The final authenticated version is available online at https://doi.org/10.1007/978-3-031-46846-9_23

Resource-Driven Process Manipulation: Modeling Concepts and Valid Allocations

Felix Schumann[0009−0006−5431−1922], Stefanie Rinderle-Ma[0000−0001−5656−6108]

Technical University of Munich, Germany; TUM School of Computation, Information, and Technology
{felix.schumann, stefanie.rinderle-ma}@tum.de

Abstract. In situations of scarce resource availability, flexibility on which resources execute which tasks is key to process and system performance. Tightly coupled control flow and resource modeling hampers flexible resource allocation. Hence, in this work, we propose resource-driven process manipulation (RDPM) to enable the separation between the business and resource requirements for a process. RDPM enables process modelers to specify resource-specific requirements for the control flow as part of resource profiles, e.g., a machine (resource) requires configuration (task) before execution. Moreover, the resource is promoted to a first-class citizen in process-aware information systems and enabled to impact the execution. The basic concepts of RDPM are defined and an algorithm is provided to find valid resource allocations for a task. The approach is prototypically implemented and compared to existing modeling approaches w.r.t. complexity for the modeler and process participant.

Keywords: Process-Aware Information Systems, Resource Perspective, Resource Modeling, Process Changes, Resource Allocation

1 Introduction

"The resources of an organization are valuable assets, often cost-intensive, and limited" [7]. Especially in cooperative environments, processes and their instances utilize resources in a shared manner [8,9]. Hence, modeling, assigning, and allocating resources to process tasks is of utmost importance. Although existing approaches address the resource perspective of processes, often resources and their assignments are only modeled as part of the control flow, e.g., as swimlanes in BPMN. This "mixed modeling" approach results in unclear effects, especially when resources change. Moreover, process and resource owner are not necessarily the same person, hampering the modeling of both views and their requirements (e.g., a resource demanding for an extra step or temporal requirement).

Hence, we propose the *separation of concerns* between control flow and resource requirements in process-aware information systems (PAIS) as depicted in Fig. 1, i.e., to model resources as objects at design time, which can manipulate the control flow of an instance at run-time to integrate its own process requirements into it. With this resource-driven process manipulation (RDPM)

approach, the process models are simplified, task modeling becomes easier for process and resource owners, and resources can be flexibly added/deleted and linked to the process. The latter particularly contributes to the adaptability of processes to varying resource availability.

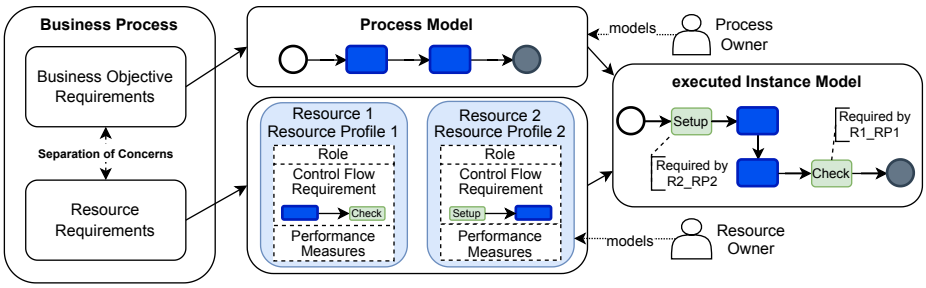

Fig. 1: Separation of process and resource requirements in RDPM

The core modeling concept of RDPM is the resource profile as depicted in Fig. 1 specifying resources, their linkage to the processes, and the manipulation of process instances required by this resource. Manipulation of process instances must not result in undesired side effects by introducing correctness issues into the instances. Hence, we have to provide means to check the validity of resource allocation in RDPM based on allocation trees.

In Sections 2 and 3, we will introduce the basic concepts of RDPM and an algorithm for valid allocations. In Sect. 4 we present a first prototype, identified use cases for RDPM and a visual representation on how RDPM simplifies the models. Sect. 5 presents related work, Sect. 6 discusses the approach and gives an outline of further planned development of the approach

2 Modeling Concepts of RDPM

The core concept of RDPM is the *resource profile*. In order to enable the separation of control flow and resource requirements, a resource profile contains all necessary information on the resource plus the linkage to the process and how the resource will manipulate the process instances it is allocated to. Equation 1 defines a resource profile rp as follows: it contains the resource r and its role $role$. Moreover, task $task$ that the resource can perform is specified. Additional attributes $Attr$ can be specified to, e.g., measure performance. Finally, the manipulation of a process instance, the resource is allocated to, is captured via a set of change patterns CP.

$$rp := (res, role, task, Attr, CP) \tag{1}$$

Change patterns have been chosen as they are a well-defined concept for the evolution and adaptation of processes and process instances [18]. In this work,

we focus on the insert, replace and delete pattern w.r.t. tasks. In the following, we will provide the RDPM change patterns which are either adapted from the change patterns in [18] (insert) or can be directly used as defined in [18] (replace, delete). Note that inserting, deleting, and replacing tasks require an adaptation of the control dependencies, e.g., to embed a newly inserted task into the control flow. For these adaptations, we rely on the formal semantics of the insert, replace, and delete patterns as defined in [14].

- **Insert:** As defined in [18], the *insert* pattern $Insert(S, X, A, B) \mapsto S'$ inserts new task X into process model S between activity sets A and B. It can be specified whether X is inserted serially or in parallel. For RDPM, we adapt the insert position of task X in relation to task $task$ as specified in the resource profile rp of $resource$, i.e., X can be inserted $before|after|parallel$ w.r.t. $task$. Overall, the insert pattern for RDPM is defined as $op_{RDPM} := Insert(S, X, before|after|parallel)$.
- **Replace:** $op_{RDPM} = Replace(S, X, Y) \mapsto S'$ is concerned with the replacement of a task with a new task. Although this operation can be expressed by delete and insert patterns, we follow the argumentation of [18] that the higher level of abstraction is favorable for users.
- **Delete:** The change pattern $op_{RDPM} = Delete(S, X) \mapsto S'$ deletes task X from process model S.

3 Finding Valid Resource Allocations

In general, when exposing a process model to change patterns, the risk of undermining the soundness of the model arises [18]. The previously refined change operations of RDPM (cf. Sect. 2) might not only affect control flow soundness, but also the validity of resource allocation and of the data objects.

Guarantee of soundness and executability of the instance process model: RDPM manipulates the instance model of a process and might affect the successful execution of this instance. The successful execution depends on the structural and behavioral soundness of the instance process model. Structural soundness depends on the meta model and behavioral soundness demands for being able to reach desired final states and the absence of tasks that can never be executed (cf., e.g., [2]). Note that we assume the process model of interest to be structurally and behaviorally sound before instance manipulation takes place.

As process meta model, we rely on refined process structure trees (RPST) [17] as the RPST model of structuring processes in fragments enables to check the soundness of each fragment independently. Each fragment follows the single entry point, single endpoint (SESE) structure. The leaves of the tree represent the tasks. The RPST structure ensures the soundness of a node, iff all subfragments of the node are also sound SESE fragments. For more description of the RPST and how to compose other representations see [17,11]. Thus the soundness of the resulting process model after the application of a change pattern is sound, iff the original process model and the change patterns are sound.

Validity of resource allocation: An approach that deals with resources having a direct impact on the execution of a process instance requires checking that the execution is possible from the point of view of the resources. For RDPM, soundness from this perspective means that a change pattern can only be applied if there exist resources so that all tasks of the resulting instance model can be successfully allocated. Thus after allocating the first task and applying the connected change patterns, it must be checked if newly inserted tasks can also be allocated. For finding valid allocations, an *allocation tree* is built with the task to be allocated as its root node (cf. Algorithm 1). The tree consists of task nodes, and resource nodes and is created in a recursive way by building one branch after the other.

Algorithm 1: build_allocation_tree

```
input  : root: task_node to allocate, ar: available resources, ex_tasks:
         excluded tasks, task_parent, res_parent
output: root: tree with all valid allocation branches for one task
for resource in ar do
    for profile in resource.resourceprofiles do
        if root.label = profile.task and root.role = profile.role then
            root.add_child(resource) ;          //Add resources as Children
        end
    end
end
if not root.children then
    delete task_nodes with current resource profile from res_parent
    return root ;                              //Prune Branch without resources
end
for resource in root.children do
    if resource.profile.change_patterns then
        for cp in profile.change_patterns do
            tasks ← cp.get_tasks
            if any task of tasks in ex_tasks then
                delete resource from root.children
                break ;                                          //Prevent cycle
            end
            for task in tasks do
                task_parent ← root
                res_parent ← resource
                resource.add_child(build_allocation_tree(task, ar, ex_tasks,
                  task_parent, res_parent)) ;      //Recursive function call
            end
        end
        return root
    end
end
```

The children of each node must be of the opposite type than its parent. As depicted in Fig. 2, a task node's children are all resource profiles with a role

that is part of the task's authorized roles (Role Based Access). Each of these resource nodes has the tasks defined in its change patterns as its children. As input, the algorithm requires the task that needs a resource allocated to it and the resources that are available for allocation. Excluded tasks, task parent, and resource parent are optional parameters that are needed for the recursive call of the algorithm. In order to prevent a cyclic allocation, a task that is already part of a branch can not become part of the same branch again.

While building the tree, the validity of each branch is checked as follows: For a branch to be valid, every task (main task and tasks introduced by change patterns) needs an authorized resource allocated to it. Therefore, a branch only represents a valid allocation if all its leaves are of type resource. A branch that ends with a task as leaf (invalid allocation) can be pruned back until a task node with a valid allocation is found (see Fig. 2, dashed box).

The pruning based on resource validity leads to a minimal tree of valid branches. Once the whole tree is built, it is used to find the optimal branch for the allocation. Each branch of the tree represents one valid allocation and can be compared to the other branches based on the attributes defined in the resource profiles. Since the role, performance attributes and change patterns are part of the resource profile, the best branch for the set objective can be identified.

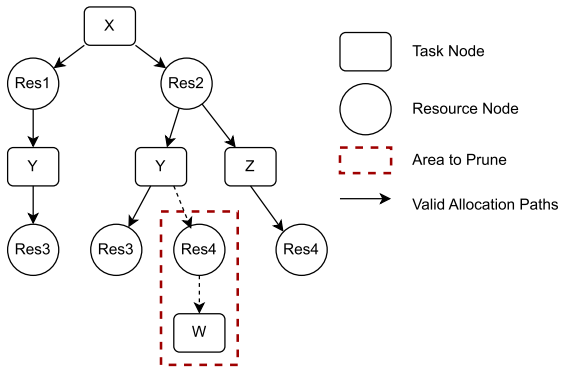

Fig. 2: Allocation tree, two valid allocation branches, one invalid branch to prune.

Validity towards data objects: It is important to mention that also the data perspective can lead to invalid allocations. During the execution of a process, data objects are created, updated, and consumed. These data objects can be created by tasks that are deleted through RDPM or tasks inserted by RDPM need to consume data objects that are not yet created. Developing a concept of ensuring the validity of data objects poses an important task for future work, but is not in the scope of this paper.

4 Evaluation

The prototypical implementation of the RDPM approach is embedded in a service-oriented architecture. As evaluation, we identify fitting use cases where

RDPM will improve quality and simplicity of the model, followed by a comparison of existing process and resource modeling approaches for a use case.

Prototypical Implementation: To realize RDPM, we propose to offer the allocation of tasks as an external service to the PAIS. Figure 3 shows how the RDPM service links the PAIS to a Resource Management System.

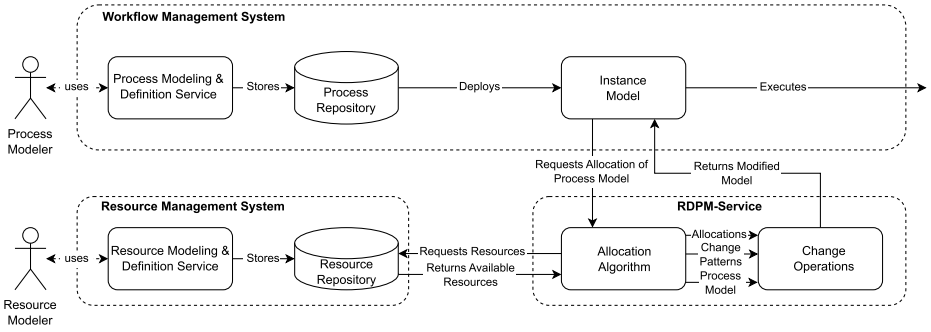

Fig. 3: Description of the system with RPDM as external service.

This service-oriented architecture enables a high level of flexibility towards the resource allocation task while realizing the separation between the business objective and the resource management as intended by RDPM by design.

For the implementation, we used the Cloud Process Execution Engine (CPEE)[1] as service-oriented PAIS. As resource repository, a description of the available resources, with resource profiles and change patterns, is given in an XML file. The creation of a complete Resource Management System is not in the scope of this work, but is considered for future development.

The allocation request is designed as a specific task in the CPEE at the beginning of the process model to call the RDPM service. Figure 4 shows this design in the CPEE. The "measure" and "operator" arguments set the objective for the best branch. A prototype with demo use cases is available.[2]

To generate the modified model, change operations are derived from the change patterns and applied to the model (cf. [3]). The modified instance model is returned to the CPEE and the execution of the instance is started. The returned instance model is a CPEE-Tree which is based on the RPST [10]. This modified CPEE-Tree is now augmented with the allocated resources.

Use cases for RDPM: RDPM enables modelers to model a relation between resources and their process requirements. This feature is also characteristic of the field of Operations Research (OR). We identified that RDPM helps to simplify the transition from OR problems, which are concerned with improving and planning operations, to BPM, which is concerned with controlling and enacting

[1] cpee.org/

[2] https://github.com/Schlixmann/RDPM

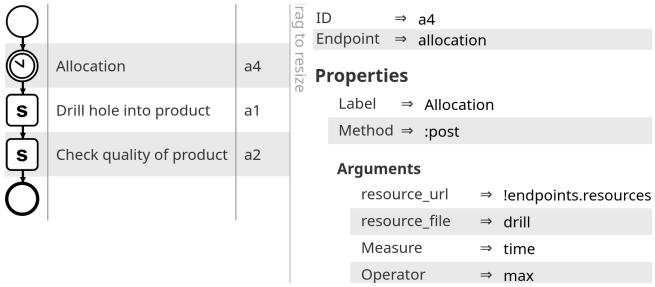

Fig. 4: Description of the allocation task and its connected parameters.

these operations in a real-world setting [1,5]. Therefore we argue that RDPM is an important step for BPM to become more proactive as demanded in [13].

From the field of OR problems, we identified the order batching problem, the flexible job shop scheduling problem and the parallel machine scheduling problem as problems that are well supported by RDPM. At the current state of RDPM, the parallel machine scheduling problem is best supported.

Comparison with Existing Modeling Approaches: Consider the following use case (cf. Fig. 5): In order to execute the task "drilling", the machine needs to be configured for "drilling". The task "configure machine for drilling" has to be inserted as predecessor (cf. setup time in yarn dyeing machines in [6]). In addition, assume that after drilling the hole, a quality check of the product is needed. Task "Check quality of product" can be done by three resources: manually by a technician or an inexperienced student technician, and automated by a 3D Measurement machine. The machine needs to be leveled before it can perform the measuring and the student needs supervision while checking the product. Figure 5(a) shows the use case modeled in traditional fashion.

The expectation of RDPM is a "clean" process model and an executable instance model. When using RDPM only the business objective must be modeled in the process model. Figure 5(b) shows this business objective-focused process. After applying RDPM to an instance of the process model, the allocated instance model of the use case is shown in Fig. 5(c) and is ready for execution.

Figure 5 shows that RDPM simplifies the process model as well as the instance model. Modelers and domain experts can profit from this visualization. Modeling multiple variants of a process is an important research field in BPM, where variants usually depend on data objects rather than the used resources. Process families (cf. [4]) model every possible trace in a separate process model and categorize these models into families. Another common approach is the definition of sub-processes in combination with late binding [18]. Both approaches lead to highly complex (sub-) process repositories. RDPM's advantage over these approaches is the simplicity of change patterns and the easy adjustment of these change patterns as part of the resource profile.

Resource-driven process manipulation generates a clean and easy-to-understand process model. After the allocation, the instance model is shown in a fully flat-

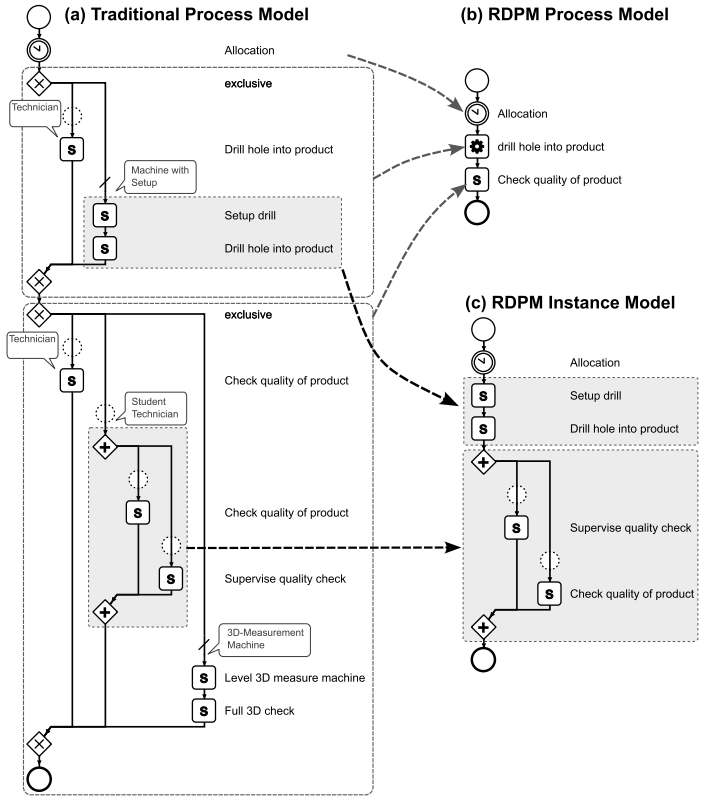

Fig. 5: (a) Process model designed traditionally for all resources; (b) process model designed following RDPM focused on the business objective of the process; (c) the instance model shows what will actually be executed.

tened way and can help users to understand the upcoming tasks. The flattened structure is considered more easily understandable [16]. If an adjustment of a change pattern is done, it is automatically applied to all related processes and thus the integrity of process models is ensured.

5 Related Work

Many approaches deal with flexibility or the usage of resources in BPM. The description of resource profiles was originally used to mine multiple attributes of resources in a process mining setting in [12]. In terms of combining processes based on their features, [4] describes the use of process families to enable multiple different process options based on one process template. A larger review on variability and change modeling in BPM is given in [15] comparing different approaches to model variability into process models.

The authors of [8] discuss the requirements of an integrated view on process and data. The study finds that neither imperative nor declarative approaches fulfill the requirements of such an integrated view, while data-driven approaches are not mature enough. [19] provides more advanced allocation approaches for human resources by utilizing process mining to optimize resource allocation with the process costs as the optimization objective. We argue that RDPM is an enabler to introduce such optimization algorithms to a more proactive BPM.

6 Discussion and Outlook

The proposed Resource-Driven Process Manipulation (RDPM) approach equips resources with the chance to alter the control flow of a process instance. This way a separation of concerns between the process requirements which serve the business objective and the process requirements which serve the underlying resource infrastructure can be achieved. Using the RDPM modeling approach, control flow and resources become closer connected at execution, while a more precise line can be drawn in modeling them.

While a strength of this approach is the focus on changing the instance model and executing it, one could argue that the process model itself does not represent all process options. Regarding the change patterns, the current implementation is not yet able to realize the change pattern concerned with deletion. To do so, we identified that dependencies are needed to further specify when a change pattern should be applied, e.g., only if the previous task is allocated to the same resource. The replace pattern in the current implementation is only applied to the task open for allocation, with the implementation of the delete pattern this will be enhanced to the full process. Lastly, the validity in terms of the data objects is not yet guaranteed. In future work, we plan to implement the delete pattern and to add the dependencies as a first step towards the data flow perspective.

Considering that one of RDPM's strengths is the identification of non-business-objective related processing times, the generated data can help to develop and integrate optimization algorithms, which realize an optimized resource allocation at run-time as described in [19]. To achieve this functionality, execution times or process performance could be predicted for the valid branches of the allocation tree. RDPM will help to enhance the flexibilty of PAIS w.r.t. resource allocation.

Acknowledgements: This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project number 277991500

References

1. van der Aalst, W.M.P.: Business process management: A comprehensive survey **2013**, 1–37 (2013). https://doi.org/10.1155/2013/507984
2. van Dongen, B.F., Mendling, J., van der Aalst, W.M.P.: Structural patterns for soundness of business process models. In: Enterprise Distributed Object Computing Conference. pp. 116–128 (2006). https://doi.org/10.1109/EDOC.2006.56

3. Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Inf. Syst. **49**, 1–24 (2015). https://doi.org/10.1016/j.is.2014.10.004
4. Gröner, G., Boskovic, M., Parreiras, F.S., Gasevic, D.: Modeling and validation of business process families. Inf. Syst. **38**(5), 709–726 (2013). https://doi.org/10.1016/j.is.2012.11.010
5. Hillier, F., Lieberman, G.: Introduction to Operations Research. McGraw-Hill Education (2021)
6. Hsu, H., Hsiung, Y., Chen, Y., Wu, M.: A GA methodology for the scheduling of yarn-dyed textile production. Expert Syst. Appl. **36**(10), 12095–12103 (2009). https://doi.org/10.1016/j.eswa.2009.04.075
7. Ihde, S., Pufahl, L., Völker, M., Goel, A., Weske, M.: A framework for modeling and executing task-specific resource allocations in business processes. Computing **104**(11), 2405–2429 (2022). https://doi.org/10.1007/s00607-022-01093-2
8. Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: Fundamental requirements and their support in existing approaches. Int. J. Inf. Syst. Model. Des. **2**(2), 19–46 (2011). https://doi.org/10.4018/jismd.2011040102
9. Leitner, M., Rinderle-Ma, S.: A systematic review on security in process-aware information systems - constitution, challenges, and future directions. Inf. Softw. Techn. **56**(3), 273–293 (2014). https://doi.org/10.1016/j.infsof.2013.12.004
10. Mangler, J., Rinderle-Ma, S.: Cloud process execution engine: Architecture and interfaces. https://doi.org/10.48550/arXiv.2208.12214
11. Munoz-Gama, J., Carmona, J., Van Der Aalst, W.M.P.: Conformance Checking in the Large: Partitioning and Topology. In: Business Process Management, pp. 130–145 (2013). https://doi.org/10.1007/978-3-642-40176-3_11
12. Pika, A., Leyer, M., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Mining resource profiles from event logs. ACM Trans. Manag. Inf. Syst. **8**(1), 1:1–1:30 (2017). https://doi.org/10.1145/3041218
13. Poll, R., Polyvyanyy, A., Rosemann, M., Röglinger, M., Rupprecht, L.: Process forecasting: Towards proactive business process management. In: Business Process Management. pp. 496–512 (2018). https://doi.org/10.1007/978-3-319-98648-7_29
14. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Conceptual Modeling. pp. 279–293 (2008). https://doi.org/10.1007/978-3-540-87877-3_21
15. Rosa, M.L., Aalst, W.M.P.V.D., Dumas, M., Milani, F.P.: Business Process Variability Modeling: A Survey. ACM Computing Surveys **50**(1), 2:1–2:45 (2017). https://doi.org/10.1145/3041957
16. Türetken, O., Dikici, A., Vanderfeesten, I.T.P., Rompen, T., Demirörs, O.: The influence of using collapsed sub-processes and groups on the understandability of business process models. Bus. Inf. Syst. Eng. **62**(2), 121–141 (2020). https://doi.org/10.1007/s12599-019-00577-4
17. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. **68**(9), 793–818 (2009). https://doi.org/10.1016/j.datak.2009.02.015
18. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66**(3), 438–466 (2008). https://doi.org/10.1016/j.datak.2008.05.001
19. Zhao, W., Yang, L., Liu, H., Wu, R.: The optimization of resource allocation based on process mining. In: Advanced Intelligent Computing Theories and Applications ICIC. pp. 341–353 (2015). https://doi.org/10.1007/978-3-319-22053-6_38