the attribute values. □

Additional to attributes describing the capabilities and capacities of a resource, a resource is also in a certain state as shown in Fig. 6.

First, a resource needs to be created in an organization, e.g. by hiring new personnel. A resource is available, if the state of the resource is ready, meaning it has capacity left to handle an additional task. Although some tasks might be already allocated to it, it is possible to allocate more tasks to this resource as long as its maximum capacity is not reached. When it is reached, the resource is in the state busy. Naturally, the resource is able to return to the ready state as soon as it has capacity left for new allocations. At all states, a resource can be disposed of, transferring it to the state non-existent. This state means that from the organization’s point of view, this resource can no longer execute tasks³.

In the next step, we now formalize the intersection of those two worlds, tasks and resources—the resource allocation, which can be defined as follows:

³ One assumption here is that a temporary drop out, for example in the case of a vacation or illness of a human resource, is not treated as a disposal. Instead, the maximum workload of that resource will be set to 0 and already allocated tasks to this resource will be considered for reallocation.

Definition 4 (*Resource Allocation*) A resource allocation is the single decision of which concrete resource(s) are going to execute which concrete task(s) at a point in time t . In the resource allocation decision, only the resources R_t , which are available in the organization at that point in time t as well as all tasks \mathcal{A}'_t that are *ready* or *allocated* need to be considered. Additionally, there exists a *goal* (e.g. an efficiency measure like minimize the cost, maximize throughput, etc.) that the resource allocation should optimize for in the context of the waiting tasks as well as constraints (e.g., maximum number of parcels on a tour) that need to be upheld. A concrete implementation (e.g. algorithm) is called allocation mechanism (*alloc*). \square

Additionally, we need to consider the environment to optimize the allocation, e.g., other resource allocations (e.g., when predicting the future), attributes of the tasks, prioritization, or constraints. If we look at the parcel delivery example, we can observe this phenomenon quite easily. Here we have to match multiple tasks, each representing one parcel to be delivered, to one resource, the vehicle. The address of the parcel influences the distance the vehicle has to travel and thus the cost of the resource allocation. In order to enable a meaningful global resource optimization, we need to plan on an organizational level. Therefore, we propose the following state transition system of a stateful orchestration to frame the organizational level:

Definition 5 (*Stateful Orchestration*) Let \mathcal{A}'_t be the set of tasks that are in the state *ready* or need to be reallocated, and R_t be the set of resources *ready* in the organization at a point in time t . Then the state S_t of an organization at time t can be defined as follows:

$$S_t = (\mathcal{A}'_t, R_t)$$

If the point in time changes ($t \rightarrow t'$), the state of an organization may be changed, too. Thus, we define three occurrences that can change the state:

1. A task gets added to or deleted from the set of \mathcal{A}'_t . This can happen due to a control flow that enables an activity instance. The state of the organization is changing: $S_t = (\mathcal{A}'_t, R_t) \rightarrow S_{t'} = (\mathcal{A}'_{t'}, R_t)$, whereby only the set \mathcal{A}'_t changes and R_t is unchanged.
2. A resource gets added to or deleted from the set of *ready* resources. This can happen by releasing a resource, reaching the capacity of the resource, or when a resource is disposed of. The state of the organization is changing: $S_t = (\mathcal{A}'_t, R_t) \rightarrow S_{t'} = (\mathcal{A}'_t, R_{t'})$, whereby only the set R_t changes and \mathcal{A}'_t is unchanged.
3. A resource allocation was performed, whereby both sets of an organization are changing: $S_t = (\mathcal{A}'_t, R_t) \rightarrow S_{t'} = (\mathcal{A}'_{t'}, R_{t'})$.

\square

This model describes an abstraction of an organization with the focus on its tasks waiting for execution and its available resources as well as their changes over time. At run time, new business cases are created within an organization, initiating the execution of new process instances. Therefore, the set of tasks of an organization, which need to be executed, is constantly changing. Similar to tasks, resources can get added to

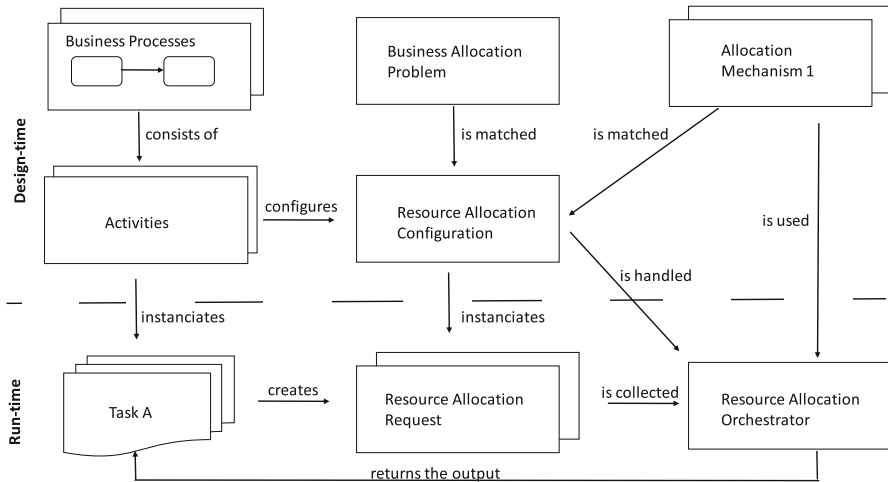


Fig. 7 Overview of the flow and dependencies of the framework, split into design-time and run-time components

and deleted from the set of available resources (i.e., in state *ready*). To keep track of the changes of resources, the second state change of stateful orchestrations, we also assume that the information (e.g., capabilities, workload, working times, etc.) on them is available. Finally, resource allocations may change the set of waiting tasks and available resources by assigning a resource to a task. This way we can capture every necessary information needed on the global level for resource allocation, considering tasks and resources at the same time.

5 Framework for resource allocation in business processes

In this section, we introduce a general framework for deciding which resources to allocate to which business process tasks (cf. Fig. 7). In the sections before, we discussed that having a more sophisticated approach, in contrast to the traditional resource allocation, has a positive effect on the overall efficiency. Next, we want to focus on the essential elements for conducting resource allocation in a stateful orchestration.

Based on the definitions provided in Sect. 4, four key components for resource allocation in business processes are necessary as shown in Fig. 7; namely, the *business allocation problem*, *resource allocation configuration*, *resource allocation request*, and the *resource allocation orchestrator*. The *business allocation problem* formalizes the resource allocation problem observed in business processes. This definition is then used at *design time* to configure the resource allocation of one or several process activities. The so-called *resource allocation configuration* enriches a process activity with resource allocation knowledge by specifying the business allocation problem and mapping the process information to the resource allocation. At *run time*, a task then creates a *resource allocation request* consisting of the process information available to the task as well as the chosen resource allocation configuration. This request is then

received by the respective *resource allocation orchestrator* with a queue for incoming requests as well as the respective solution technique for solving the optimization and fulfilling the optimization goal. After giving a small intuition of the different concepts, they are defined in more detail in the following.

Our framework suggests to first create a business allocation problem for the resource allocation in business processes that describes the input and output information required. This can be reused for one or several activities in different business processes. It is defined formally as follows:

Definition 6 (*Business Allocation Problem*) A business allocation problem describes the concrete resource allocation needed in a business process to be able to execute the task. The business allocation problem is therefore a tuple (I, O) , where

- I describes the expected input information of tasks, e.g., the type of the tasks, the process information, data attributes, etc.,
- O describes the expected outcome or the blueprint of the solution, which is the type of resource(s) allocated to the corresponding input tasks as well as their attributes.

□

In the case of our given parcel delivery example, the business allocation problem would be the parcels that need to be planned on a tour. The input I is the information about the parcel (address) and their receiver (time window). The output O is the tour, consisting of the vehicle, the order of the parcels as well as the start time.

At design time, the process designer links each activity to a business allocation problem. Additionally, the designer decides on the concrete goal of the optimization as well as how the information of the business process (e.g. process data, data object attributes) is transformed to match a business allocation problem, and how the solution is transformed back. This is summarized as the so-called *resource allocation configuration*, which is defined as follows:

Definition 7 (*Resource Allocation Configuration*) A resource allocation configuration is a tuple $(bap, goal, alloc, g, h)$, where

- bap is a business allocation problem
- $goal$ is a decision on what the bap should be optimized for (e.g. minimizing the cost, cycle time, etc.)
- $alloc$ is a concrete implementation matching the bap and $goal$, (a concrete algorithm that solves the bap by also optimizing for the $goal$),
- g is a function $g : attr(a') \rightarrow I$ that transforms the information available at run time by the task a' , into the input I of the bap .
- h is a function $h : O \rightarrow attr(a')$ that transforms a solution that is returned as the output O of the bap into available data attributes of the task a' .

□

To keep the configuration in a manageable frame, we propose to use a default resource allocation configuration that falls back to traditional allocation strategies used by BPMSs for activities where a more sophisticated approach is not necessary.

During run time, processes get instantiated and create new tasks that need to be executed. Based on the resource configuration of its related activity, the task creates a *resource allocation request* at run time in order to start the resource allocation, which is defined as follows:

Definition 8 (*Resource Allocation Request*) Each time a new task a' is enabled or needs a reallocation, it creates a corresponding resource allocation request. The resource allocation request consists of the resource allocation configuration, as well as the values for the input I needed of the *bap*. \square

As we want to optimize the decision on what resources handle which task, we want to make sure that multiple requests can be handled at the same time. This means, just having the algorithms to solve the allocation problems is not enough as we additionally need to have a scheduling in place that executes these algorithms at the right time and order. This way we can enable prioritization of different activity instances as well as the use case of shared resources. Another point of notice is that even though we can define goals and constraints for a resource allocation problem, they are not constant and change depending on the current situation. For example, the cost $c_{a'r}$ of an allocation of the task a' and resource r is dependent on which other task a'' can potentially use the same resource. If a'' is from a higher prioritized process (not necessarily the same process model), the cost $c_{a'r}$ will increase, so that a'' will be chosen more likely by the allocation solver. This means that a resource allocation cannot be limited to only one process model, but need to take care of tasks of all process models working with the same resources. The framework allows for cross-process resource allocation by building resource allocation orchestrators that handle resource allocation requests with resources and tasks of similar nature. We defined this approach as follows:

Definition 9 (*Resource Allocation Orchestrator*) A resource allocation orchestrator is created for a set of resource allocation configurations. The orchestrator consists of a queue for collecting resource allocation requests during run time, as well as a scheduling logic that decides when the resource allocation is executed based on the current information (S_t). The temporary results O are then either further optimized or returned to the process. Every time an orchestrator decides on a resource allocation, the corresponding tasks and resources get updated. This means the stateful orchestration gets updated to a new state $S_t \rightarrow S_{t'}$. \square

In the parcel delivery example, this would mean that the resource allocation orchestrator is creating tours based on the current information (parcels, couriers, traffic, etc.). However, not every new request (created by a new parcel in the system) triggers the execution of the solution algorithm. The orchestrator is able to wait, for example until a sufficient number of parcels have been collected or when a certain time has passed, before the solving of the problem is started. Moreover, the resulting solution is not immediately returned to the process, but can be further refined in the future, when new information gets available.

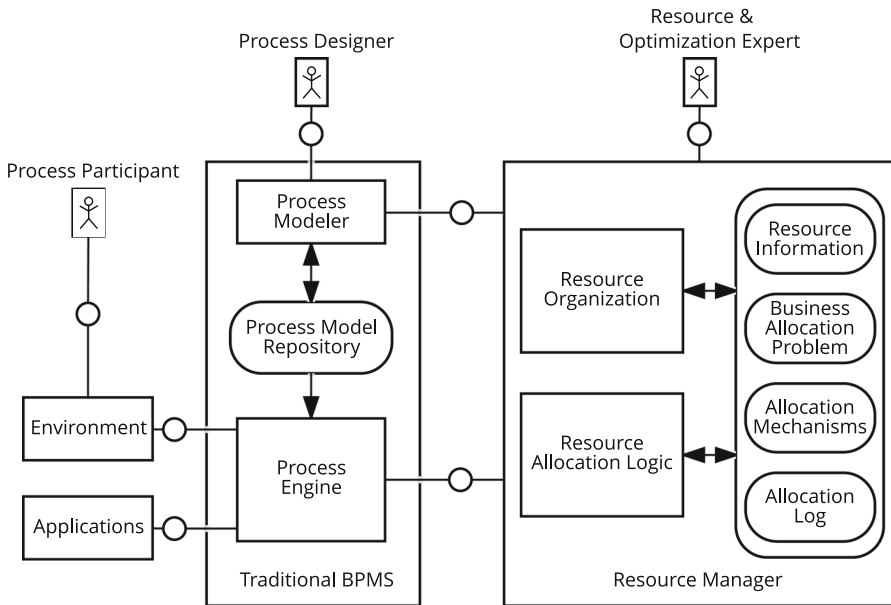


Fig. 8 Architecture of a resource-aware BPMS for smart resource allocation shown as FMC diagram, which represents active software components as rectangles and storage as ellipses

6 Architecture of a resource-aware BPMS

In this section, we want to present a software architecture to realize the above presented framework for resource allocation in business processes.

Traditionally, the execution of business processes is supported by so-called BPMSs [35]. Such a BPMS usually consists of a *Process Modeler*, where business process models can be designed by a process designer and stored in a *Process Model Repository*, and a *Process Engine*, where the modelled processes can be executed in interaction with the *Environment* and *Applications* (cf. Fig. 8). A *Process Engine* usually supports simple resource allocation rules (e.g. role-based distribution or shortest queue) [30]. The main idea of the resource-aware BPMS is to decouple the resource allocation functions from the *Process Engine* and to bundle them into an own component, the so-called *Resource Manager* as shown in Fig. 8. The resource manager is designed as a component with the centralized, transparent knowledge on resources and their attributes, and the possibility to design, integrate, and call different allocation services. This decoupling also promotes a separation of concerns, as processes can continue to be designed by the process designer, while the resource-related knowledge required by the resource manager can be contributed by another role, the resource and optimization expert.

As shown in Fig. 8, the *Resource Manager*, that extends the traditional BPMS consists by two main components: the *Resource Organization* and *Resource Allocation Logic*. The resource organization component is responsible for managing resource-related information, like attributes and meta-information, including, for example, the

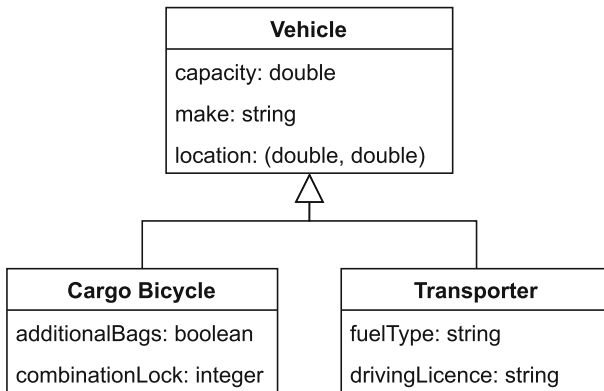


Fig. 9 Hierarchy of resource types with inheritance

costs of resources. Requests for resource allocations are handled by the resource allocation logic component, which invokes the appropriate resource allocation solver and returns the resource(s) determined by the service. Both main components are briefly explained in the subsections.

6.1 Resource organization

In this component, information about resources, their characteristics, and their current state can be managed. To be able to reflect the diversity of the resources that exist in a company, and to model their similarities and differences, the notion of *Resource Types* is introduced. Resource Types describe the structure of resources of a certain kind, which includes their set of attributes, like a *Cargo Bicycle* always having a specific *capacity*. Considering the different resources within a company, their sets of attributes might overlap: For example, a *Transporter*, also some kind of freight vehicle, has a capacity, too. Additionally, it might also have a *fuel type* or a required *driving licence*, attributes that are not applicable to bicycles. So, although transporters have some attributes in common with cargo bicycles, they are not of the same resource type.

To be able to model such commonalities, as well as specializations, the resource types management of the platform is built on the concept of hierarchies: Using a tree-like structure, resource types can inherit the set of attributes of another type, as illustrated in Fig. 9. By this, the shared attributes of cargo bicycles and transporters can be defined in a super-ordinate *vehicle* resource type. Both can now inherit the characteristics of a vehicle, while adding their own, type-specific attributes.

In addition to resource types, the component is also responsible for managing the digital representations of the resources available in the organization, which are also referred to as *resources*. New resources can be created by using a resource type definition as a blueprint and providing the resource-specific attributes. Furthermore, resources can be modified by setting new attribute values, or deleted if a resource becomes permanently unavailable.

6.2 Resource allocation logic

Based on the resource types and resources defined in the organization component, the *Resource Allocation Logic* component enables the optimized allocation of resources to tasks of process instances as introduced in Sect. 5. For this, apart from the resource definition, two steps are necessary: the specification of the resource allocation at design time and the calculation of the allocation during run time.

6.2.1 At design time—configuration

At design time, the means of *how* to find the best fitting resource to be allocated for the given task must be defined by specifying the so-called business allocation problem (cf. Definition 6). Here, the *constraints* and the *goal* of the allocation play an important role and have to be incorporated (cf. Definition 4).

Initially, the types of resources that are relevant for the problem have to be specified by the resource and optimization expert. For this, the resource types needed as input and output, i.e. the type(s) of the allocated resource(s), are differentiated: For example, a vehicle for a list of parcels should be allocated. Therefore, at design time, this problem is defined to take a list of resource of the type *parcel*, as well as all available resources of the type *vehicle* as input (which also includes all sub-types) and to return a vehicle resource as output. The output of an allocation does not necessarily have to be an already existing resource, as long as the corresponding resource type already exists. For example, when a tour should be allocated to a given set of parcels and vehicles, the tour object itself does not exist beforehand, but is created during the allocation process based on the input resources and returned as a new, allocated resource.

Between the input and the output, the mechanisms of actually finding the most appropriate resource to allocate based on the inputs can be manifold: there are, for example, rule-based, history-based or heuristic-based approaches. To cater for this variety, the way of defining the resource allocation mechanisms should be flexible and adaptable within the resource allocation platform.

For simple allocations, like finding the right packaging for a parcel, a simple rule-based allocation might suffice: rules covering possible dimensions and weights of parcels and assigning the appropriate box can easily be defined. However, when it comes to planning routes for delivering parcels or even assigning a suitable driver to it, rules are not adequate. In this case, a heuristic for route-planning or even an online-service could be useful, whereas assigning the driver might benefit from historic data, e.g. by analysing former routes in the same area and the respective drivers. Furthermore, a combination of these mechanisms is also conceivable, since it is not sufficient to consider only the historical allocations, but also only the currently available drivers.

In summary, a problem definition consists of the input resource types and the type(s) of the resource(s) that are returned and therefore allocated. The specifications of how to find the best resource for the given problem from the input in regard to a specific goal are stored as *Allocation Mechanisms* in the resource-aware extension of the BPMS as shown in Fig. 8.

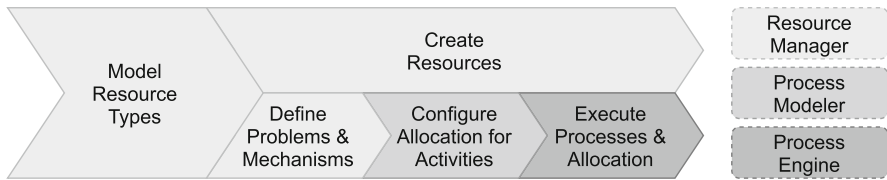


Fig. 10 Workflow when working with the resource manager

6.2.2 At run time—resource allocation

At run time, the actual resource allocation takes place, meaning that a concrete resource, or multiple resources, is/are allocated to the requesting activity based on a received resource allocation request (cf. Definition 8). The request is passed to the corresponding resource allocation orchestrator (Definition 9) in the Resource Allocation Logic component, which invokes the desired resource allocation configuration with the required input parameters, e.g. starts the tour allocation by executing the corresponding allocation mechanism with the current set of parcels.

To keep track of resources that are currently in use and therefore probably unavailable, the results of allocations are stored in a log within the resource-aware extension (cf. *Allocation Log* in Fig. 8). This also enables history-based approaches and performance analysis.

6.2.3 Connection to BPMS

So far, only the internal parts of the resource manager have been described. But also the interplay of this resource-aware extension and the BPMS is important and is enabled by a documented interface that allows existing BPMS to connect to the resource-aware extension and to use its functionalities described above.

The sequence of steps necessary for achieving an organization-wide optimized resource allocation for tasks is pictured in Fig. 10.

First, the resource types available in the organization have to be modeled in the resource manager. From now on, concrete resource representations, i.e. instances of the modeled resource types, can be created at any time. Based on the knowledge of the available resource types, the required problem definitions and associated allocation mechanisms can be defined. As soon as these are created, the activities of the processes in the BPMS can be connected to the resource-aware extension by specifying the resource allocation configuration (cf. Definition 7), which connects the activity to a problem definition and a concrete allocation mechanism.

Now, the processes are ready for execution. As soon as a task is enabled, the BPMS sends a resource allocation request to the resource manager based on the allocation configuration specified for this activity and subsequently receives the information about the allocated resource(s) from it. After the allocated resource has completed the task, the process engine gives feedback to the extension in order to release the resource from the allocation.

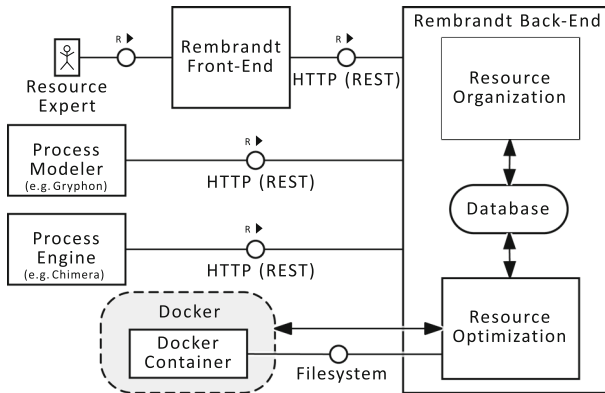


Fig. 11 The concrete prototypical implementation, Rembrandt, of the general Architecture of Fig. 8

6.3 Implementation

The described architecture of a resource-aware BPMS was used as foundation for a prototypical implementation called *Rembrandt*⁴. The implementation, based on TypeScript and MongoDB for data storage, is available under the MIT-License. As proposed, Rembrandt consists of two major parts: the *Resource Organization* and *Resource Optimization* components, which are briefly described below. It runs separately from any BPMS, but integrates into existing approaches by offering a documented application programming interface (API) based on REST/HTTP. For demonstration purposes, we have integrated Rembrandt with the process engine Chimera⁵ and the process modeler Gryphon⁶.

As shown in the FMC model in Fig. 11, Rembrandt also includes a front-end, which enables users to accomplish nearly the same tasks as provided by the API, e.g. creating resource types and resources, or adding resource allocation definitions.

Resource Organization The implementation of the resource organization allows, as described previously, for the creation of resource types within a hierarchy. For each new resource type, a *name* and a *parent resource type* can be defined, where all attributes of the specified parent type are then inherited.

Resource Optimization At design time, the process of finding the optimal resource to allocate must be defined. In Rembrandt, this is done using so-called *ingredients*, which are combined into *recipes*, which correspond to the concept of problem definitions in conjunction with a concrete allocation mechanism. A recipe defines the steps (ingredients) and their sequence that are necessary to determine the optimal resource(s) as illustrated in a screenshot of Rembrandt in Fig. 12. Initially, the required ingredients must be set up, whereby a distinction is made between the following types of ingredients:

⁴ <https://github.com/bptlab/rembrandt>.

⁵ <https://bptlab.github.io/chimera>.

⁶ <https://bptlab.github.io/gryphon>.

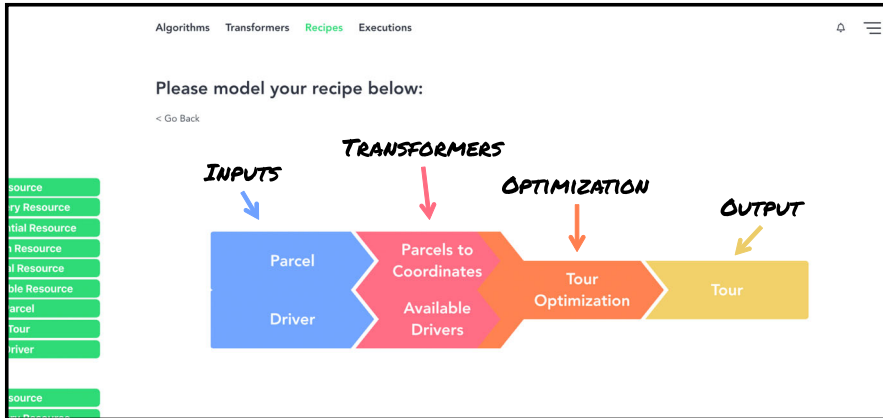


Fig. 12 Annotated screenshot of Rembrandt illustrating the recipe definition for defining an allocation mechanism in the *Resource Optimization* component

- The **input** ingredient represents the list of resources of a certain resource type, e.g., a set of *Parcels* and *Drivers*.
- **Transformer** ingredients can be used to manipulate a list of resources. This includes *filtering*, *combining*, and *modifying* the resources in the list. For instance, in Fig. 12 two transformers are used: a filter to select available, unassigned drivers only, and a modification to convert the addresses of the parcels into coordinates.
- The **optimization** ingredient includes the most important part of the allocation: the logic determining the best resource based on the provided list of resources, e.g. the *Tour Optimization* algorithm. This ingredient reflects the concept of the allocation mechanism.
- Like input ingredients, **output** ingredients are auto-generated for each resource type and are used to indicate the resource types returned by the allocation. For instance, the result of the *Tour Optimization* is a resource of the type *Tour*.

As motivated in Sect. 6.2.1, the allocation mechanisms can take on various forms. To support this diversity, Rembrandt's optimization ingredients are based on Docker containers⁷ as shown in Fig. 11. This has several advantages: Optimized allocation algorithms can be reused and shared between different Rembrandt instances. Additionally, the implementation of the algorithm is mostly independent regarding platform and programming language, and existing algorithms, e.g. from the field of OR, can be used in Rembrandt.

After all required ingredients have been created, they can be plugged together to create recipes for resource allocation. If a resource allocation is requested at run time, the respective recipe is executed by Rembrandt and the result of the execution is then returned to the BPMS.

⁷ <https://www.docker.com>.

7 Case study

In this section, we use the resource-aware BPMS to realize the parcel delivery example presented in Sect. 3.2 for evaluating the feasibility, flexibility of our proposed architecture, and the potential to result in qualitative higher allocation solutions. For this evaluation, we compare the outcome of a traditional rule-based optimization approach, as being used by current BPMSs, with an heuristic-based approach favoring a global optimization, originating from OR. Furthermore, the necessary steps in the different phases introduced above are explained. In the following, we review the parcel delivery scenario including assumptions and constraints, then we explain the setup of the experiments, the results, and finally discuss the outcome.

Evaluation scenario As discussed previously, the problem of the last mile delivery of parcels is studied in the SMile project [26]. This research project intends to innovate the last mile delivery of parcels with the help of a dense pick-up place infrastructure. From a pick-up place (e.g., a fitness studio, a gas station, a small grocery store), the parcel can be collected by a small carrier (e.g., a bike carrier), who brings the parcel to the receiver in a preferred time slot. To achieve a reasonable price for the delivery in preferred time slots, several parcels need to be scheduled on a tour and assigned to a carrier. For this scenario, a process instance is created for each parcel, which should be delivered to the receiver in a certain time frame. The selected scenario considers one area of delivery in a city in Germany similar to the research project. We assume three different available carriers for this area, each of which can handle the same number of parcels at a time. The carriers get paid an hourly wage and, as we assume each driver travels at the same speed, the length of the tour influences the price. Therefore, the goal of the optimization is to minimize the cost per parcel by reducing the distance travelled.

Design time As described in the previous chapters, at design time, the process, the resources, and the business allocation problem with appropriate configurations must be defined.

The process model as given in Fig. 3 is deployed at *Chimera*. The required resource types, i.e., the schemas for parcels, drivers, receivers, as well as for tours are configured in the prototypical setup as described in Sect. 6.3—the resource-aware extension *Rembrandt*.

Additionally, the allocation logic is defined in *Rembrandt*, which will be able to create tour resources out of the given drivers and parcels at run time. To test the flexibility, we decided to use two approaches for the tour optimization, a rule-based approach (as provided by BPMSs) and an heuristic-based approach (classical OR approach). Both were developed in cooperation with the logistics experts from the SMile project:

- Rule-based: This approach is based on the FIFO principle [32], standard of current BPMSs. Therefore, it considers at most one parcel at a time (i.e., local optimization approach) and adds it to the tour of a carrier until the tour cannot be extended further.
- Heuristic-based: For this approach, the Munkres [19] algorithm is reused and adapted to our setting. It handles all parcels concurrently (i.e., a global optimization

approach). In each iteration, it will extend existing tours by at most one parcel, or it will add at most one new tour.

For each of the approaches, we employed one recipe in the resource optimization part of *Rembrandt* as shown in Fig. 12. Both recipes were similar in their structure, but called a different tour optimization algorithm. At last, the resource allocation configuration is carried out so that the corresponding activity in the process is connected to the newly created recipe in *Rembrandt*.

Run time To simulate different realistic situations, three different scenarios with recipient's addresses and time frames were developed. The scenarios differ in the number of parcels that have to be delivered in the time frame of 6 to 10 pm (1: 20 parcels; 2: 100 parcels; 3: 35 parcels).

To reflect the real-world occurrence of delivering multiple parcels to the same address, we introduce the notion of stops that describes one address where at least one parcel gets delivered to. Furthermore, each driver can deliver up to 35 parcels at a time. Each scenario was simulated ten times with a different set of addresses randomly drawn from a pool of addresses within the ZIP area 10585 in Berlin.

At run time, for each parcel, the process is invoked and the "Assign parcel to tour" activity is executed, which in turn requests *Rembrandt* to run the previously configured allocation algorithm.

Metrics Since all three scenarios were executed twice, once for each allocation algorithm, the resource-aware extension allows deriving further insights. While testing the use cases, we observed that the Munkres approach achieves a much shorter total distance to deliver every parcel, as shown in Table 2. At the same time, the Munkres approach in general creates more tours, which, in combination with the advantage of deciding the allocation for all cases at once, results in more optimized tours. Based on the use case, we could observe that the stops seem to be the same, independent of the approach used. In the case of 100 parcels, the rule-based approach more frequently adds parcels with the same address to different tours, such that optimization potential is unused as it has to follow the temporal order of instance creation. It is also noticeable that the overall computation time tends to stay similar for both approaches. For the case with 100 parcels, we can observe small differences.

Discussion The use case shows the feasibility of the framework to support run-time decisions on allocating resources to tasks. We showed that existing (rule-based) as well as more sophisticated (Munkres) algorithms can be incorporated and used within the resource-aware extension, both assigning *many* tasks to one carrier. In contrast to related work, we could show that we can incorporate different types of allocation approaches, a local and a global one that use different techniques, a rule and an heuristic.

In general, more sophisticated algorithms have to deal with the trade-off between quality and the computation time needed to achieve a result. However, because of the advancement of technology and its computation power, we did not notice any significant difference with regards to time. At the same time, the quality of the result increases significantly with the sophisticated algorithm: Only half the distance was needed for the use case with the fewest parcels, and only a fourth was needed for the use case with the most parcels. All in all, the experiment shows that a smarter decision

Table 2 Result of running the three test cases. As each case was simulated ten times, the resulting values are the averages of the test runs

	Case 1		Case 2		Case 3	
	Rule-based	Munkres	Rule-based	Munkres	Rule-based	Munkres
Computation time	0.78 s	0.75 s	2.66 s	3.13 s	1.26 s	1.23 s
Number of tours	1	3.3	3	5	1	3.6
Total distance	10 km	4.6 km	37.1 km	9.1 km	15 km	6.6 km
Number of stops	14	14	76.5	76.1	24	24

for the resource allocation has a substantial influence for an organization, it can act much more efficiently. With the architecture provided, different resource allocation procedures can be simply added by a new recipe calling the new algorithm. These can be tested, deployed, as well as exchanged at any time.

8 Conclusion

This research work provided a conceptual framework for supporting the decision-making of resource allocations in business processes and the employment of optimization techniques of operation research problem-specific for process activities. The framework was realized as a software architecture extending a traditional BPMS with a dedicated resource manager and was implemented as a prototype. Experiments in a simulated logistics real-world example showed the advantage of employing a problem-specific optimization algorithms and the benefits of exchanging different allocation algorithms at design time.

In existing organizational settings, decision-making for resource allocations is still a challenging task because a transparency on resources and their availability is missing. The designed resource manager of this work eases this task and gives support in collecting the information centrally. With the advance of digitization, we believe that the process of collecting this information organization-wide in the resource manager will get more comfortable.

The resource manager is designed in a way, such that a set of allocation requests by process instances of a certain time frame are batched until a resource allocation algorithm is started again. That is a simple option to consider priorities of different process instances. For prioritizing certain business cases, operations research offers different static and dynamic batching approaches, whose application we want to investigate in the future.

Even though the main contribution of the framework is related to the efficient execution of business processes at run time, namely by influencing the resource allocation, we think the framework is potent for other applications, too. The aspect of process performance is also the focus of process analysis and improvement. With the lack of a framework that connects processes and resource information on an organizational level, such analysis, e.g. with process mining techniques, could so far identify symptoms for a bad process performance as well as guess root causes for treatment. Irregularities and bottlenecks can be identified, but as there was no whole picture of the organization available, the effectiveness of the suggested improvement is limited. By using our framework, all made allocations and the corresponding circumstances can be recorded. This way, we can enrich the analysis by providing the information on the available resources and running process cases at the same time, which in turn enables a meaningful root cause analysis and treatment.

Acknowledgements The research leading to these results has been partly funded by the BMWi under grant agreement 01MD18012C, Project SMile. <http://smile-project.de>.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Arias M, Munoz-Gama J, Sepúlveda M, Miranda JC (2018) Human resource allocation or recommendation based on multi-factor criteria in on-demand and batch scenarios. *Eur J Ind Eng* 12(3):364–404
2. Arias M, Saavedra R, Marques MR, Munoz-Gama J, Sepúlveda M (2018) Human resource allocation in business process management and process mining: A systematic mapping study. *Manag Decis* 56(2):376–405
3. Bang-Jensen J, Gutin G, Yeo A (2004) When the greedy algorithm fails. *Discret Optim* 1(2):121–127
4. Bellaaj Elloumi F, Sellami M, Bhiri S (2018) Avoiding resource misallocations in business processes. *Concurrency and Computation: Practice and Experience* 32:e4888
5. Cabanillas C (2016) Process-and resource-aware information systems. In: *EDOC, 2016 IEEE 20th International, IEEE*, p 1–10
6. Cabanillas C, García JM, Resinas M, Ruiz D, Mendling J, Ruiz-Cortés A (2013) Priority-based human resource allocation in business processes. In: *ICSOC, Springer*, 374–388
7. Djedovic A, Karabegovic A, Avdagic Z, Omanovic S (2018) Innovative approach in modeling business processes with a focus on improving the allocation of human resources. *Math Probl Eng*
8. Doerner K, Gutjahr WJ, Kotsis G, Polaschek M, Strauss C (2006) Enriched workflow modelling and stochastic branch-and-bound. *Eur J Oper Res* 175(3):1798–1817
9. Dumas M, La Rosa M, Mendling J, Reijers HA et al (2013) *Fundamentals of business process management*, vol 1. Springer, Berlin
10. Durán F, Rocha C, Salaün G (2019) A rewriting logic approach to resource allocation analysis in business process models. *Sci Comput Program* 183:102303
11. Ha BH, Bae J, Park YT, Kang SH (2006) Development of process execution rules for workload balancing on agents. *Data & Knowl Eng* 56(1):64–84
12. Havur G, Cabanillas C, Mendling J, Polleres A (2016) Resource allocation with dependencies in business process management systems. In: *BPM, Springer*, pp 3–19
13. Hirsch MJ, Ortiz-Peña H (2017) Information supply chain optimization with bandwidth limitations. *Int Trans Oper Res* 24(5):993–1022
14. Huang Z, van der Aalst WM, Lu X, Duan H (2010) An adaptive work distribution mechanism based on reinforcement learning. *Expert Syst Appl* 37(12):7533–7541
15. Huang Z, van der Aalst WM, Lu X, Duan H (2011) Reinforcement learning based resource allocation in business process management. *Data & Knowl Eng* 70(1):127–145
16. Huang Z, Lu X, Duan H (2012) A task operation model for resource allocation optimization in business process management. *IEEE Transactions on Systems, man, and cybernetics-part a: systems and humans* 42(5):1256–1270
17. Ihde S, Pufahl L, Lin MB, Goel A, Weske M (2019) Optimized resource allocations in business process models. In: *Business Process Management Forum. BPM 2019.*, Springer International Publishing, Cham, pp 55–71
18. Kamrani F, Ayani R, Moradi F (2012) A framework for simulation-based optimization of business process models. *SIMULATION* 88(7):852–869
19. Kuhn HW (1955) The hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1–2):83–97
20. Kumar A, Van Der Aalst WM, Verbeek EM (2002) Dynamic work distribution in workflow management systems: How to balance quality and performance. *J Manag Inf Syst* 18(3):157–193
21. Kumar A, Dijkman R, Song M (2013) Optimal resource assignment in workflows for maximizing cooperation. In: *BPM, Springer*, pp 235–250

22. Liu T, Cheng Y, Ni Z (2012) Mining event logs to support workflow resource allocation. *Knowl-Based Syst* 35:320–331
23. OMG (2011) Notation BPMN version 2.0. OMG Specification, Object Management Group pp 22–31
24. Pellerin R, Perrier N, Berthaut F (2019) A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *EJOR*
25. Plug J, Rinderle-Ma S (2016) Application of dynamic instance queuing to activity sequences in cooperative business process scenarios. *Int J Coop* 25(01):1650002
26. Pufahl L, Ihde S, Glöckner M, Franczyk B, Paulus B, Weske M (2020) Countering congestion: A white-label platform for the last mile parcel delivery. In: 23rd BIS Conference, Springer, pp 210–223
27. Pufahl L, Ihde S, Stiehle F, Weske M, Weber I (2021) Automatic resource allocation in business processes: A systematic literature survey. [arXiv:2107.07264](https://arxiv.org/abs/2107.07264)
28. Reijers HA, Jansen-Vullers MH, Zur Muehlen M, Appl W (2007) Workflow management systems+ swarm intelligence= dynamic task assignment for emergency management applications. In: *BPM*, Springer, pp 125–140
29. Rhee SH, Cho NW, Bae H (2010) Increasing the efficiency of business processes using a theory of constraints. *ISF* 12(4):443–455
30. Russell N, van der Aalst WMP, ter Hofstede AH, Edmond D (2005) Workflow resource patterns: Identification, representation and tool support. In: *CAiSE*, Springer, pp 216–232
31. Schöning S, Cabanillas C, Jablonski S, Mendling J (2016) A framework for efficiently mining the organisational perspective of business processes. *Decis Support Syst* 89:87–97
32. Tanenbaum A (2009) *Modern operating systems*. Pearson Education, Inc.,
33. Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications*. No. 18 in *MOS-SIAM Series on Optimization*, SIAM
34. Van Hee K, Reijers H, Verbeek H, Zerguini L (2001) On the optimal allocation of resources in stochastic workflow nets. In: *UK Performance Engineering Workshop*, Print Services University of Leeds, pp 23–34
35. Weske M (2019) *Business Process Management - Concepts, Languages, Architectures*, 3rd edn. Springer, Berlin
36. Xu J, Liu C, Zhao X (2009) Resource planning for massive number of process instances. In: *CooPIS Conference*, Springer, pp 219–236
37. Xu J, Liu C, Zhao X, Yongchareon S, Ding Z (2016) Resource management for business process scheduling in the presence of availability constraints. *ACM TMIS* 7(3):1–26
38. Yaghoibi M, Zahedi M (2017) Cycle time reduction and runtime rebalancing by reallocating dependent tasks. *Int J Eng* 30(12):1831–1839

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.