Optimized Resource Allocations in Business Process Models

Sven Ihde[1(✉)], Luise Pufahl[1], Min-Bin Lin[2], Asvin Goel[2], and Mathias Weske[1]

[1] Hasso Plattner Institute, University of Potsdam, 14482 Potsdam, Germany
{sven.ihde,luise.pufahl,mathias.weske}@hpi.de
[2] Kühne Logistics University, Hamburg, Germany
{min-bin.lin,asvin.goel}@the-klu.org

Abstract. The allocation of resources to process activities can have a huge influence on overall performance, in particular, if resources are costly and limited in their availability. Rule-based allocations can lead to unnecessarily low resource utilization rates, high costs, and large delays. In this paper, we present a framework allowing for optimized resource allocations by extending a traditional Business Process Management System by a new component that we call the *Resource Manager*. Our framework allows a process designer to specify resource requirements which are used by the Resource Manager to decide on allocations of resources to process activities. We describe the functionality of the Resource Manager, its interaction with the process engine, and the data needed. The framework is implemented by extending an open-source process modeler and engine, and applied to a use case concerning the last mile delivery.

Keywords: Resource allocation · Resource optimization · Business process modeling · Business Process Management System

1 Introduction

Business processes are indispensable for organizations to achieve their goals for providing goods and services. As many resources are cost-intensive and limited [3], process performance is highly dependent on an efficient allocation of resources to tasks. Inefficient resource allocation can lead to low resource utilization, high costs, large delays, and low quality [7]. Resource allocation ensures that each activity of a certain process instance (i.e., a task) is executed at the right time and with the right resources, thereby balancing the demand of process execution with the availability of resources [21].

Business Process Management Systems (BPMSs) coordinate process activities, resources, and data given in a process model and allow process automation [13,38]. Existing BPMSs provide two basic allocation mechanisms: push and pull [34]. Using the push mechanism, tasks are pushed to qualified resources based on pre-defined rules (e.g., the FIFO mechanism) that are specified at design time [8,33]. A summary of those are given by the so-called resource allocation patterns [34]. Using the pull mechanism, tasks are provided in a task list to

T. Hildebrandt et al. (Eds.): BPM 2019, LNBIP 360, pp. 55–71, 2019.
https://doi.org/10.1007/978-3-030-26643-1_4

resources (usually human actors), such that they can prioritize and select based on their knowledge. These two approaches might be useful for several use cases, but are too limited for the domains such as logistics, healthcare, or manufacturing, with complex process structures and costly resources requiring advanced resource allocation mechanisms [18,29]. In the SMile project[1], which motivates this research, cost-efficient vehicle routes must be generated for delivering parcels with promised delivery time windows.

In the business process management (BPM) domain, the resource perspective was intensively studied [5]. Especially the definition of resource requirements is supported (see [3,9]). Different approaches were developed to improve resource allocations by generating an optimized resource allocation plan for a process instance [18], by mining resource allocation rules [24], or by dynamically allocating resources based on historic logs [21]. However, most of them are stand-alone solutions for specific problems and not necessarily integrated into the process configuration and execution. In the community, the need for a resource-aware BPMS being capable of smart resource allocation has been identified [8,29].

In this paper, we present a new component called *Resource Manager* that extends the traditional BPMSs and is responsible for the allocation of resources to tasks. We describe how the Resource Manager can be integrated into a traditional BPMS and which information must be provided by the process modeler. Our proposed architecture allows to provide use cases with specific constraints and performance measures that can be applied to optimize resource allocations. The Resource Manager can rely either on a human decision maker who manually allocates resources to tasks, or on an optimization algorithm for the automatic allocation of resources. To demonstrate the practicality of this research the Resource Manager is implemented to a real-world logistics process.

In the reminder, existing works about the resource perspective of business processes are discussed in Sect. 2. After describing a motivating example Sect. 3, some background on resource-constrained scheduling problems is given in Sect. 4 and requirements are discussed in Sect. 5. Section 6 describes the extended architecture of a resource-aware BPMS and shows how resource allocations can be optimized by a resource manager. A prototypical of the extended architecture and its application to a real-world use case are presented in Sect. 7. Limitations and future work are discussed in Sect. 8.

2 Related Work

In addition to the control-flow and data-flow perspectives of business processes, the resource perspective is essential for the successful execution of business processes [13]. Cabanillas [8] distinguishes among three key operations of resource management: (1) resource assignment (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 runtime), and (3) resource analysis (i.e., evaluation of process execution with the focus on resources).

[1] http://smile-project.de.

Recent works have focused on resource meta-models for a detailed specification of resource characteristics to support resource assignment, such as the organizational meta-model [34], resource meta-model [27], or resource classification role-model [29]. A taxonomy of the different approaches is given by Arias et al. [3]. Advanced textual and graphical resource assignment languages were developed in [9] that go beyond simple role-assignment being available in the standard process modeling notation BPMN [28].

Russel et al. [34] introduced 43 workflow resource patterns; a comprehensive study on use and representation of (mostly) human resources in the existing BPMSs. It also included allocations patterns (i.e., role-based, history-based, etc.), which are greedy solutions that may become sub-optimal, especially as resources are not independent from each other [18]. Batch activities developed by [32] efficiently combine the execution of certain activities from several process instances to optimize resource utilization. However, these approaches only support a fixed set of allocation algorithms and only one type of resource. Considering complex resource constraints, Senkul et al. [36] and Havur et al. [18] use logic programming to identify optimal resource allocation plans for deterministic process instances.

Based on historical execution logs, it is possible to generate allocation rules using machine learning techniques [24]. Liu et al. [25] gives resource allocation recommendations to a human assigner during execution based on historical logs with a supervised machine learning algorithm that does not consider the availability or workloads of resources. This work is extended by Arias et al. [4] considering the current workloads of resources, and by Zaoh et al. [39] applying a heuristic to handle several process instances running in parallel. Huang et al. [21] uses reinforcement learning for dynamic resource allocation and considers the influence of resource allocations on the process environment.

In summary, various approaches for resource allocations have been proposed already. However, they are mainly based on rule-based approaches and efficient resource allocation is derived from a business process perspective. From the literature on optimization we know that rule-based approaches usually cannot guarantee an efficient utilization of resources. Additionally, as organizations have multiple processes working on the same limited set of resources, we propose to plan resource allocation form a resource point of view. The Operation Research community has a long history of developing approaches for optimizing resource allocations that way. In order to leverage on these achievements, we propose a framework which integrates sophisticated solution techniques developed in the Operations Research community into traditional BPMSs.

3 Motivating Example

Online shopping has led to an increase in parcel deliveries. The delivery of parcels in the last mile (from the final hub to the recipients) imposes various challenges on logistics service providers in particular due to low profit margins and a high risk of not being able to deliver parcels if recipients are not at home [35]. The

SMile projects aims to innovate the last mile logistics by ensuring a delivery at the first try in a so-called *micro-depot*. In a micro-depot a parcel can be delivered by small and local delivery services to a recipient at a desired time frame. This section presents this last mile delivery process shown in Fig. 1.

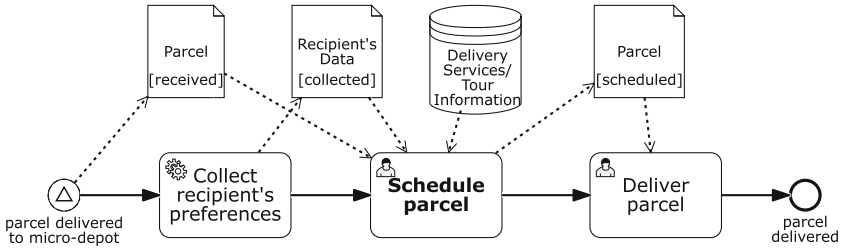

Fig. 1. Process model of parcel deliveries with promised delivery time windows.

When a parcel is delivered to a micro-depot, the recipient's preferences are collected. Based on the preferences, the delivery is scheduled and when the scheduled time is reached, the parcel is delivered to the customer. In order to efficiently utilize the delivery vehicles, the scheduling of individual parcels must be conducted in such a way that cost-efficient routes for vehicles can be found, allowing to deliver parcels at their scheduled times. Thus, the scheduling represented as a simplified user task (cf. Fig. 1) is a complex task, where a suitable vehicle (i.e., resource) and other available parcels in the micro-depot (i.e., other process instances) should be considered. In Fig. 1 it is assumed that this task is done manually by a dispatcher, who has knowledge about available resources and delivery requests. However, the task could also be executed by an engine using appropriate optimization algorithms. The use of such algorithms would not only help in improving process performance by allowing for better delivery routes, it would also allow for full automation of decision making.

This example shows that business processes often include process activities for which an optimized scheduling is required considering the currently running process instances and available resources. This can be also observed in other domains where resources must be allocated to process activities subject to the limitations in the way they can be used.

4 Resource-Constrained Scheduling Problems

The scheduling task of our motivating example requires the solution of a special case of a *resource-constrained scheduling problem* (RCSP). RCSPs are problems in which tasks must be scheduled, and where the execution of the tasks depends on one or more resources and their limited availability [17]. The usual goal of RCSPs is to find a schedule for all relevant tasks to minimize the total

resource utilization cost. However, other optimization goals like minimizing the time required for executing all tasks or tardiness may also be specified.

For some RCSPs optimal solutions can be found by using exact approaches based on mixed-integer linear programming [22] and branch-and-bound algorithms [12]. However, as these problems are usually $\mathcal{NP}$-hard [23], they are notoriously difficult to solve, in particular for problems of relevant size. Therefore, RCSPs are usually solved using so-called metaheuristics [30]. While it is possible to search for solutions of RCSPs using rule-based approaches, the solutions are usually of poor quality [6] resulting in unnecessarily high costs, as rule-based approaches can be interpreted as very primitive heuristics. Sophisticated heuristics systematically exploring the search-space and escaping local optima of poor quality can often find near-optimal solutions in reasonable amount of time [14].

In the motivating example, the limited resources are vehicles. Vehicles can conduct multiple parcel deliveries in one route, however, they are limited concerning the total volume and weight of the parcels. Furthermore, different parcels have different destinations and different required times of the delivery. This heterogeneity of the tasks results in the need to plan the order of deliveries and the allocation of vehicles to delivery requests. Poorly planned schedules result in unnecessarily long (and costly) travel times between delivery locations and an unnecessarily high number of vehicles required. An overview of optimization approaches that can be used for similar problems related to the transportation of goods can be found in [37].

While the remainder of this paper focuses on our motivating example, the framework presented in this paper is not restricted to this particular example. For other application scenarios, similar RCSPs may have to be considered, for example, in manufacturing or healthcare. In the following we provide brief examples of typical tasks and resource-requirements in such domains to illustrate the variations of application scenarios that can benefit from our framework.

Manufacturing. Resources in the manufacturing industry are broadly defined as production units, e.g., machines, human operators, material handling vehicles, or storage buffers, which are limited. Multiple tasks are required to be performed by resources to create desired goods. According to types of manufacturing systems, operational processes can be altered [1]. For example, a job shop, which produces goods in high variety but low volume, involves diverse tasks that consist of several operations which have to be processed on different resources, e.g., tool and die making. In contrast, in a flow shop, tasks have to be processed in a consistent order on all resources, e.g., automobile assembly line. Additionally, batch production can occur in any manufacturing system, where components or goods proceed to various stages, in groups of a certain size, to different resources. Thus, resource allocation here is highly related to the design of manufacturing systems. Some common objectives are throughput, makespan, lead time, and resource utilization considering line balancing.

Healthcare. In healthcare services, resources typically refer to doctors, nurses, equipment, and rooms. As available healthcare resources are considerably limited in comparison to patient demands, effectively allocating resources is crucial in enhancing service quality for patients. Moreover, doctors and nurses are restricted in their sub-specializations, for example: surgeon or orthopaedist. The patient groups are likewise bound in their needs to a specialized professional and are thus required to utilize a specific resource [11,26]. Medical staff are generally shared between several patients and are often blocked by who can be served when, and the same is true for equipment and rooms. Hence, to coordinate resource availability with the right resource skills to patient demands, multi-skill resources should be considered in the healthcare resource allocation [2]. General objectives in this class of problems are patient waiting time, makespan, resource utilization, and the choice of doctor preferences.

Given the heterogeneity of RCSPs, no single model can be defined that is suitable for all possible application scenarios. However, all problems have in common that it usually benefits from using sophisticated optimization algorithms exploiting knowledge about the entirety of all tasks and all resources compared to simply assigning tasks to resources one by one.

5 Requirements

Usually multiple process instances simultaneously and concurrently require limited resources. Although greedy rules as provided by the workflow patterns [34] or mined from historical data could be used to allocate resources, such approaches often lead to unnecessary inefficiencies as discussed in the previous section. In order to optimize resource allocations within business processes, a BPMS should provide the possibility to configure and solve a RCSP capturing the characteristics of the specific use case. For this we can identify multiple requirements that should be fulfilled by a BPMS:

R1 *Knowledge on resources and their availability.* To enable an optimized resource allocation, the BPMS needs to have knowledge of the resource characteristics as well as their current or future availability [29]. Therefore, static and dynamic information concerning the resources must be captured.

R2 *Resource requirements.* The BPMS must allow a process designer to specify when which resource is required in which quantity. For more complex application scenarios multiple resources may be required simultaneously. Thus, it should be able to define the information in a process model.

R3 *Constraints and performance measures.* The BPMS must allow a designer to specify the constraints on the utilization of resources and the impact of resource utilization on the overall performance. The constraints and performance measures define the problem of allocating resources.

R4 *Allocation service.* While the problem definition specifies the constraints and goals, it does not describe how the allocation is conducted. As the allocation of resources can be conducted in many different ways, the BPMS must allow

the specification and implementation of one or several alternative allocation services that can be selected by the designer.

Given that the problem of allocating resources to tasks is a complex problem that can be tackled in many different ways (e.g. manually or by the use of optimization algorithms), we can observe that there is a need for separation of concerns between the process and resource allocation aspects.

6 Optimized Resource Allocation in Business Processes

In this section we propose to extend the traditional BPMS with additional components for resource management that completely encapsulate the knowledge of resources and their optimized allocation. The extension to the resource components and their interaction with traditional components is presented on the architectural level in Sect. 6.1. In Sect. 6.2 we show how the current process modeling needs to be extended in order to specify the requirements on the optimized resource allocation for process execution. Finally, in Sect. 6.3, the detailed functionality while executing a *resource-allocation* activity is explained.

6.1 Architecture of Resource-Aware BPMS

In Fig. 2, the extended BPMS architecture for realizing optimized resource allocation is presented. It consists of the traditional components of a BPMS and the extension for the resource components. First, we shortly repeat the structure of a traditional BPMS before we introduce the resource-aware extension with the *Resource Modeling* and the *Resource Manager*.

Traditional BPMS. In general, a BPMS consists of four main components: *Process Modeling*, *Process Engine*, *Environment*, and *Applications* [13,38]. The *Process Modeling* component supports the *Process Designer*s to model their business processes with all the relevant technical information for an automatic process execution, such as user interfaces, services to-be called, or resource assignment information. This is stored in a *Process Model Repository* and can be also updated. The *Process Engine* is the main component for guiding the process execution, which can access the repository. An instance of a process can be manually started or automatically triggered in the process engine. Then, the process engine initiates the process activities in their prescribed sequence. Business processes involve different *Process Participant*s in order to execute/check/document certain process steps. The participants are connected mostly via a User Interface that shows them the corresponding tasks to be done summarized as *Environment*. In case of service activities in a process, other services might be called grouped together as *Applications* representing of all kinds of services.

Resource-Aware Extension. The resource-aware extension of a BPMS consists of two components: *Resource Modeling* and *Resource Manager*.

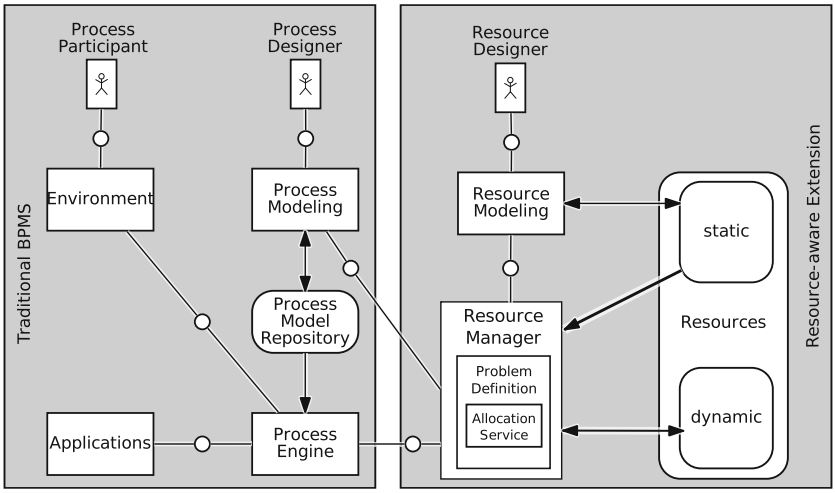

Fig. 2. Architecture of a resource-aware BPMS for smart resource allocation (The diagram is a FMC block diagram (http://www.fmc-modeling.org) representing active software components as rectangles and passive data storage as ellipses).

Resource Modeling. The purpose of the *Resource Modeling* component is to define and store all relevant information concerning resources and the constraints on the resource utilization. Similar to process modeling, an expert is assumed, the *Resource Designer*, who is responsible for prescribing the information and constraints on resources, referred to as static resource data. The static resource data includes, for example, the cost of a resource per use/time-frame, its maximum capacity, type of resource (e.g., reusable, consumptive, or producible). Different organizational resource models were proposed [27,29] which could be used as starting point for a structure of the static resource data. However, use case-specific resource models are required in many cases.

Resource Manager. The purpose of the *Resource Manager* component is to decide on the allocation of resources to process tasks. A resource manager can handle multiple problem definitions based on the constraints provided during resource modeling. For each problem definition, the resource manager provides an interface in which the requested input and expected output for a resource allocation request is defined. During process modeling, dedicated *resource-allocation* activities (further discussed in the next subsection) can be used to request a resource allocation for a selected problem definition. For each problem definition one or several *Allocation Services* can be implemented. An allocation service can be a simple rule-based algorithm, a heuristic, a sophisticated optimization algorithm, a mined allocation or even an interactive component combining algorithms with expert knowledge via the user interface. At process design time, the allocation service can be selected for a resource-allocation activity.

Whenever a resource-allocation activity is executed by the process engine, it maps the respective data objects of the activity to the required inputs and expected outputs of the problem definition. The data is assumed to contain all relevant information allowing the resource manager to update information about current and future availability of resources, i.e., dynamic resource data.

6.2 Configuration

In the following, we define dedicated activities for resource allocations and their configuration. Assume we are given a set of business process models P modelled by the process designer and a set of problem definitions R provided by a resource manager. For each process model $p \in P$ let A_p denote the activities of the process model. Then, the process designer can specify a subset $A_p^* \subset A_p$ representing dedicated activities for requesting the allocation of resources. We refer to these activities as *resource-allocation activities*. For each of these resource-allocation activities $a \in A_p^*$ the process designer must select a problem definition $r \in R$.

For each problem definition r, the required input I_r and expected output O_r is defined, as a set of data attributes $i = (k_i, t_i)$ where k_i is a key that can be used to identify a data item and t_i specifies the data type. These data types can be elementary types or arbitrary complex data types. Section 7.2 shows the data definition with non-elementary data types for our motivating example.

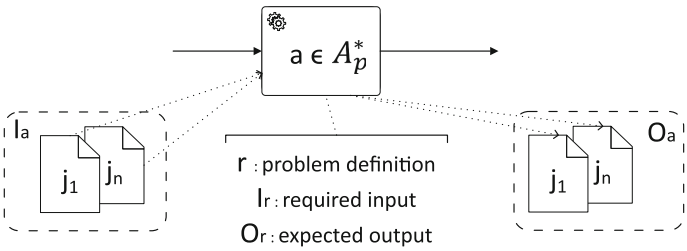

Fig. 3. *Resource-allocation* activity shown as BPMN activity.

The process designer ensures that the data input and output of a resource-allocation activity a is aligned with the input and output of the problem definition r selected for the activity a. The process designer similarly defines the allowed data input I_a for the activity a as a set of process variables $j = (k_j, t_j)$. The proposed resource-aware BPMS checks that for each data attribute $i \in I_r$, a process variable $j \in I_a$ exists for which t_j can be mapped to t_i. Furthermore, the resource-aware BPMS checks that for each data attribute $i \in O_r$, a process variable $j \in O_a$ exists for which t_i can be mapped to t_j. Thus, any output of the resource manager can be mapped to data output of the resource-allocation activity. Figure 3 illustrates the data in- and output.

During process execution, the process engine generates a data input for each resource-allocation activity a that can be represented by a set of process instance

variables $j = (k_j, v_j)$ where k_j is the key of the process variable and v_j is the value which must be of the correct type t_j according to the process variable for key k_j. This data input is mapped to the expected input of the problem definition which generates an output that is then mapped to the output of the resource-allocation activity.

The advantage of this proposed configuration is, that it provides a high degree of flexibility. The process designer may, for example, specify all the data that is available and potentially relevant for the resource allocation. A simple allocation service, that may be initially deployed, may not require all the data and may work on a limited subset thereof. If, at a later stage, a more sophisticated allocation service with higher data requirements shall be deployed, the process model does not need to be changed.

6.3 Functionality

The execution of the resource-allocation activity at runtime and the interaction between the process engine and resource manager is visualized as a UML sequence diagram in Fig. 4.

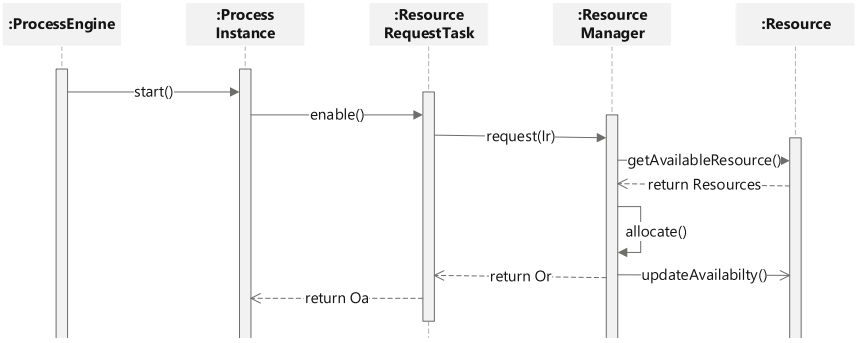

Fig. 4. Interaction between the Process Engine and Resource Manager for executing a *resource-allocation* activity shown as UML sequence diagram.

The `ProcessEngine` instantiates a process model to start a new `Process Instance`. During the execution of the process instance, activities are enabled and executed based on the prescribed order in the process model. In the case of a `Resource AllocationActivity`, the selected interface of the `ResourceManager` is called with the input data I_r for the respective optimization problem. The `ProcessInstance` is now in a waiting state until the request is answered by the `ResourceManager`. After receiving a request the `ResourceManager` collects the information on the resource(s), which can be used to fulfill the request, and checks their availability. In the next step, the `ResourceManager` uses the allocation service for the selected problem definition in order to find a suitable resource

allocation. The `ResourceManager` collects multiple requests and conducts the resource allocation for these requests together. After running the resource allocation, it returns the resource(s) O_r to the requesting *resource-allocation* activity. The resource(s) are now reserved for use in the process instance. With the information on the planned usage, the `ResourceManager` updates the availability of the resource. The entry is a timeframe, when the resource is not available in future and cannot be used for any other allocations in this specific timeframe.

Currently, we assume that the usage of the resources is executed as planned by the resource manager. However, for treating delays and exceptions the resource manager needs to be informed about the actual start and the end of usage of resources by process instances. It could be realized, for example, by having events published by the process engine about the start and end of process activities as described in [19] to which the resource manager can subscribe. Then the resource manager can compare the actual start and end to the planned allocation information. If there are any variations to the plan, the resource manager might need to adapt previous plannings. This exception handling will not be the focus of the current work.

7 Evaluation

This section presents an evaluation of the resource-aware BPMS extension in a two-fold manner: (1) a prototypical implementation is presented in Sect. 7.1, and (2) an application of the approach to a logistics use case is illustrated in Sect. 7.2.

7.1 Prototypical Implementation

As shown in Fig. 5, we used for our prototypical implementation *Chimera*[2] [20] – a process engine for flexible, knowledge-intensive processes. It also provides a process modeler – *Gryphon*, which is based on the open-source BPMN modeler *bpmn.io*. As soon as the process model is designed and configured in *Gryphon*, it can be stored in the *Process Models* repository and can be deployed to *Chimera* engine as depicted in Fig. 5. As these two components represent a traditional BPMS, we used it as a base and adapted it to fit our architecture proposal in Fig. 2. For this we implemented a simplified Resource Manager, the *Sphinx*[3] that is connected to *Chimera* and *Gryphon* via REST interfaces.

The interface between *Sphinx* and *Gryphon* is used during design time and provides the information a process designer needs for configuring the task, as shown in Fig. 6. That is why we extended *Gryphon*[4] to allow the configuration of resource-allocation activities. It allows to specify the Resource Manager, a problem definition, and the allocation service that is used (see requirements R2

[2] https://bptlab.github.io/chimera/.
[3] https://github.com/bptlab/smile/tree/master/sphinx.
[4] https://github.com/bptlab/gryphon/tree/resource/add-resource-type.

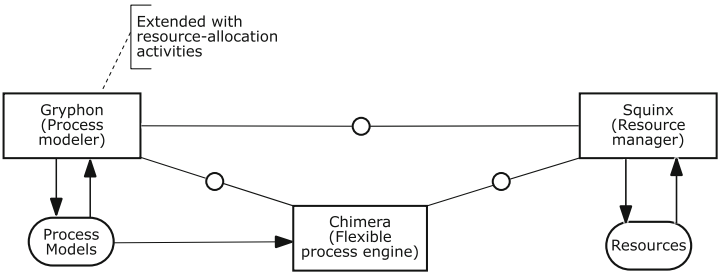

Fig. 5. Architecture of prototypical implementation

and R4). Based on this selection the corresponding input and output sets are shown, that have to be mapped by the designer.

If *Chimera* reaches a resource-allocation activity during the process execution, then it calls the referenced Resource Manager with the defined input. Based on the steps described by Fig. 4 the Resource Manager will access its resource information to find the best possible allocation, as shown in the next section.

7.2 Application for Our Motivating Example

Problem Definition and Allocation Algorithm. For an efficient last-mile delivery, delivery routes must be found which minimize total distance travelled. In our motivating example, the deliveries must be conducted between an earliest and a latest given delivery time. The optimization problem for finding efficient delivery routes is a so-called *vehicle routing problem with time windows (VRPTW)* [37]. A rule-based solution would, for example, allocate the next available vehicle to a request. As parcels can arrive at the micro-depot at any time, not all delivery requests are known to the Resource Manager when a vehicle is requested, thus resulting in the rule-based approach leading to an overall poor solution. Instead routes have to be created given all the information known to date and revised when new information about request becomes available, which is possible with a heuristic. This variant of the vehicle routing problem is known as a *dynamic vehicle routing problem (DVRP)* [31].

In our implementation we used an *insertion method* [10] as the allocation service. An insertion method successively selects unscheduled requests and adds them to the route of a vehicle. For each request the method determines all feasible insertion possibilities. That is, for each vehicle route and each request already in the route, the algorithm checks whether the unscheduled request can be added to the route before or after the already scheduled request. If the request can be feasibly inserted, the method inserts the request at the position in the route which has the lowest incremental costs. If no feasible insertion possibility is found, a new route is created and the request is added into the new route.

To handle the dynamics of parcels arriving at the micro-depot, the deterministic VRP is iteratively solved with a rolling-horizon procedure [15] in which the

optimization period moves forward at each iteration and adds each time period to the total planning time horizon. The rolling-horizon procedure allows the schedule for the previous planning time period, which is actually implemented, to be influenced by the future in the form of the next planning time period. The insertion method is a straightforward strategy that can easily adapt complex constraints and generate a feasible solution within a short computation time.

In this paper, we use the insertion method due to its simplicity and ease of implementation, as our goal is to provide a proof-of-concept demonstrating how an algorithm is used for resource allocations. However, it is easy to replace the allocation service by more powerful algorithms for dynamic vehicle routing [31] or by manual or even interactive dispatching systems [16]. From the literature on vehicle routing problems it is known that the performance of delivery processes can be significantly improved by state-of-the-art planning approaches.

Configuration of the Resource Allocation Task for Parcel Delivery. The process of our motivating example shown in Fig. 1 can be easily adapted to leverage the functionality of our resource manager and to fully automate the scheduling of parcels.

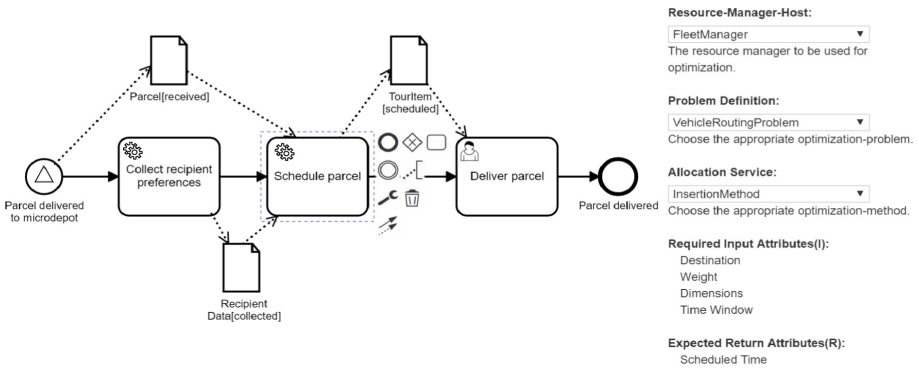

Fig. 6. Configuration in Gryphon

As shown in Fig. 6 we replace the user task for scheduling the parcel by a service task. This service task is a resource-allocation task for which we implemented a configuration sidebar allowing the selection of a *Resource Manager Host*, a *Problem Definition*, and an *Allocation Service*. We implemented a resource manager named *Fleet Manager* that is responsible for the allocation of vehicles in the fleet. The *Fleet Manager* provides a list of problem definitions and allocation services. From this list the process designer can choose the *Vehicle Routing Problem* as problem definition and the *Insertion Method* as allocation service. The chosen resource managers defines the *Required Input*, and *Expected Return* which are shown to the process designer in the sidebar.

The input definition provided by the resource manager includes data definition items $(\text{“}Destination\text{”}, (\texttt{float}, \texttt{float}))$ for the geographic coordinates of the destination of the parcel, $(\text{“}Weight\text{”}, \texttt{float})$ for the weight of the parcel, $(\text{“}Dimensions\text{”}, (\texttt{float}, \texttt{float}, \texttt{float}))$ for the width, height, and length of the parcel, and $(\text{“}TimeWindow\text{”}, (\texttt{time}, \texttt{time}))$ for the earliest and latest delivery times. The process designer has to make sure that for each process instance (i.e., each parcel to be delivered) the respective data items are generated during process execution and provided to the resource-allocation task by data objects. The output definition provided by the resource manager includes the data definition item $(\text{“}Scheduled\ Time\text{”}, \texttt{time})$ representing the scheduled delivery time. The process designer can utilize the respective output by passing a data object from the resource-allocation activity to subsequent activities for further utilization. In our example, the data item for the scheduled delivery time is included into a data object that is used by the activity *Deliver parcel*.

8 Conclusion

In this paper, we provide a framework to integrate optimized resource allocations in business processes by extending the traditional BPMS architecture through a component called the Resource Manager. The Resource Manager is responsible for maintaining all relevant information concerning the availability of resources and for allocating resources to a process instance. The process designer can specify resource requirements within the business process model through dedicated resource-allocation activities. By implementing and running our approach for a use case concerning time-constrained parcel deliveries, we demonstrate that the proposed architecture can be used to integrate complex allocation services in business processes. This enables process automation by using powerful operations research techniques. An advantage of our approach is, that it allows a process designer to focus on specifying the requirements on resources, whereas an expert in optimization can focus on how resources are best allocated to process instances. A change of allocation services (e.g., an optimization algorithm, a simple heuristic, machine learning algorithms) is easily possible without adaptation of the respective process model. In the future, historical execution data of resource allocations could also be used for learning and selecting an appropriate allocation service. Currently, we assume that the process execution always follows the allocation plan. This could be extended by considering real-time execution data to adapt the allocation plans automatically.

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

References

1. Abedinnia, H., Glock, C.H., Grosse, E.H., Schneider, M.: Machine scheduling problems in production: a tertiary study. Comput. Ind. Eng. **111**, 403–416 (2017)
2. Ağralı, S., Taşkın, Z.C., Ünal, A.T.: Employee scheduling in service industries with flexible employee availability and demand. Omega **66**, 159–169 (2017)
3. Arias, M., Munoz-Gama, J., Sepúlveda, M.: Towards a taxonomy of human resource allocation criteria. In: Teniente, E., Weidlich, M. (eds.) BPM 2017, vol. 308, pp. 475–483. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_37
4. Arias, M., Rojas, E., Munoz-Gama, J., Sepúlveda, M.: A framework for recommending resource allocation based on process mining. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 458–470. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_37
5. Arias, M., Saavedra, R., Marques, M.R., Munoz-Gama, J., Sepúlveda, M.: Human resource allocation in business process management and process mining: a systematic mapping study. Manag. Decis. **56**(2), 376–405 (2018)
6. Bang-Jensen, J., Gutin, G., Yeo, A.: When the greedy algorithm fails. Discrete Optim. **1**(2), 121–127 (2004)
7. Bellaaj Elloumi, F., Sellami, M., Bhiri, S.: Avoiding resource misallocations in business processes. Concurrency Comput.: Practice Exp. e4888 (0000). https://doi.org/10.1002/cpe.4888
8. Cabanillas, C.: Process-and resource-aware information systems. In: 2016 IEEE 20th International EDOC, pp. 1–10. IEEE (2016)
9. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: RAL: a high-level user-oriented resource assignment language for business processes. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 50–61. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_5
10. Campbell, A.M., Savelsbergh, M.: Efficient insertion heuristics for vehicle routing and scheduling problems. Transp. Sci. **38**(3), 369–378 (2004)
11. Cardoen, B., Demeulemeester, E., Beliën, J.: Operating room planning and scheduling: a literature review. Eur. J. Oper. Res. **201**(3), 921–932 (2010)
12. Coelho, J., Vanhoucke, M.: An exact composite lower bound strategy for the resource-constrained project scheduling problem. Comput. Oper. Res. **93**, 135–150 (2018)
13. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, 2nd edn. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-56509-4
14. Gendreau, M., Potvin, J.Y., et al.: Handbook of Metaheuristics, vol. 2. Springer, Heidelberg (2010). https://doi.org/10.1007/978-1-4419-1665-5
15. Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.: Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies. Eur. J. Oper. Res. **151**(1), 1–11 (2003)
16. Goel, A.: Fleet Telematics - Real-Time Management and Planning of Commercial Vehicle Operations. Operations Research/Computer Science Interfaces, vol. 40. Springer, Heidelberg (2007). https://doi.org/10.1007/978-0-387-75105-4
17. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. Eur. J. Oper. Res. **207**(1), 1–14 (2010)

18. Havur, G., Cabanillas, C., Mendling, J., Polleres, A.: Resource allocation with dependencies in business process management systems. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_1
19. Herzberg, N., Meyer, A., Weske, M.: An event processing platform for business process management. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference, pp. 107–116. IEEE (2013)
20. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 38–54. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_3
21. Huang, Z., van der Aalst, W.M., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. Data Knowl. Eng. **70**(1), 127–145 (2011)
22. Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C.: MILP formulations for single- and multi-mode resource-constrained project scheduling problems. Comput. Chem. Eng. **36**, 369–385 (2012)
23. Lenstra, J.K., Kan, A.R.: Computational complexity of discrete optimization problems. Ann. Discrete Math. **4**, 121–140 (1979)
24. Liu, T., Cheng, Y., Ni, Z.: Mining event logs to support workflow resource allocation. Knowl.-Based Syst. **35**, 320–331 (2012)
25. Liu, Y., Wang, J., Yang, Y., Sun, J.: A semi-automatic approach for workflow staff assignment. Comput. Ind. **59**(5), 463–476 (2008)
26. May, J.H., Spangler, W.E., Strum, D.P., Vargas, L.G.: The surgical scheduling problem: current research and future opportunities. Prod. Oper. Manag. **20**(3), 392–405 (2011)
27. Oberweis, A.: A meta-model based approach to the description of resources and skills. In: AMCIS 2010 (2010)
28. OMG: Notation BPMN version 2.0. OMG Specification, Object Management Group, pp. 22–31 (2011)
29. Ouyang, C., Wynn, M.T., Fidge, C., ter Hofstede, A.H., Kuhr, J.C.: Modelling complex resource requirements in business process management systems. In: ACIS 2010 Proceedings (2010)
30. Pellerin, R., Perrier, N., Berthaut, F.: A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. Eur. J. Oper. Res. (2019). https://doi.org/10.1016/j.ejor.2019.01.063
31. Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. Eur. J. Oper. Res. **225**(1), 1–11 (2013)
32. Pufahl, L., Weske, M.: Batch activities in process modeling and execution. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 283–297. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_20
33. Reijers, H.A., Jansen-Vullers, M.H., zur Muehlen, M., Appl, W.: Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 125–140. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_10
34. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005). https://doi.org/10.1007/11431855_16

35. Savelsbergh, M., Van Woensel, T.: 50th anniversary invited article-city logistics: challenges and opportunities. Transp. Sci. **50**(2), 579–590 (2016). https://doi.org/10.1287/trsc.2016.0675
36. Senkul, P., Toroslu, I.H.: An architecture for workflow scheduling under resource allocation constraints. Inf. Syst. **30**(5), 399–422 (2005)
37. Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, no. 18. SIAM, Philadelphia (2014)
38. Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd edn. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28616-2
39. Zhao, W., Liu, H., Dai, W., Ma, J.: An entropy-based clustering ensemble method to support resource allocation in business process management. Knowl. Inf. Syst. **48**(2), 305–330 (2016)