Optimizing Resource Utilization by Combining Activities Across Process Instances

Christine Natschläger[1(✉)], Andreas Bögl[2], Verena Geist[1], and Miklós Biró[1]

[1] Software Competence Center Hagenberg GmbH, Hagenberg Im Mühlkreis, Austria
{christine.natschlaeger,miklos.biro,verena.geist}@scch.at
[2] Pascom Kommunikationssysteme GmbH, Arbing, Austria
andreas.boegl@pascom.at

Abstract. Resource-efficient business processes are a key asset of an organization in a competitive market environment. Current efforts address this issue either at the process schema level by specifying an optimal sequence of process activities or at the process instance level by optimizing resource utilization within a single running process instance.

In this paper, we present a novel approach for combining activities across process instances to optimize resource utilization. The proposed approach comprises the (i) definition of business processes and combinable activities, (ii) identification of possible combinations at runtime, (iii) determination of optimization potential, and (iv) actual combination of activities across process instances. The approach further supports partial activity combinations and dynamic resource selection for combined activities. The applicability of the approach is demonstrated by a case study.

1 Introduction

Efficient business processes are a key asset for a company's success and are addressed by the discipline *Business Process Management* (BPM), which comprises the definition, implementation, control, and improvement of business processes. In this paper, we present the *Combined-Instance Approach*, which improves resource utilization in business processes by combining activities across running processes instances. It thereby considers different resource types including physical, human, and financial resources. Optimization potential is given if the capacity of a resource is (or can be) higher than required by the actual process instance, so that activities from further process instances can be integrated, thereby saving their setup and/or execution costs. With the proposed approach, we address the goals of the Industry 4.0 project of the German government, which emphasizes the demand for adaptable processes and resource efficiency in

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH. This publication has been written within the project *AdaBPM* (number 842437), which is funded by the *Austrian Research Promotion Agency* (FFG).

R.V. O'Connor et al. (Eds.): EuroSPI 2015, CCIS 543, pp. 155–167, 2015.
DOI: 10.1007/978-3-319-24647-5_13

traditional industries. Some preliminary ideas were presented at a workshop on resource management in service-oriented computing (see [12]).

2 The Combined-Instance Approach

The *Combined-Instance Approach* combines activities of different process instances to optimize resource utilization and consists of the following four steps: (1) definition of business processes with data objects and constraints, (2) identification of candidates for process instance combinations, (3) determination of an optimization potential, and (4) actual combination of business process instances. An overview of the four steps applied to a running example is shown in Fig. 1. The running example relates to an order execution process of a sand and fertilizer producer in Upper Austria, called S&F company for short. The company produces and delivers several thousand tons of sand and fertilizer products to various destinations in Europe, where they are mainly used for cultivating sport fields, golf courses and for producing other final products in the cement industry.

2.1 Business Processes with Data Objects and Constraints

The first step of the combined-instance approach is to extend the business process with additional definitions required for possible instance combinations. The basis for all process instances is the *business process schema* ($\mathcal{S}$), which provides the process specification including data objects and resources. For the combined-instance approach, $\mathcal{S}$ is extended with (i) *meta-information* specifying possible resources and their capacities, (ii) *type-level constraints* defining general restrictions that apply to all instances either based on given data or being specified manually, and (iii) *combinable activities* ($\mathcal{C}$) comprising a *combinable condition* (*cc*) that defines the required matching of two activity instances for a possible combination and an *optimization function* (*of*) that determines the optimization potential of a combination. The *cc* and the *of* must be specified individually for every combinable activity and the combinable activity is then marked with a 'C' in the process diagram. All constraints must be formally specified (e.g., based on the *Object Constraint Language* (OCL)) to support business process execution.

In our running example shown in Fig. 1, $\mathcal{S}$ defines an order execution process comprising activities for order handling, production, delivery, and invoicing with corresponding data objects (e.g., order, product, and invoice) and resources (e.g., production and transportation means). Due to space limitations not all of them are shown in Fig. 1. $\mathcal{S}$ is then extended with (i) *meta-information* like possible transportation means and their capacity, (ii) *type-level constraints* like the capacity of a resource (based on data) or that a ship can only be used if both departure and destination point have a harbor and are connected by a river (manually specified constraint), and (iii) *combinable activities*. In our example, the production and delivery activity specify a *cc* and an *of* (see Section 2.3 for *of*), so these two activities are combinable ($\mathcal{C}$). The *cc* of the production activity specifies that two production activities are combinable, if the product types

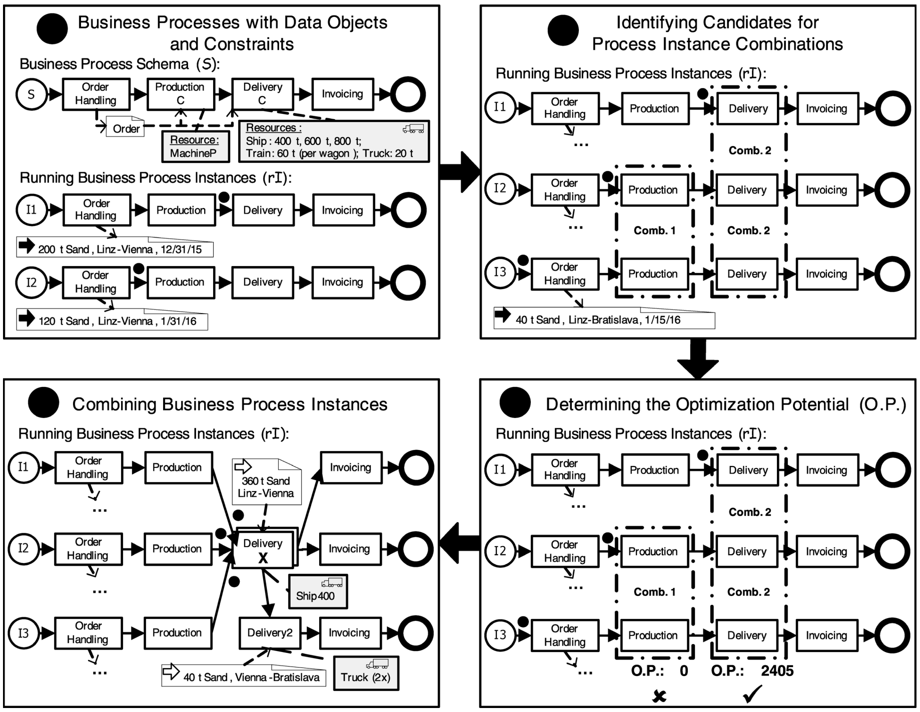
Business Processes with Data Objects and Constraints
Business Process Schema (S):
Order Handling
Production C
Delivery C
Invoicing
Order
Resource: MachineP
Resources: Ship: 400 t, 600 t, 800 t; Train: 60 t (per wagon); Truck: 20 t
Running Business Process Instances (rI):
200 t Sand, Linz-Vienna, 12/31/15
120 t Sand, Linz-Vienna, 1/31/16
Identifying Candidates for Process Instance Combinations
Comb. 1
Comb. 2
40 t Sand, Linz-Bratislava, 1/15/16
Determining the Optimization Potential (O.P.)
O.P.: 0
O.P.: 2405
Combining Business Process Instances
360 t Sand Linz-Vienna
Delivery X
Ship400
Delivery2
40 t Sand, Vienna-Bratislava
Truck (2x)

Fig. 1. Combined-Instance Approach

are the same (`ProductType1 = ProductType2`). The *cc* of the delivery activity defines that the routes must be overlapping or, more specifically, that it is shorter to go from the departure point (`dep`) to one destination (`dest`) and then to the other than starting twice from the departure point (`min(distance(dep, dest1) + distance(dest1, dest2), distance(dep, dest2) + distance(dest2, dest1)) < distance(dep, dest1) + distance(dep, dest2)`).

An instantiation of $\mathcal{S}$ is called a business process instance $\mathcal{I}$. During execution of $\mathcal{I}$, data objects and corresponding constraints defined in $\mathcal{S}$ are instantiated and provide concrete instance-specific data and restrictions. No additional constraints are expected on the instance-level, since process execution must be kept simple for end-users. A running business process instance $r\mathcal{I}$ indicates that $\mathcal{I}$ is active and comprises one or more tokens that mark the current position(s) in the process flow (shown by a black-filled circle in Fig. 1).

In our running example, the order execution process is instantiated three times ($\mathcal{I}1$, $\mathcal{I}2$, and $\mathcal{I}3$). The first process instance comprises an order of 200 t of sand that must be sent from Linz to Vienna until 12/31/15. The order of $\mathcal{I}2$ has the same route but with 120 t of sand that must be delivered until 01/31/16. Finally, 40 t of sand are ordered in $\mathcal{I}3$ (see step 2) and must be sent to Bratislava until 01/15/16. All three instances are active, but the current positions differ.

We, thus, have the following special resource-constrained scheduling problem:

- Activities $\mathcal{I}i_{OrderHandling}$, $\mathcal{I}i_{Production}$, $\mathcal{I}i_{Delivery}$, $\mathcal{I}i_{Invoicing}$, $i = 1, 2, 3$ with processing times $\mathrm{p}i_{Activity}$, earliest release times $\mathrm{e}i_{Activity}$ given by the tokens, and deadlines $\mathrm{d}i_{Delivery}$ (only for delivery activities).
- Resources $k = 1, ..., n$ with R_k units of resource k available (capacity), in our case $k = MachineP, Ship400, Ship600, Ship800, TrainWagon, Truck$ with $R_{MachineP} = unlimited$, $R_{Ship400} = 400\mathrm{t}$, $R_{Ship600} = 600\mathrm{t}$, etc.
- Activity x occupies $r_{x,k}$ units of resource k (demand) for $k = 1, ..., n$ during processing, e.g., if x is $\mathcal{I}1_{Delivery}$ then x occupies $r_{x,Ship400} = 200\mathrm{t}$.
- Precedence constraints $m \rightarrow n$ only between the activities $\mathcal{I}i_{OrderHandling}$, $\mathcal{I}i_{Production}$, $\mathcal{I}i_{Delivery}$, $\mathcal{I}i_{Invoicing}$ of the same process instance $\mathcal{I}i$.

Finally, we require some auxiliary functions that return all $r\mathcal{I}$, the current token position(s) of $r\mathcal{I}$, the state of $\mathcal{C}$ (open, active, or completed), whether $\mathcal{C}$ is still reachable, and the expected costs and execution time of activities.

2.2 Identifying Candidates for Process Instance Combinations

The second step of the combined-instance approach receives the extended business process schema $\mathcal{S}$ and all running process instances (set of $r\mathcal{I}$). The goal then is to identify two combinable instances of $\mathcal{C}$ that satisfy all constraints and can be called a candidate pair. The search for candidate pairs is initiated whenever a token reaches an instance of $\mathcal{C}$ and the triggering activity is compared with the corresponding activity of every other $r\mathcal{I}$. If the other activity is open and reachable and all constraints and the *cc* are satisfiable, then the two instances of $\mathcal{C}$ are a candidate pair. Several candidate pairs may be combined to sets of higher cardinality. In subsequent research, we intend to also consider future process instances, i.e. the current process instance waits for further combination possibilities for as long as possible while not violating any deadlines or until the maximum capacity of the resource is reached.

In our running example, the token of $\mathcal{I}2$ reaches the production activity and triggers a search. A combination with $\mathcal{I}1_{Production}$ is not possible, since the activity already completed. However, $\mathcal{I}3_{Production}$ is still open, reachable and all constraints are satisfiable. The *cc* is also fulfilled, since the product type (sand) is the same. Thus, $\mathcal{I}2_{Production}$ & $\mathcal{I}3_{Production}$ are a candidate pair. In addition, the token of $\mathcal{I}1$ reaches the delivery activity and triggers a further search. In this case, $\mathcal{I}2_{Delivery}$ and $\mathcal{I}3_{Delivery}$ are still open, reachable and all constraints are satisfiable. The *cc* of $\mathcal{I}1_{Delivery}$ and $\mathcal{I}2_{Delivery}$ is obviously fulfilled, since both activities have the same route. Furthermore, the *cc* of $\mathcal{I}1_{Delivery}$ and $\mathcal{I}3_{Delivery}$ is also satisfied, since the distance from Linz to Vienna to Bratislava (approx. 265 km) is shorter than the distance from Linz to Vienna (approx. 185 km) plus the distance from Linz to Bratislava (approx. 265 km). Thus, we identify two candidate pairs $\mathcal{I}1_{Delivery}$ & $\mathcal{I}2_{Delivery}$ and $\mathcal{I}1_{Delivery}$ & $\mathcal{I}3_{Delivery}$ which can, after re-evaluating the *cc*, be combined to $\mathcal{I}1_{Delivery}$ & $\mathcal{I}2_{Delivery}$ & $\mathcal{I}3_{Delivery}$.

Whether identified candidate pairs can actually be combined depends, however, on three further conditions:

1. if the other (not-triggering) activity ever receives a token (after a preceding split an alternative path may be taken).
2. if waiting for the other (not-triggering) activity does not violate a deadline.
3. if the combination achieves improvement (optimization potential). This condition is evaluated within the next step.

2.3 Determining the Optimization Potential

The third step of the combined-instance approach receives a set of combinable candidates. The goal then is to identify possible combinations and whether they are economically worthwhile by applying the *optimization function* (*of*) defined for every $\mathcal{C}$. The *of* calculates a value for comparison (e.g., costs, time) with a predefined optimum (lowest/highest value). In some cases, possible candidates satisfy the *cc*, but concrete tasks differ, so that activities cannot be fully combined, e.g., due to routes being overlapping but not identical. To cope with this issue, the combined-instance approach provides the possibility to split activities so that part of it can be combined and the remaining part is executed individually (partial activity combination). The method for splitting activities can also be applied to optimally use resources with respect to their maximum capacity (i.e. activities are split to fill resources). If activities are split, the optimization function has to consider all sub-activities.

So the main goal of the third step is to compare the separate execution of activities with possible combined executions, in the following called separate and combined solutions. Considering the separate solution, all activities are defined by $\mathcal{S}$ and receive instance specific data and constraints. The *of* then calculates a value for comparison for the separate solution that reflects the main goal of the optimization, e.g., costs, time, or quality indicators. The calculation is based on a predefined formula and a list of given values.

Considering combined solutions, it is first necessary to determine all possible combinations of identified candidates (also taking activity splits into account). If predefined resources of the separate solution do not fit for a combination, then either a fitting resource is identified (dynamic resource selection) or the combination is disregarded. The next step is to apply the *of* and calculate a value for each combination, thereby using the same value type and calculation process as defined for the separate solution. The values of the combined solutions are then compared with that of the separate solution to identify all combined solutions with an optimization potential. The combination with the highest potential is ranked first; however, if several expected combinations initiated by different activities are overlapping, then an overall optimal solution must be identified.

In our running example, there is only one resource ($MachineP$) with no capacity limitation available for the production activity, which is taken by separate and combined solutions. To calculate a value for comparison, we define the formula: `productionCosts = productAmount x costsPerUnit(productAmount)` and a list of costs: producing 1 t sand costs €3; 20 t sand costs €2.9 (per t); 200 t sand costs €2.5 (per t); and 1000 t sand costs €2 (per t). So the calculated production costs for $\mathcal{I}2_{Production}$ (120 t) are €348 and for $\mathcal{I}3_{Production}$ (40 t) €116 (sum: €464). Considering possible combinations, both production activities can fully be combined so that 160 t of sand can be produced at once. However, the resulting production costs are again €464, so no optimization potential is given by combining $\mathcal{I}2_{Production}$ & $\mathcal{I}3_{Production}$.

Considering transportation means, all possible resources are defined in $\mathcal{S}$ and shown in Fig. 1. Not explicitly given are the costs of a resource, but we assume that $Ship400$ costs €10/km, $Ship600$ €14/km, $Ship800$ €17/km, $TrainWagon$ €4.5/km, and $Truck$ €2/km. The formula is then: `transportationCosts = resourceCosts x distance`. So for the separate solution, the optimal means of transportation for $\mathcal{I}1_{Delivery}$ is $Ship400$ (€10 x 185 km = €1850), for $\mathcal{I}2_{Delivery}$ two $TrainWagon$ (2 x €4.5 x 185 km = €1665), and for $\mathcal{I}3_{Delivery}$ two $Truck$ (2 x €2 x 265 km = €1060) (sum: €4575). For the combined solution, different combinations of candidates are possible:

1. $\mathcal{I}1_{Delivery}$ & $\mathcal{I}2_{Delivery}$ (resource for combination: $Ship400$, costs of combination: €1850, optimization potential: €1665),
2. $\mathcal{I}1_{Delivery}$ & part of $\mathcal{I}3_{Delivery}$ (from Linz-Vienna) (resource for combination: $Ship400$, costs of combination: €1850 + €320 (remaining part of $\mathcal{I}3_{Delivery}$: two $Truck$ from Vienna-Bratislava, 80 km), optimization potential: €740),
3. $\mathcal{I}2_{Delivery}$ & part of $\mathcal{I}3_{Delivery}$ (from Linz-Vienna) (resource for combination: $Ship400$, costs of combination: €1850 + €320 (remaining part of $\mathcal{I}3_{Delivery}$), optimization potential: €555), and
4. $\mathcal{I}1_{Delivery}$ & $\mathcal{I}2_{Delivery}$ & part of $\mathcal{I}3_{Delivery}$ (from Linz-Vienna) (resource for combination: $Ship400$, costs of combination: €1850 + €320 (remaining part of $\mathcal{I}3_{Delivery}$), optimization potential: €2405).

So compared to the separate solution, all combined solutions provide an optimization potential, but the best solution is the fourth combination ($\mathcal{I}1_{Delivery}$ & $\mathcal{I}2_{Delivery}$ & part of $\mathcal{I}3_{Delivery}$) with an optimization potential of €2405.

2.4 Combining Business Process Instances

The fourth step of the combined-instance approach receives a set of combinable candidates with optimization potential. The goal then is to actually combine activities to permit common use of resources. However, combining activities of different process instances requires a new process element, which we call *Combined Activity* ($\mathcal{X}$). The difference between a *Combinable Activity* ($\mathcal{C}$) and a *Combined Activity* ($\mathcal{X}$) is that $\mathcal{C}$ marks potentially combinable activities at the schema-level, whereas $\mathcal{X}$ represents actually combined activities at the instance-level. Syntactically, $\mathcal{X}$ is addressed by several incoming and outgoing flows from different process instances and receives the data objects and resources of all participating activities. The semantics is that $\mathcal{X}$ consumes a token from every participating process instance, executes the combined activity using the recommended resource(s) and, finally, produces the same amount of tokens and returns them to the process instances. So, $\mathcal{X}$ comprises two constructs that can merge and split tokens from different process instances. For the graphical representation in the process diagram, we recommend two overlapping activities where the front activity is marked with a bold 'X'. A similar element with the same semantics is not available in any other business process modeling language (BPMN, UML, EPC, or YAWL).

For the actual combination, we consider all activities from the set of combinable candidates with optimization potential. However, some candidates may not be reached at all (alternative path is taken) or not reached in time. Thus, we propose a deferred approach for the actual combination. The activity triggering the search for possible combinations has necessarily been activated by an incoming token. We then have to wait for further candidates to be activated. However, the waiting time must be restricted:

– by a predefined amount of time (e.g., 24 h),
– by not explicitly delaying the activity but considering the time before execution as implicit waiting (e.g., if delivery is planned for next Monday then the time in between is considered as implicit waiting), or
– by a given deadline of the activity minus the expected processing time.

In addition, it should be checked regularly whether reachable candidates still exist, since otherwise waiting can be stopped. When a second combinable activity is reached in time, $\mathcal{X}$ is created and replaces the two activities (if necessary an activity is thereby split). If a further candidate receives a token in time, it is also integrated in $\mathcal{X}$. When all possible activities are integrated or the waiting time expired, $\mathcal{X}$ is executed and separately written in the log-file of every process instance (together with further split activities).

In our running example, $\mathcal{I}1_{Delivery}$ is first activated by a token and then waiting for corresponding candidates in other $r\mathcal{I}$. The waiting time is limited by deadline $\mathrm{d1}_{Delivery}$ (12/31/15) minus the expected processing time $\mathrm{p1}_{Delivery}$ (approx. 1 day), so delivery must at latest be started by 12/30/15. We assume that all other delivery activities receive a token in time. The next delivery activity being activated is, presumably, $\mathcal{I}2_{Delivery}$. $\mathcal{I}1_{Delivery}$ and $\mathcal{I}2_{Delivery}$ are fully combined and replaced by an $\mathcal{X}$. Afterwards, also $\mathcal{I}3_{Delivery}$ receives a token. However, this activity must be split so that part of it (delivery from Linz to Vienna) can be integrated in $\mathcal{X}$ and the remaining part (delivery from Vienna to Bratislava) is executed individually (i.e. it is inserted as new activity "Delivery2" after $\mathcal{X}$ in process instance $\mathcal{I}3$).

3 Related Work

Related work is provided by different domains. In the mathematical domain, scheduling problems are investigated and algorithms are provided that calculate the optimal solution. Interesting for our research are dynamic optimization problems, constrained optimization problems, and resource-constrained scheduling problems [2,5,8,9]. If several objectives must be optimized simultaneously, then multi-objective optimizations [6] are applied.

In the business process domain, resource optimization is typically based on goals, constraints, or performance optimizations (e.g. [10]). However, presented approaches either focus on the schema level or on single active process instances sometimes in combination with completed process executions as suggested in [7].

The probably most relevant related research in the business process domain is provided by Pufahl and Weske in [13–15]. The authors synchronize multiple process instances by introducing the concept of batch activities in process modeling and execution. Commonalities with the combined-instance approach are the combination of activities across different processes instances, the goal to thereby save setup and execution costs, and the consideration of resources and their capacities. A difference, however, is the identification of possible instance combinations. The approach by Pufahl and Weske is data-based (i.e., a data view comprises relevant attributes for comparison and only instances with identical values are combinable), whereas the combined-instance approach is constraint-based (i.e., constraints specify how data values must correlate for a possible instance combination). The data-based approach provides better usability, since only relevant attributes must be selected, but the approach is restricted to equality of data attributes. The constraint-based approach is more complex but also supports extensive definitions, thereby enabling the concept of activity splits. Another difference is the synchronization method. Batch activities are not merged to retain single instance autonomy and only the execution of activities is delayed. In contrast, the combined-instance approach provides an actual combination of activities across process instances and therefore introduces the *Combined Activity* element. Advantages of the individual execution are instance autonomy, no additional process element, and the support of batch regions. The advantage of the combined-instance approach is the possibility to dynamically select a different resource for the combined activity. In addition, a further important benefit of the combined-instance approach is the consideration of optimization potential. So, summing up, although both approaches combine activities across different process instances, there are significant differences regarding the chosen methods.

Another similar approach supporting instance-level adaptation is provided by Browne et al. in [3]. The authors introduce the concept of activity crediting to eliminate redundant or overlapping activities at runtime. The paper provides a notion for partial crediting resembling our concept of activity splits, but the suggested approach is restricted to single workflow instances.

For further related research in the business process domain, we refer to the areas of service composition and dynamic resource allocation (e.g. [1]).

Finally, applications and methods have been developed that optimize resource utilization in specific domains like logistics or production (e.g., dynamic logistics process management problems [4,17]). However, our goal is to address resource optimization on a higher level of abstraction, i.e. business processes, so that resources from different domains can be considered within the same business process (e.g., optimization of production and transportation means).

4 Discussion and Case Study

In this section, we provide a detailed discussion of possible resource optimizations and validate the approach by a more comprehensive case study. First of all, we identify all resources with optimization potential. A classification of business resources is given in [16] and consists of the following seven groups:

- **Physical Resources** like machinery, equipment, raw material, or land.
- **Financial Resources** like internal/external funds or financial instruments.
- **Legal Resources** like patents, licenses, copyrights, or trademarks.
- **Relational Resources** including relationships within the company or towards suppliers/customers/competitors/external parties.
- **Human Resources** including their experience, education, and networks.
- **Organizational Resources** like routines, processes, and reputation.
- **Informational Resources** including information about the industry, customers, suppliers, internal processes, and products.

Obviously only the utilization of scarce resource types can be optimized by activity combination. If a resource type is not scarce, it means that the full amount remains available while using any amount of it. For example, combining activities that both use the same patent does not improve the utilization of the patent. So no optimization potential is given for legal, relational, organizational, and informational resources. The remaining resource groups (physical, financial, and human) provide the possibility for resource optimization for most types, e.g., an exception is the education of an employee which is not scarce. Nevertheless, these three resource groups are interesting for our approach and are considered in the following examples:

1. A company produces twelve types of ice-cream cones which are all baked with a distinct batter composition by one machine using different plates. Changing production to another type requires cleaning the machine, changing the plates, and mixing the required batter. Thus, combining orders with same ice-cream cone types improves the workload of the machine (physical resources) and saves cleaning and cooking time (human resources).
   cc: same ice-cream cone type
2. Every employee requiring office supplies forwards the demand to the administration which places the orders. A combination of (not urgent) order activities saves working time (human resources) and may provide the opportunity for a quantity discount (financial resources). *cc*: same office supplies
3. Starting a new project in a company frequently requires that project members install new software. In some cases the activity "Install Software" can be combined so that instead of forcing every project member to install the software locally, the software is once installed on a server and provided as virtual application. This saves all types of resources: working time of the project members (human resources), disk space on local computers (physical resources), and sometimes license fees (financial resources).
   cc: same software (available as virtual application)

The examples presented above show that human resources are often optimized by saving working time whereas financial resources are addressed by saving money. Only physical resources have a greater variation and reach from raw material to production means to disk space. Nevertheless, also physical and human resource optimizations finally result in a financial benefit. The financial benefit of activity combination can emerge from saving setup and/or execution costs, e.g. combining orders in the ice-cream cones example saves setup costs, whereas execution costs are saved by combining transportation in the running example in Section 2.

In addition, we used the combined-instance approach in an actual business environment to evaluate the practical utility of the suggested concepts. We therefore considered the support process of a hardware manufacturer, software developer and system solution provider for building security, service billing, alerting, multimedia and IT services in health care. The business process schema of the support process is shown in Fig. 2. An instance of the support process is created whenever the company receives a support request of a customer. A service ticket is then opened and the actual problem is analyzed. If the problem is already solved, the service ticket is closed. However, if the problem is not solved, then the ticket is forwarded to a technician who decides whether it is a software or hardware problem. If it is a hardware problem, then a replacement device is ordered or created and the firmware is installed. Afterwards, the replacement unit is delivered to the customer. However, if it is a software problem, then a software ticket is created followed by an error analysis and correction in the software component. After correcting the error, it is checked whether remote maintenance is possible. If this is the case, then the software is updated remotely, otherwise a technician visits the customer to update the firmware on-site.

The support process is instantiated several thousand times a year making activity combination and resource optimization an interesting improvement opportunity. In sum, we identified the following combinable activities:

- *Order or Create Replacement Device:* Replacement devices are, e.g., Media-Boxes, PT-Terminals, or Multimedia Telephones. Optimization potential is given by combining orders to external companies (lower delivery costs, quantity discount) and by combining internal production orders (saving working time and setup time of production means). *cc*: same external company (for external orders); same product (for internal orders)
- *Deliver Replacement Device to Customer:* Optimization potential is given by combining different deliveries to the same customer. *cc*: same customer
- *Error Analysis and Correction in Software Component:* Optimization potential is given by combing analysis and correction with other errors in the same software component. Similar errors can be assigned to the same software developer reducing familiarization effort and time for quality control. *cc*: same software component
- *Plan Visit at Customer:* Optimization potential is given by combining visits of nearby customers (similar to our running example in Section 2), thereby

saving traveling time of the technician and transportation resources. *cc*: see *cc* of delivery activity in Section 2

– *Update Firmware at Customer:* Optimization potential is given by combining firmware updates (also of other products), e.g., updates to correct errors with planned updates comprising new features. *cc*: same customer

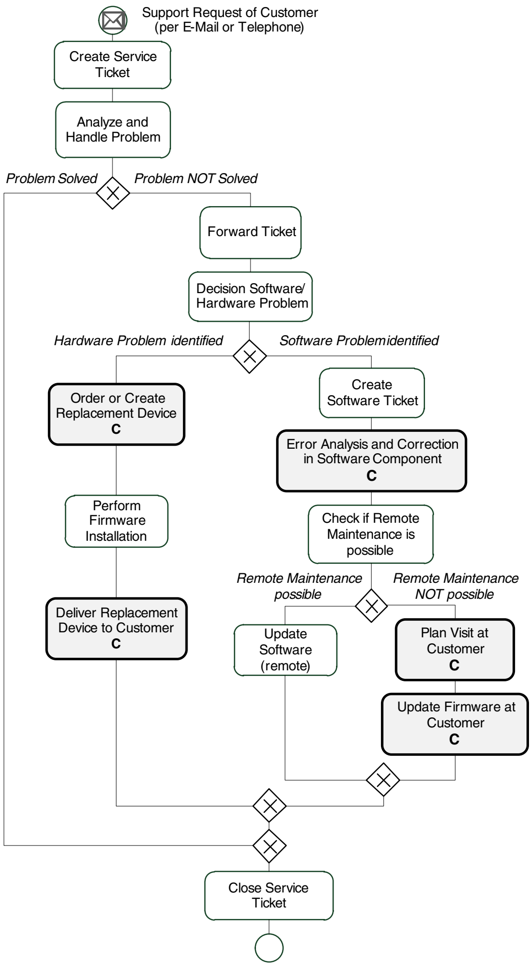
Support Request of Customer (per E-Mail or Telephone)
Create Service Ticket
Analyze and Handle Problem
Problem Solved
Problem NOT Solved
Forward Ticket
Decision Software/ Hardware Problem
Hardware Problem identified
Software Problem identified
Order or Create Replacement Device C
Create Software Ticket
Error Analysis and Correction in Software Component C
Perform Firmware Installation
Check if Remote Maintenance is possible
Remote Maintenance possible
Remote Maintenance NOT possible
Deliver Replacement Device to Customer C
Update Software (remote)
Plan Visit at Customer C
Update Firmware at Customer C
Close Service Ticket

Fig. 2. Case Study: Support Process

5 Conclusion

In this paper, we suggest a novel approach for resource optimization. The key idea of the proposed *Combined-Instance Approach* is to combine corresponding activities of different process instances and to thereby improve resource utilization (e.g., by combining delivery activities of different orders). The approach consists of four steps and various concepts to increase resource efficiency, e.g., calculation of optimization potential, dynamic resource allocation for combined activities, and activity splitting to partly combine activities. The applicability of the combined-instance approach was demonstrated by applying it to a running example and by using it within a case study. We further expect that using this approach provides a financial benefit for many companies.

Our future goals are to formally define activity splits, to provide exception handling for combined activities, and to combine similar activities derived from different process schemas (e.g., based on the identification of similarities [11]). We will further extend the case study by considering other processes of the company as, e.g., an *order execution process*.

References

1. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering **33**(6), 369–384 (2007)
2. Biró, M.: Object-oriented interaction in resource constrained scheduling. Information Processing Letters **36**(2), 65–67 (1990)
3. Browne, E.D., Schrefl, M., Warren, J.R.: Activity crediting in distributed workflow environments. In: ICEIS, vol.3, pp. 245–253 (2004)
4. Chow, H.K., Choy, K.L., Lee, W.B.: A dynamic logistics process knowledge-based system - an RFID multi-agent approach. Know-Based Syst **20**(4), 357–372 (2007)
5. Cruz, C., González, J., Pelta, D.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing **15**(7), 1427–1448 (2011)
6. Deb, K.: Multi-objective optimization. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, pp. 403–449. Springer, US (2014)
7. Ernst, M.: Method and apparatus for dynamic optimization of business processes managed by a computer system (1999). US Patent Number 5890133
8. Golden, B., Raghavan, S., Wasil, E.: The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces Series. Springer, US (2008)
9. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. EJOR **207**(1), 1–14 (2010)
10. Huang, Z., van der Aalst, W., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. Data & Knowledge Engineering **70**(1), 127–145 (2011)
11. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 319–334. Springer, Heidelberg (2012)
12. Natschläger, C., Bögl, A., Geist, V.: Optimizing resource utilization by combining running business process instances. In: Toumani, F., Pernici, B., Grigori, D.e.a. (eds.) ICSOC 2014 workshops and satellite events. LNCS, Springer (2014)
13. Pufahl, L., Herzberg, N., Meyer, A., Weske, M.: Flexible batch configuration in business processes based on events. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 63–78. Springer, Heidelberg (2014)

14. Pufahl, L., Meyer, A., Weske, M.: Batch regions: Process instance synchronization based on data. In: EDOC, pp. 150–159. IEEE (2014)
15. 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)
16. Seppänen, M., Mäkinen, S.: Towards classification of resources for the business model concept. International Journal of Management Concepts and Philosophy **2**(4), 389–404 (2007)
17. Wang, Y., Caron, F., Vanthienen, J., Huang, L., Guo, Y.: Acquiring logistics process intelligence: Methodology and an application for a chinese bulk port. Expert Systems with Applications, pp. 195–209 (2014)